evropský
sociální
fond v ČR

EVROPSKÁ UNIE

MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY

2007–13
OP Vzdělávání
pro konkurenceschopnost

UNIVERSITAS MASARYKIANA BRUNENSIS

INVESTICE
DO ROZVOJE
VZDĚLÁVÁNÍ

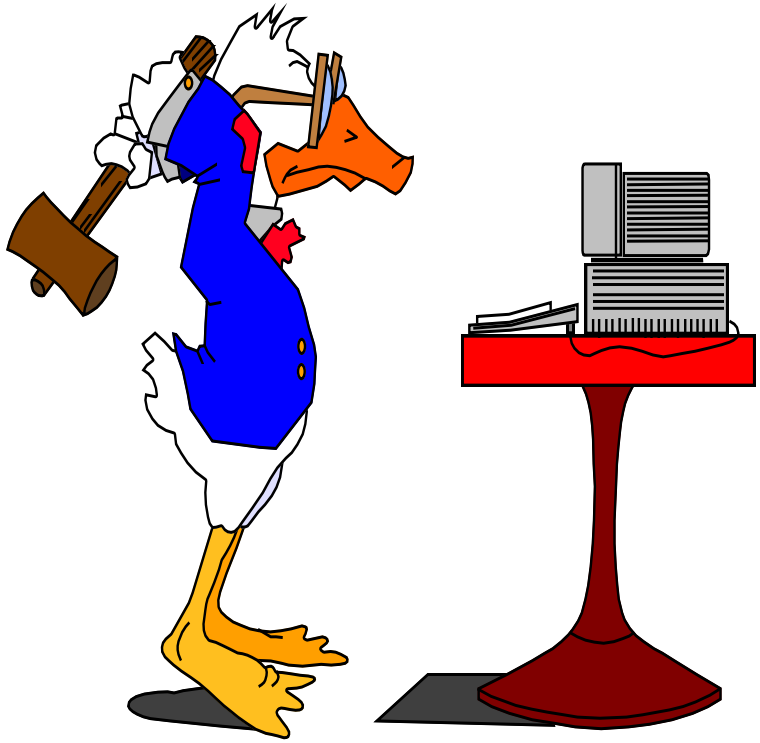
Data Mining I

Martin Řezáč

2012

Obsah:

1. Úvod do data miningu



Co je to Data Mining?

- Data mining (DM), nebo také dolování z dat či vytěžování dat, je analytická metodologie získávání netriviálních skrytých a **potenciálně užitečných informací**.

Aplikace

- Bankovnictví: schvalování úvěrů/kreditních karet
  - Predikce dobrých zákazníků.
- Pojišťovnictví: schvalování pojistných smluv
  - Odhad pravděpodobnosti pojistné události/výše škody.
- CRM (marketing):
  - Identifikace zákazníků, kteří mají v úmyslu přejít ke konkurenci.
  - Cross-selling.
  - Up-selling.
- Cílený marketing:
  - Identifikace pravděpodobných respondentů na nabídku.
- Detekce fraudu: telekomunikace, finanční transakce, pojistné podvody
  - Online/offline identifikace podvodného chování.

Aplikace

- Medicína: efektivita léčebné péče
  - Analýza pacientovy historie (předchozí nemoci a jejich průběh): nalezení vztahu mezi nemocemi.
- Farmacie: identifikace nových léků
- Vědecká analýza dat:
  - Identifikace nových galaxií.
- Design webových stránek:
  - Nalezení vztahu návštěvníka stránek a příslušná změna podoby stránek.

Aplikace

- Rozpoznávání psaného textu, řeči, obrázků.
- Supermarkety
  - Identifikace současně nakupovaného zboží
- Průmysl:
  - automatické přenastavení ovládacích prvků při změně parametrů procesu.
- Sport:
  - NBA-optimalizace herní strategie
- další…

Aplikace - Rozmístění zboží v supermarketech

- Cíl: identifikovat zboží, které je nakupováno souběžně dostatečným množstvím zákazníků.

- Výsledek: Jestliže zákazník nakupuje dětské pleny a mléko, pak si velmi pravděpodobně koupí i pivo.

- Jedna z možných interpretací:

- Správné interpretace výsledků analýz je schopen jen zkušený analytik.

Ukázka pracovních nabídek (rok 2011):

http://kariera.homecredit.cz/cz/analytik-rizeni-rizik/job.html?id=493

Analytik řízení rizik

Náplň práce:
• zodpovědnost za vývoj rizikových ukazatelů na svěřeném portfoliu produktů (kreditní karty a hotovostní úvěry)
• identifikace rizik v portfoliu daného produktu
•odborné posouzení nestandardních žádostí o úvěr

Požadujeme:
• VŠ, případně SŠ vzdělání (výhodou zaměření na **matematiku** nebo ekonomii)
• velmi dobrá znalost MS Office (zejména MS Excel)
• výhodou zkušenost ve finančním sektoru
• **analytické myšlení**
• výhodou znalost **statistických metod a matematiky** a zkušenosti se skóringovými modely
• velmi dobré komunikační, vyjednávací a argumentační schopnosti
• asertivita, smysl pro přesnost, odolnost vůči stresu

http://kariera.homecredit.cz/cz/specialista-analytik-strategie-vymahani/job.html?id=487

Specialista - analytik strategie vymáhání

Náplň práce:
• měření vymáhacího procesu
• analýzy a podpora pro vymáhání
• testování a podpora změn
• Reporting

Požadujeme:
• VŠ vzdělání – nejlépe **matematika**, ekonomie, informatika
• pokročilou znalost MS Excel a SQL
• výhodou znalost SAS a VBA
• středně pokročilou znalost AJ
• velkou výhodou praxe v oblasti analýz, reportingu, orientace v oblasti vymáhání
• aktivní přístup
• chuť využít své zkušenosti a zapojit se do nových věcí
• **analytické myšlení a logické uvažování**
• komunikační schopnosti, samostatnost při řešení úkolů, spolehlivost, pečlivost, odolnost vůči stresu, flexibilita

Ukázka pracovních nabídek (rok 2011):

KB

https://www.recrutement-societegenerale.com/jpapps/kbLocal/jobs/jobview.jsp?TOKEN=ff7ec84a3a440fd12c7c783ebf&requestno=RQ00056954

Analytik Business Intelligence

Náplň práce:
• Zajišťovat činnosti spojené s podporou a rozvojem systémů Business Intelligence,
• V rámci vývoje zajišťovat analýzy uživatelských požadavků, definovat procesy, navrhovat controllingové postupy, provádět datové modelování, navrhovat transformační procesy, připravovat uživatelské rozhraní a metodicky připravovat nasazení nových nástrojů

Požadujeme:
• VŠ nebo SŠ s praxí ve finančním sektoru
• Znalost **finanční matematiky**
• Znalost oblasti bankovních aplikací výhodou, znalost centrálního bankovního systému KBI vítána
• Výborná znalost **MS Office**
• Znalost **SQL**
• Znalost účetnictví vítána
• Znalost controllingu / MIS / performance managementu
• Znalost AJ
• Pečlivost, spolehlivost, schopnost komunikace a týmové spolupráce
• Samostatnost, odolnost vůči stresu, flexibilita

KB

https://www.recrutement-societegenerale.com/jpapps/kbLocal/jobs/jobview.jsp?TOKEN=ff7ec84a3a440fd12c7c783ebf&requestno=RQ00054440

Skoringový analytik

Náplň práce:
• Vývoj skóringových modelů, tvorba metodiky vývoje modelů
• Zpětné testování skóringových funkcí a tvorba metodiky pro jejich použití
• Statistický reporting rizikových parametrů portfolia a spolupráce na jeho vývoji

Požadujeme:
• Vysokoškolské vzdělání **matematického** nebo ekonomického zaměření **se znalostí statistických metod**
• Analytické schopnosti, tvořivost, samostatnost a zodpovědnost
• Schopnost komunikovat v angličtině (zejména písemně)
• Zkušenosti se statistickým software formou skriptů (výhodou SAS a S-Plus) a prací s databázemi (SQL)
• Znalost základních ekonomických pojmů (časová hodnota peněz, opravné položky…)
• Znalost metodiky Basel 2 (konkrétně modelování LGD, CCF, EAD) výhodou
• Základní znalost bankovních produktů výhodou

Ukázka pracovních nabídek (rok 2011):

Kooperativa
Vienna Insurance Group

http://www.onrea.com/pd/176680227?brand=g2&rcm=24045507&sourcebrand=g2&source=3&exportRCM=24045507&trackingBrand=www.koop.cz

ANALYTIK DATA MININGU

Náplň práce:

• Tvorba predikčních modelů a příprava dat pro ně
• Prezentace výsledků odborně i laickým uživatelům
• Organizace datových podkladů pro modely na úrovni zadávání dalším útvarům
• Spoluvytváření firemního data skladu na straně uživatelů
• Průběžná aplikace pro firmu nových metod data miningu, reportingu a čištění dat

Požadujeme:

• VŠ – obor: ekonometrie, **matematická statistika, pojistná matematika** a podobně (možno i student postgraduálního studia)
• Znalost vícerozměrných **statistických metod**
• Znalost alespoň jednoho ze statistických SW SPSS, KXEN, Rapidminer, SAS (**SAS - výhodou**)
• Alespoň základní znalost relačních databází a SQL
• Znalost MS Excel alespoň na úrovni maker / VBA (VBA – výhodou)
• **Analytické** a komunikační **schopnosti**
• Znalost AJ na alespoň technické úrovni
• Praxe v oblasti tvorby predikčních modelů / data miningu / BI – výhodou
• Zkušenost v pojišťovnictví, telekomunikacích nebo finančním sektoru – výhodou
• Zkušenost s CRM / Camapaign managementem – výhodou

T-Mobile

http://careers.peopleclick.com/careerscp/client_tmobile/external/cs/jobDetails.do?functionName=getJobDetail&jobPostId=107551&localeCode=cs#

Specialista zákaznických analýz

Náplň práce:

• Analyzovat a interpretovat DWH data.
• Komunikovat se zadavateli analýz.
• Zdokonalovat strukturu zdrojových dat v souladu s potřebami datových analýz.
• Využívat analytické nástroje s ohledem na potřeby a rozvoj analýz.
• Hledat nové přístupy v oblasti datových analýz.

Požadujeme:

• VŠ/SŠ vzdělání ekonomického, technického, **matematického** směru
• Zkušenost s analytickým SW - např. SPSS, SAS, Access, SQL
• Excel - vynikající znalost (databázové funkce, makra, formuláře)
• Spolehlivost, pečlivost
• Analytické myšlení
• Komunikace

Ukázka aktuálních pracovních nabídek (17.1.2012):

www.jobs.cz

Klíčové slovo „SAS“ :

- RISK ANALYTIK – PROGRAMÁTOR
- Pojistně technický a datový analytik -úsek pojištění vozidel
- Risk Data Analyst
- Pricing Specialist
- Fraud Analytik/-čka produktů spotřebního financování
- Statistician (Brno)
- …

Klíčové slovo „matematika“ :

- POJISTNÝ MATEMATIK – JUNIOR
- DWH/BI Specialista/tka
- Specialista ALM / pojistný matematik
- …

ČSOB
Požadovaná kvalifikace a další požadavky
- zkušenost s **dataminingem**, **statistickými prediktivními metodami**, či neuronovými sítěmi
- výhodou zkušenost s používáním nástrojů pro odhalování podvodů a nástrojů pro navrhování a řízení strategií
- **VŠ**/SŠ vzdělání **matematického**, či technického nebo **ekonomického** zaměření
- dobrá znalost **MS Office, EXCEL, ACCESS, SQL**, výborná znalost práce s PC,
- znalost statistických systému, či systémů pro dolování dat (**SAS, SPSS/Clementine**, …)
- komunikativní znalost anglického jazyka nezbytná
- znalost vnitřních informačních systémů banky výhodou

AXA
We Require:
- university degree in **statistics or mathematics**
- English language fluently written and spoken
- excellent skills in MS Office (Excel, **SQL programmer for data mining**)
- experience in insurance and ability to use Pretium or **SAS** is advantage
- **analytical and statistical skills**

Ukázka aktuálních pracovních nabídek (17.1.2012):

www.linkedin.com

- Modeling, Scoring, & Analysis Sr. Manager - CBNA Risk Management (Long Island City, NY)
- Head of level – Decision Science / Modelling (London, UK)
- Senior Credit Risk Analyst, Basel II Modeling (Detroit)
- Statistician (Dallas)
- + další **TISÍCE** volných míst požadujících matematické/ekonomické vzdělání, znalost statistiky, SQL, SASu

CITY

- Master Degree with specialization in **Statistics, Economics, Finance**, Engineering or other quantitative fields, PhD preferred.
- 10+ years hand-on **statistical risk modeling** experience in financial industry with demonstrated proficiency in scorecard development.
- Diversified modeling experience in Fraud and/or Mortgage modeling strongly preferred.
- In-depth understanding of regulatory requirements, and proven experience in interacting with regulators and internal auditors.
- Strong communication and project management skills.

Santander

- Graduate degree in **Statistics, Economics**, Operations Research or other quantitative discipline required.
- Familiarity with logistic regression models, segmentation and variable reduction techniques, hypothesis testing, non-parametric testing, design of experiments, ANOVA, CHAID analysis and linear regression.
- **SAS: SAS base, SAS/STAT, PROC SQL, SAS Macro programming,** using SQL and SAS to extract data from different data sources. Ability to merge, concatenate, import/export datasets, clean data and check for data consistency and accuracy.

Data mining a princip indukce

- Dedukce zachovává platné vztahy:
  1. Koně jsou savci.
  2. Všichni savci mají plíce.
  3. Proto platí, že všichni koně mají plíce.

- Indukce přidává informace:
  1. Všichni doposud pozorovaní koně mají plíce.
  2. Proto platí, že všichni koně mají plíce.

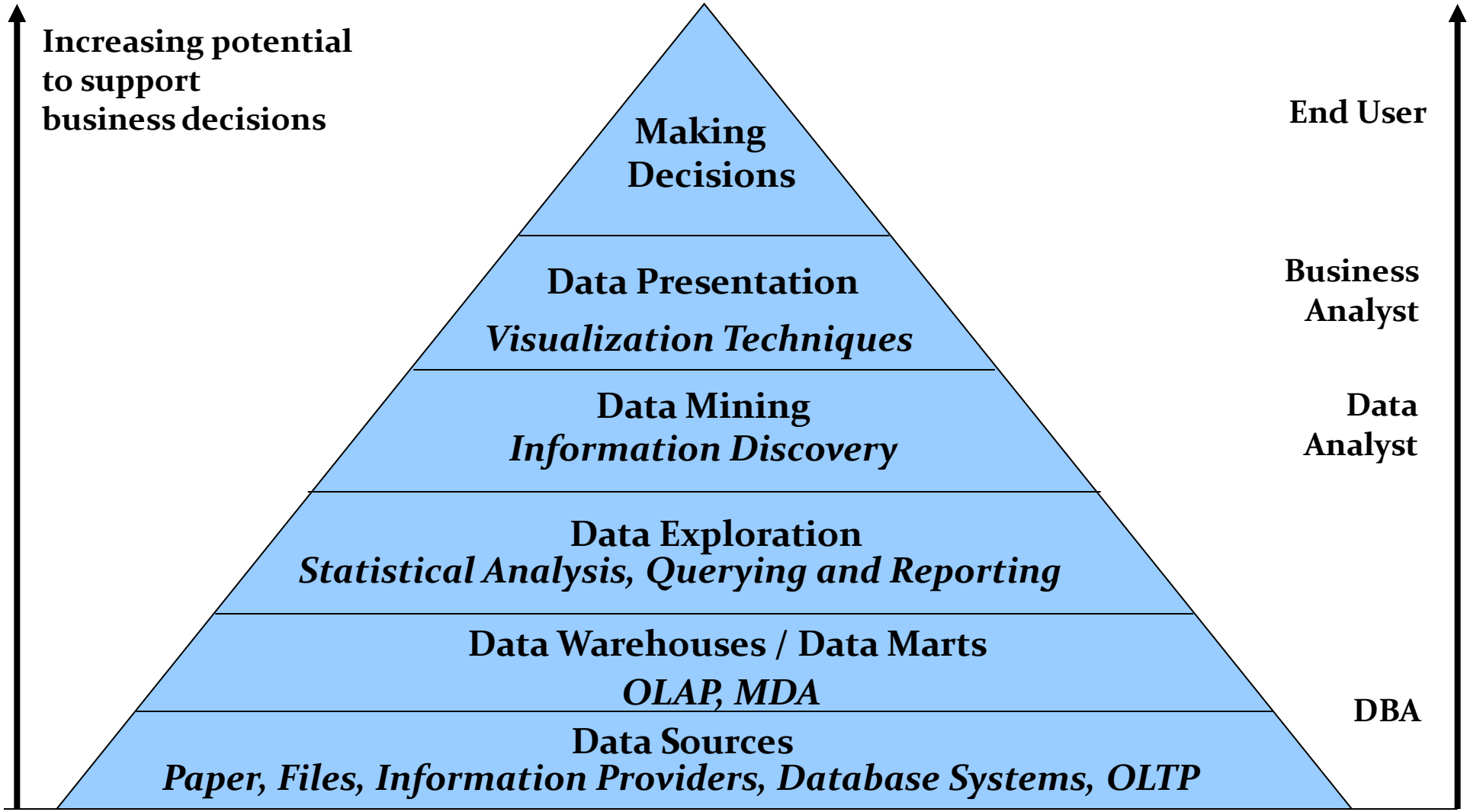
Problém s indukcí

- Z platných faktů můžeme vyvodit nepravdivé tvrzení (model).

- Příklad:
  - Evropské labutě jsou bílé
  - Indukce: „Labutě jsou bílé" jakožto obecné pravidlo.
  - Objevením Austrálie se objevili i černé labutě…
  - Problém: množina pozorování nebyla náhodná a tudíž reprezentativní.

http://cs.wikipedia.org/wiki/Labu%C5%A5_%C4%8Dern%C3%A1

Data mining –podpora business rozhodování



Historie názvu

1960 Data Fishing, Data Dredging (bagrování):

- užíváno statistiky

1989 Knowledge Discovery (KD, KDD):

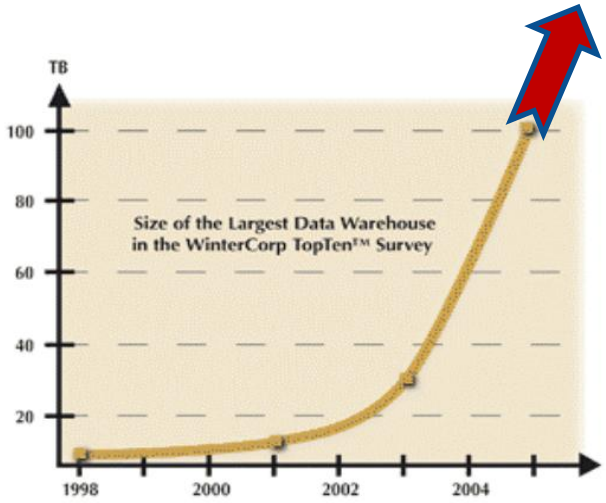
- užíváno komunitou zabývající se umělou inteligencí a strojovým učením

1990 Data Mining (DM):

- užíváno v komerční sféře a databázové komunitě

Další názvy: Data Archaeology, Information Harvesting, Information Discovery, Knowledge Extraction, …

Data mining – nutnost?



Největší světové databáze v r. 2005:

- Max Planck Inst. for Meteorology ~ 222 TB
- Yahoo ~ 100 TB
- AT&T ~ 94 TB

V roce 2008:

- Max Planck Inst. for Meteorology ~ 6000 TB
- Yahoo ~ 2000 TB

Další velké databáze:

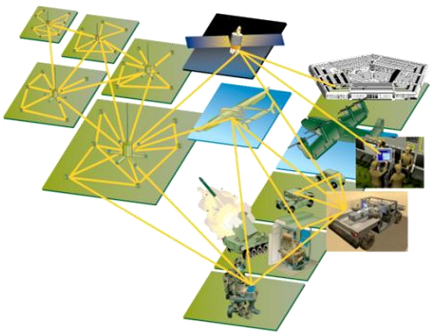
- CIA, Amazon, Google, YouTube, AT&T,...

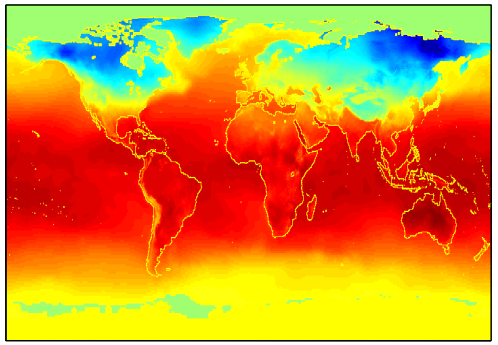
Více např. na http://www.focus.com/fyi/10-largest-databases-in-the-world/

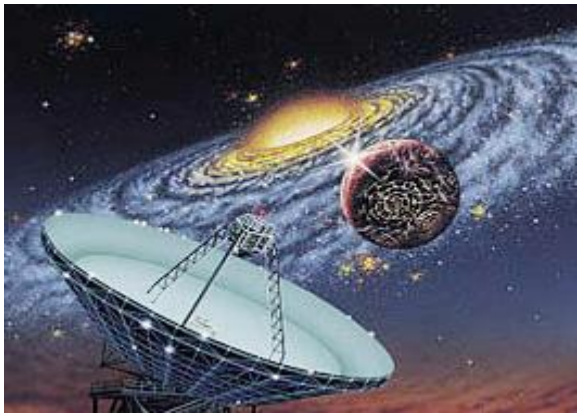
Data mining – nutnost?

- Terabytes -- 10^12 bytes: data obchodních řetězců, bank,…
- Petabytes -- 10^15 bytes: geografická data
- Exabytes -- 10^18 bytes: národní databáze zdravotních záznamů
- Zettabytes -- 10^21 bytes: databáze meteo-snímků
- Zottabytes -- 10^24 bytes: video-databáze

Data mining – nutnost?







Proč data mining? Proč dnes?

- Data jsou produkována.
- Data jsou skladována.
- Výpočetní síla je dostupná.
- Výpočetní síla je cenově dostupná.
- Konkurenční tlak je velice silný.
- Komerční produkty (DM software) jsou k dispozici.

Data mining vs. Statistická analýza

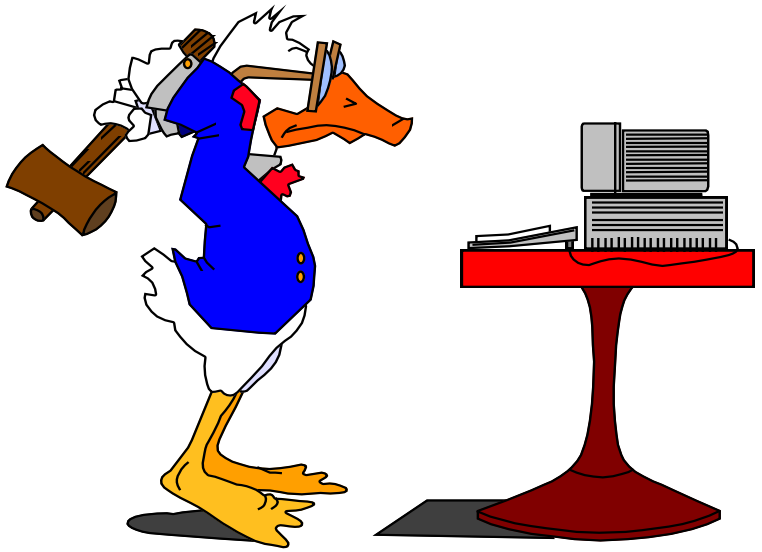
- **Data Mining**
  - Původně vyvinuto pro expertní systémy automaticky řešící zadané problémy.
  - Neklade takový důraz na přesné porozumění použité metody.
  - Pokud něco dává smysl, pak to použijme!
  - Žádné předpoklady o datech.
  - Funguje i pro velmi rozsáhlá data.
  - Vyžaduje porozumění problému z datovému a business pohledu.
- **Statistická analýza**
  - Testuje se statistická korektnost modelu.
    - Jsou statistické předpoklady modelu splněny?
  - Testování hypotéz.
  - Intervalové odhady.
  - Pracuje se s výběrem hodnot.
  - Standardní metody nejsou optimalizovány pro rozsáhlá data.
  - Vyžaduje pokročilé statistické znalosti.

Data mining

- Proces (polo-) automatické analýzy (rozsáhlých) databází k identifikaci vztahů, které jsou:
  - validní: platí na nových datech s určitou jistotou obecné platnosti
  - nové: doposud neznámé
  - užitečné: dají se v praxi nějak použít
  - srozumitelné: (vždy) se nalezený vztah dá nějak vysvětlit

Data mining není:

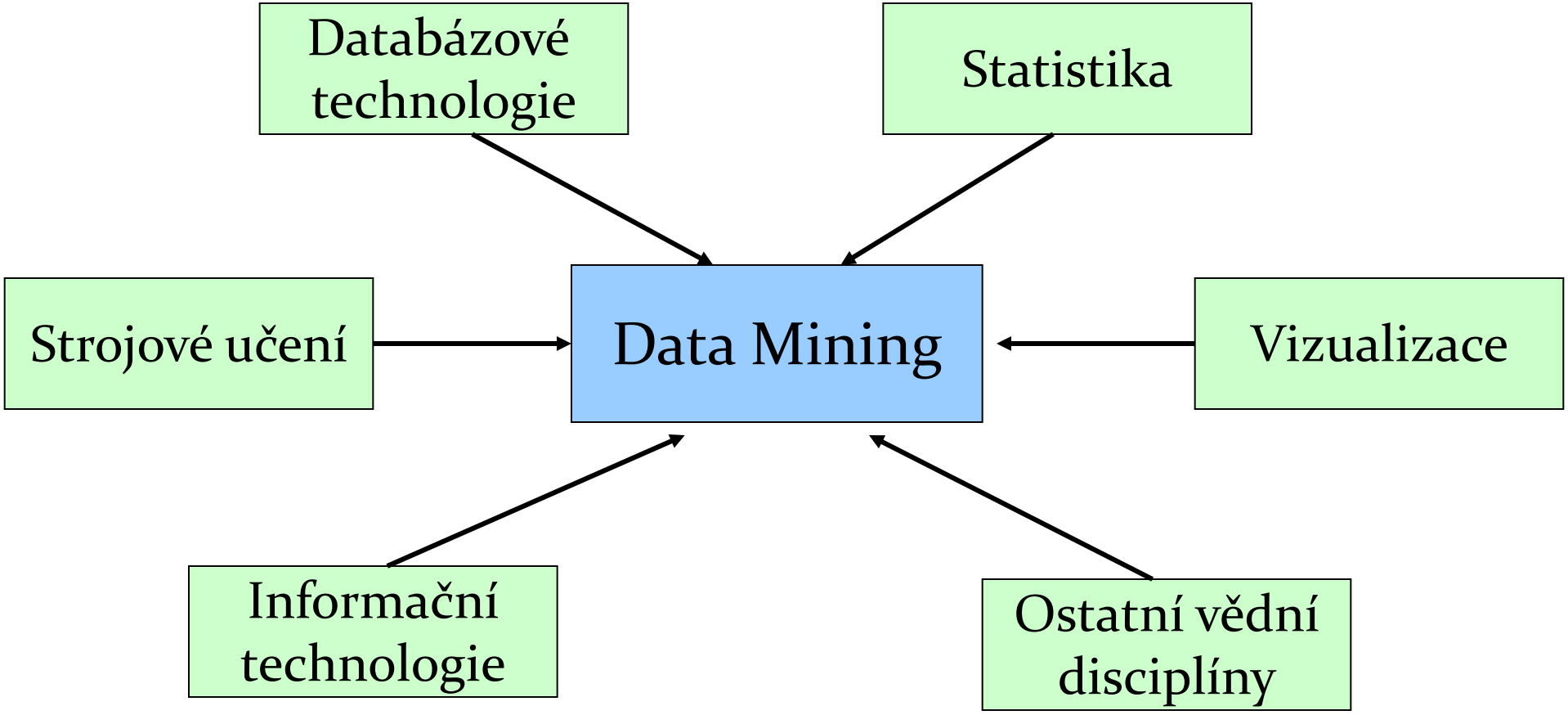
- **Brutální hromadné zpracování dat.**
- **Slepé použití algoritmů.**
- **Hledání vztahů tam, kde žádné neexistují.**



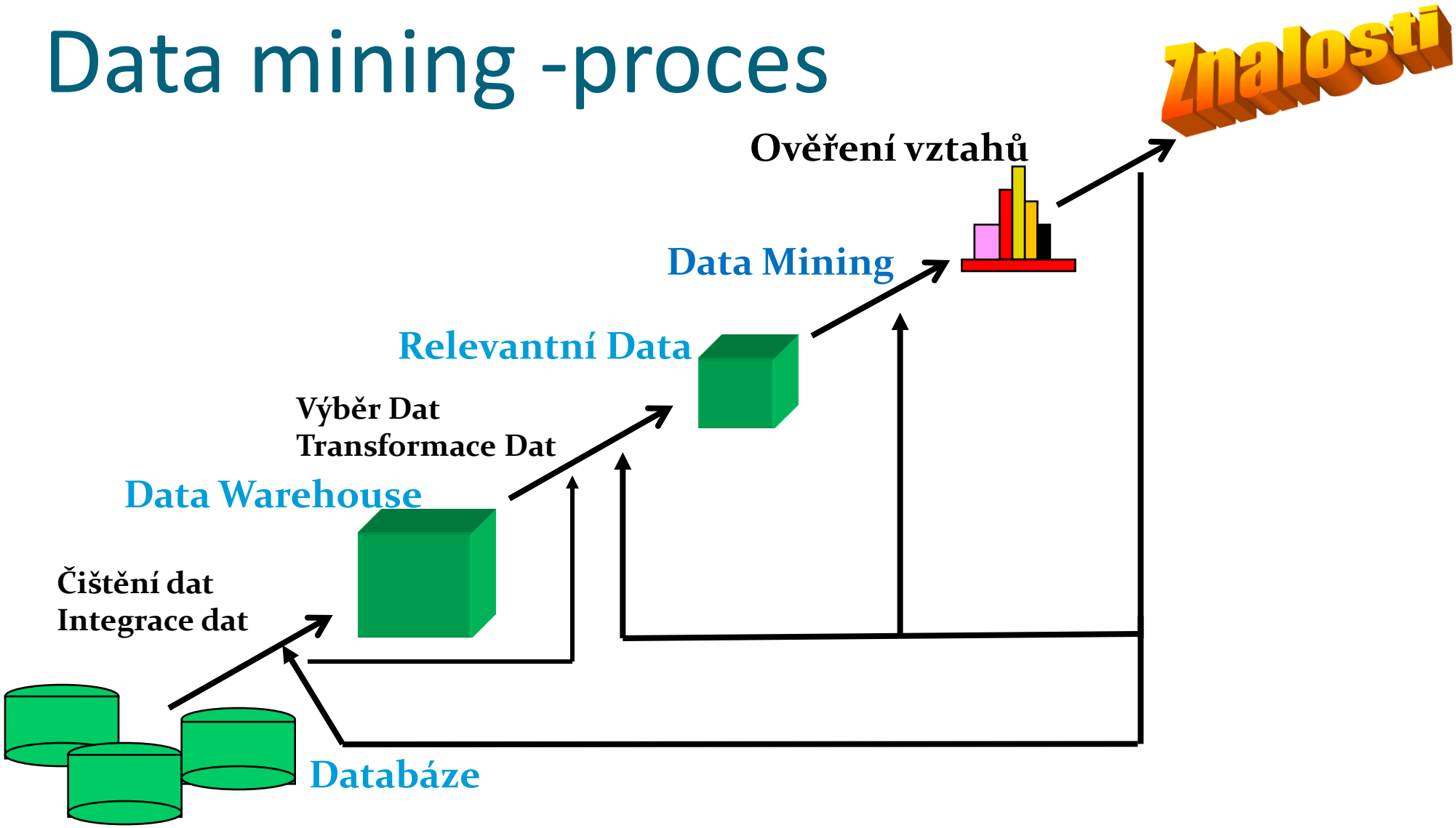
Známé ≠ Zajímavé

- Zajímavé jsou ty vztahy, které se liší od obecných očekávání.
- Data mining se vyplácí právě díky objevování dosud neznámých a překvapivých vztahů.

Vztah s ostatními disciplínami

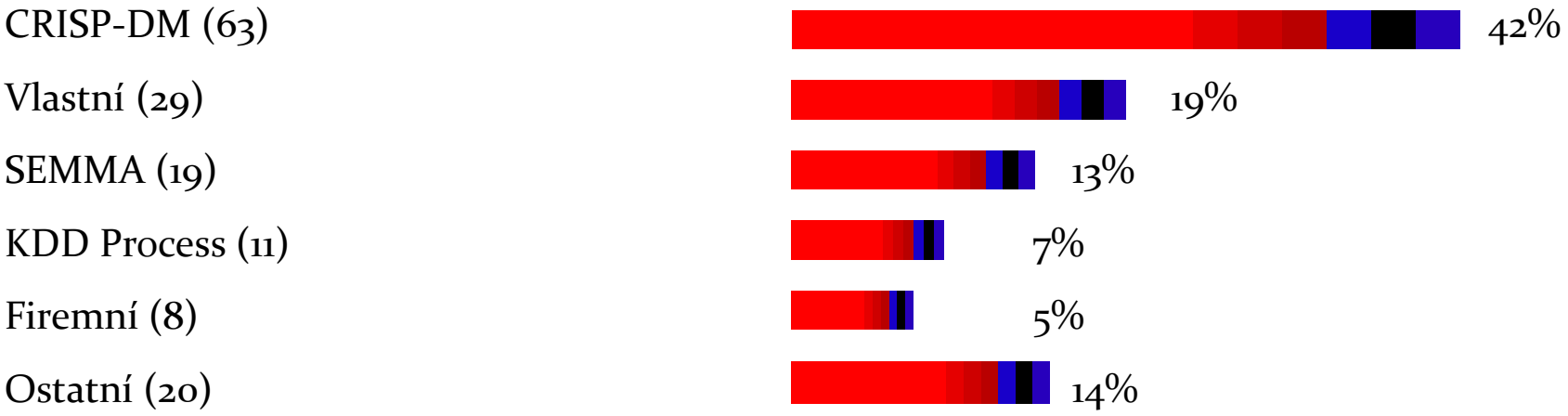


Data mining -proces



Data Mining Methodology (2007)

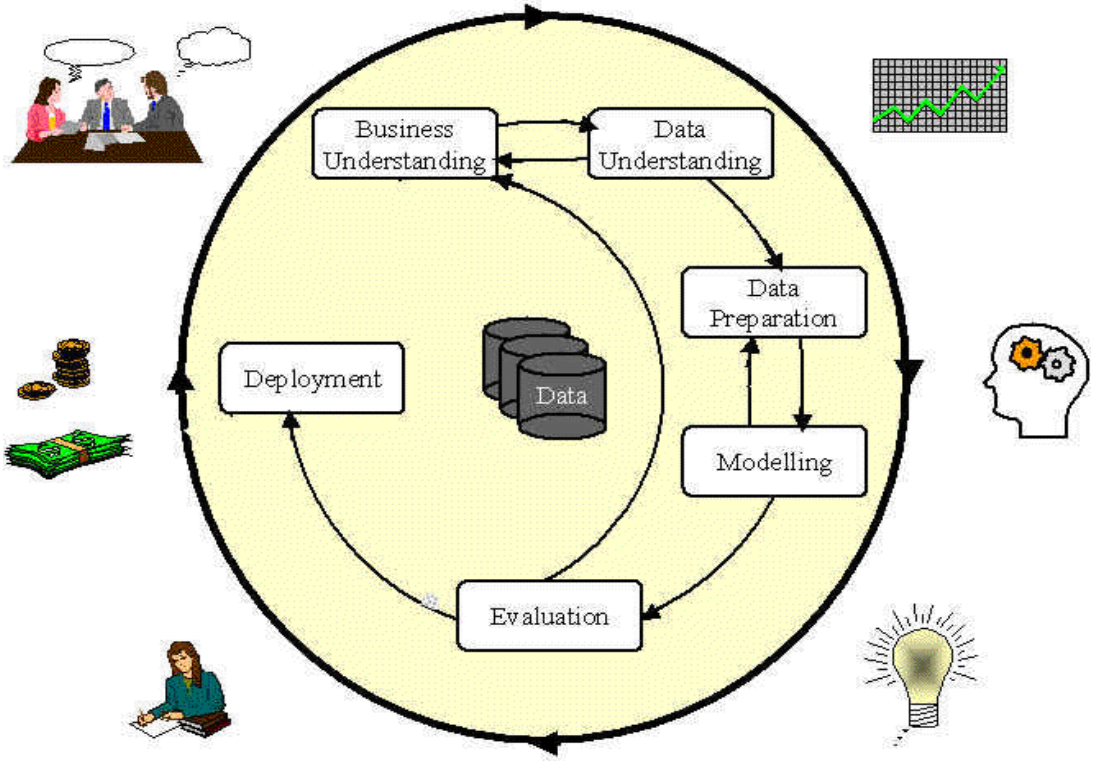
Kterou metodologii používáte pro data mining?



CRISP-DM

(CRoss Industry Standard Process for Data Mining)

1. pochopení obchodních souvislostí
2. pochopení dat
3. příprava dat
4. modelování
5. vyhodnocení modelu
6. nasazení modelu do obchodního procesu



http://community.udayton.edu/provost/it/training/documents/SPSS_CRISPWPlr.pdf

SEMMA

(Sample, Explore, Modify, Model, Assess)

• **Sample** - identifikovat vhodná učící data, určit odpovídající rozsah dat, a to jak z pohledu časového okna tak i z pohledu počtu případů. Dále se doporučuje rozdělit data na 3 skupiny:

Trénovací – využívá se pro vývoj modelu.

Validační – využívá se pro vyhodnocení modelu a pro prevenci proti přeučení (over fitting) modelu.

Testovací – využívá se pro finální vyhodnocení modelu. Zajímá nás především jak dobře se model chová na datech disjunktních s daty, na kterých byl model vyvinut.

• **Explore** - připravit popisné statistiky, které poskytnou základní představu o obsahu a kvalitě podkladových dat. Pomocí vizualizačních technik odhalit skryté trendy a závislosti v datech.

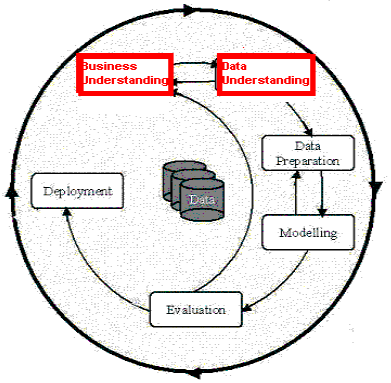
• **Modify** - na základě předchozího kroku konsolidovat data a odvodit nové proměnné. Následně transformovat data do tvaru vhodného pro modelování.

• **Model** - vytvořit příslušný model. Mezi často používané techniky patří např. neuronové sítě, rozhodovací stromy, logistické modely.

• **Assess** - vyhodnotit úspěšnost modelu a případně implementovat model do praxe.

Fáze DM procesu (1 & 2)

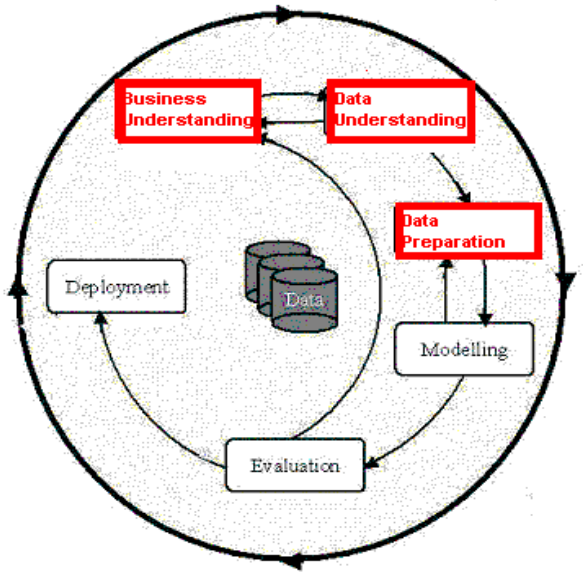
- Porozumění obchodu (Business Understanding):
  - Stanovení business cílů.
  - Stanovení data miningových cílů.
  - Statnovení kriterií úspěchu.



- Porozumění datům (Data Understanding):
  - Průzkum dat a ověření jejich kvality.
  - Nalezení odlehlých hodnot.

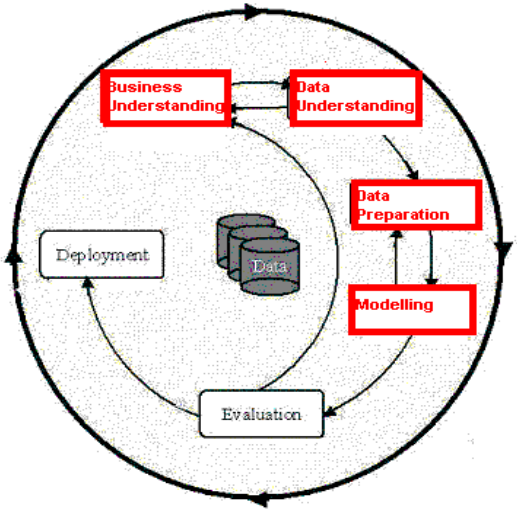
Fáze DM procesu (3)

- Příprava dat (Data preparation):
  - Obvykle zabírá přes 90% celkové času.
    - Sběr dat
    - Konsolidace a čištění
      - Vazební tabulky, agregace, chybějící hodnoty,…
    - Selekce
      - Ignorování neužitečných dat?
      - Odlehlá pozorování?
      - Výběr dat?
      - Vizualizační nástroje.
    - Transformace – vytváření nových odvozených proměnných



Fáze DM Procesu (4)

- Modelování (Model building)
  - Výběr vhodných modelovacích technik závisí na stanovených data miningových cílech.
  - Modelování je většinou iterační proces propojený s přípravou dat
  - Rozdílný přístup pro „*supervised*“ a „*unsupervised learning*“

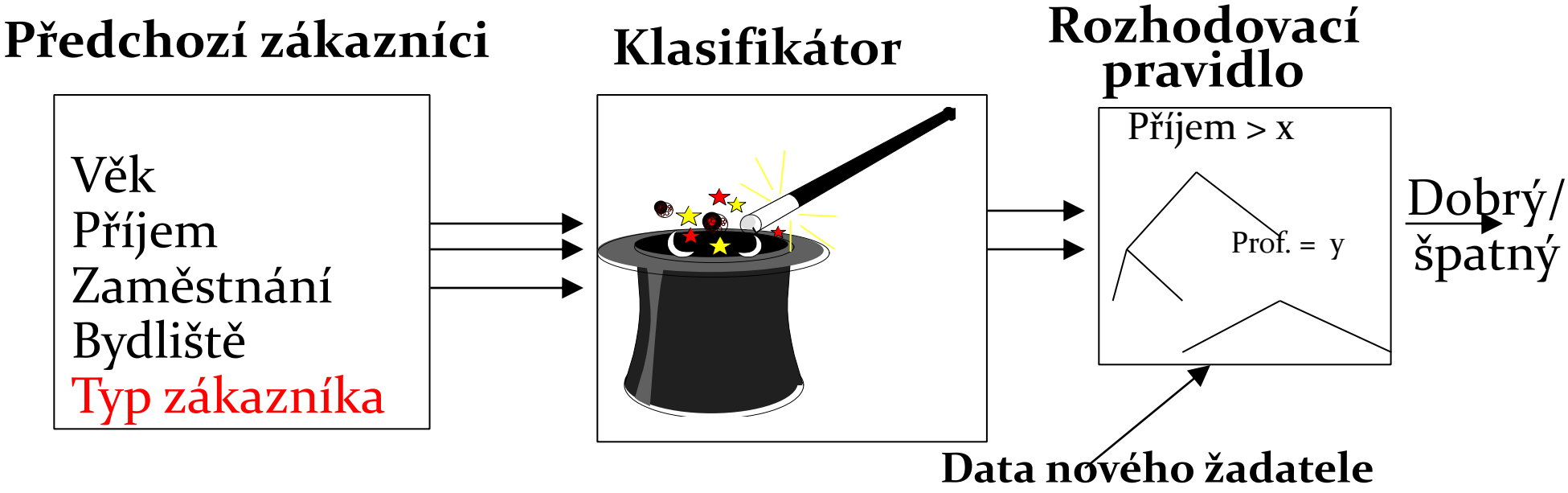


Základní přístupy k modelování

- Prediktivní: jde o matematický model předpovídající (s určitou přesností) budoucí hodnotu/chování nějaké veličiny (entity).
  - Regrese/ Klasifikace
  - Analýza časových řad
- Deskriptivní: jde o matematický model popisující historické události a předpokládané nebo reálné vazby mezi nimi.
  - Klastrová (shluková) analýza
  - Asociační pravidla
  - Detekce deviací/zlomů
  - Faktorová analýza / analýza hlavních komponent

Klasifikace

- Na základě známých údajů o „starých“ zákaznících a jejich platební morálce máme predikovat platební způsobilost nového žadatele o úvěr.



Klasifikační metody

- Cíl: Predikovat třídu Ci = f(x1, x2, .. Xn)
- Regrese: (lineární nebo polynomiální)
  - a*x1 + b*x2 + c = Ci
- Metody nejbližšího souseda (KNN)
- Rozhodovací stromy
- Pravděpodobnostní modely (GLM) – např. logistická regrese.
- Diskriminační analýza (LDA,...)
- Neuronové sítě
- Support vector machines (SVM)
- Bayesovské modely

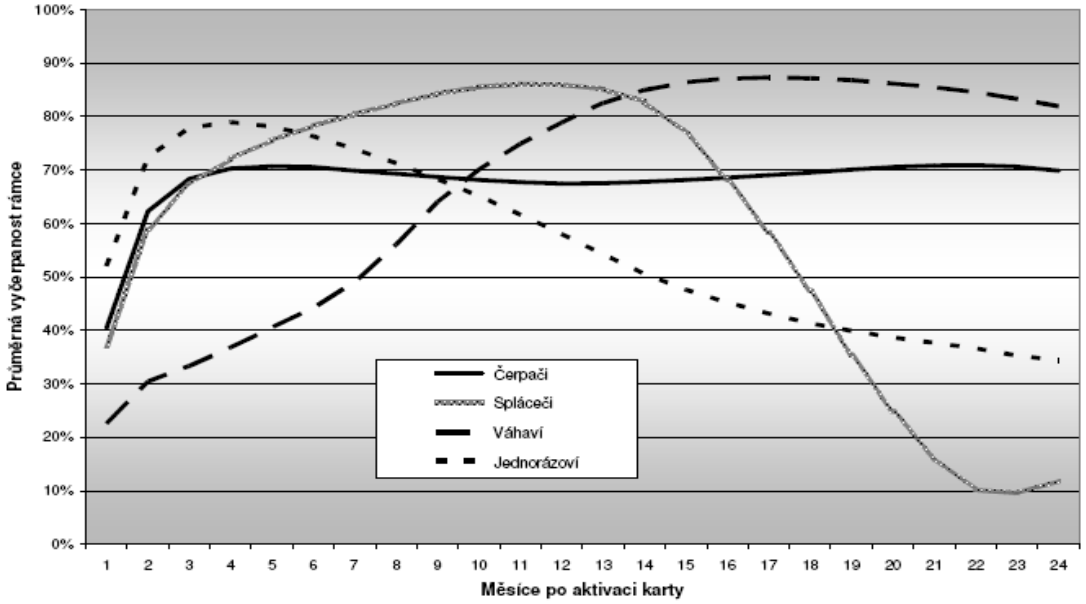
Deskriptivní modelování

- Základním cílem je získání ucelených a snadno srozumitelných informací z dostupných dat.

- Někdy součástí průzkumové (explorační) analýzy předcházející prediktivnímu modelování, někdy je vytvoření deskriptivního modelu hlavním cílem DM projektu.

Klastrová analýza

- Máme nalézt skupiny/ klastry stávajících zákazníků na základě platební historie tak, aby podobní klienti byli ve stejné skupině/ klastru.
- Základní požadavek: Kvalitní míra podobnosti (http://cs.wikipedia.org/wiki/Shluková_analýza).



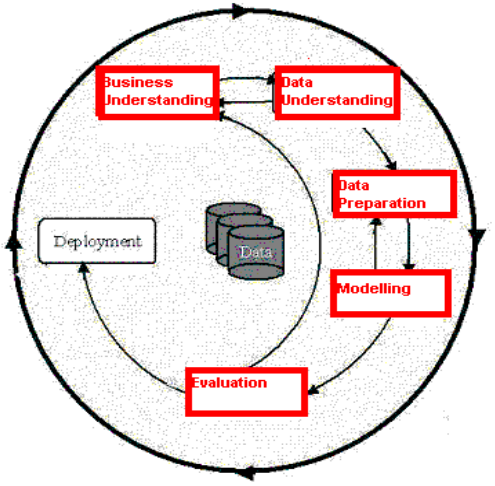
Zdroj: NEPIL, M. *Data mining v praxi.* Brno : MU v Brně, 2007. s 25-38.

Supervised vs. unsupervised learning

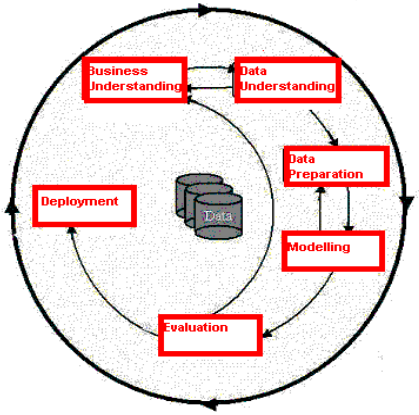
- Supervised learning:
  - Supervize: Data (pozorování, měření, atp.) jsou označena předem definovanými/známými třídami.
  - Nová/testovací data jsou následně rozřazena do těchto tříd.
  - Z pohledu kauzality daný model definuje vztah mezi vstupními daty a daty výstupními.
- Unsupervised learning:
  - Předem nejsou definované žádné třídy.
  - Pro daná data je cílem prokázat existenci nějakých tříd.
  - Z pohledu kauzality jsou všechna data chápána jako výstupní. Modelujeme závislost daných dat na jakýchsi neznámých skrytých proměnných.

Fáze DM Procesu (5)

- Vyhodnocení modelu (Model Evaluation):
  - Evaluace modelu: jak se chová na testovacích datech.
  - Metody a kritéria závisí na typu modelu:
    - Např. koincidenční matice pro klasifikační modely, průměrná chyba pro regresní modely,...
  - Interpretace modelu: důležitost a obtížnost interpretace značně závisí na zvolené modelovacím algoritmu.



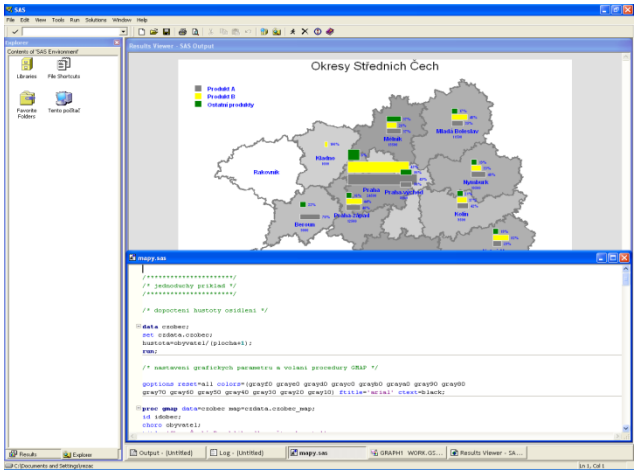
Fáze DM Procesu (6)



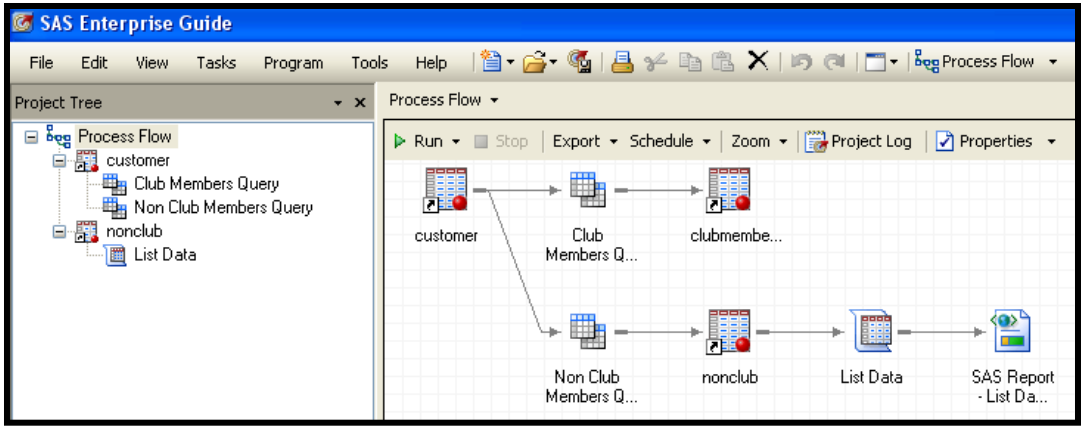
- Nasazení do praxe (Deployment)
  - Je třeba určit, jak mají být výsledky využity.
  - Kdo je bude využívat?
  - Jak často budou využívány?
- Nasazení data miningových výsledků pomocí:
  - Skórování databáze.
  - Využití výsledků pomocí obchodních pravidel.
  - Interaktivní on-line scoring.
  - …

SAS - stručné seznámení

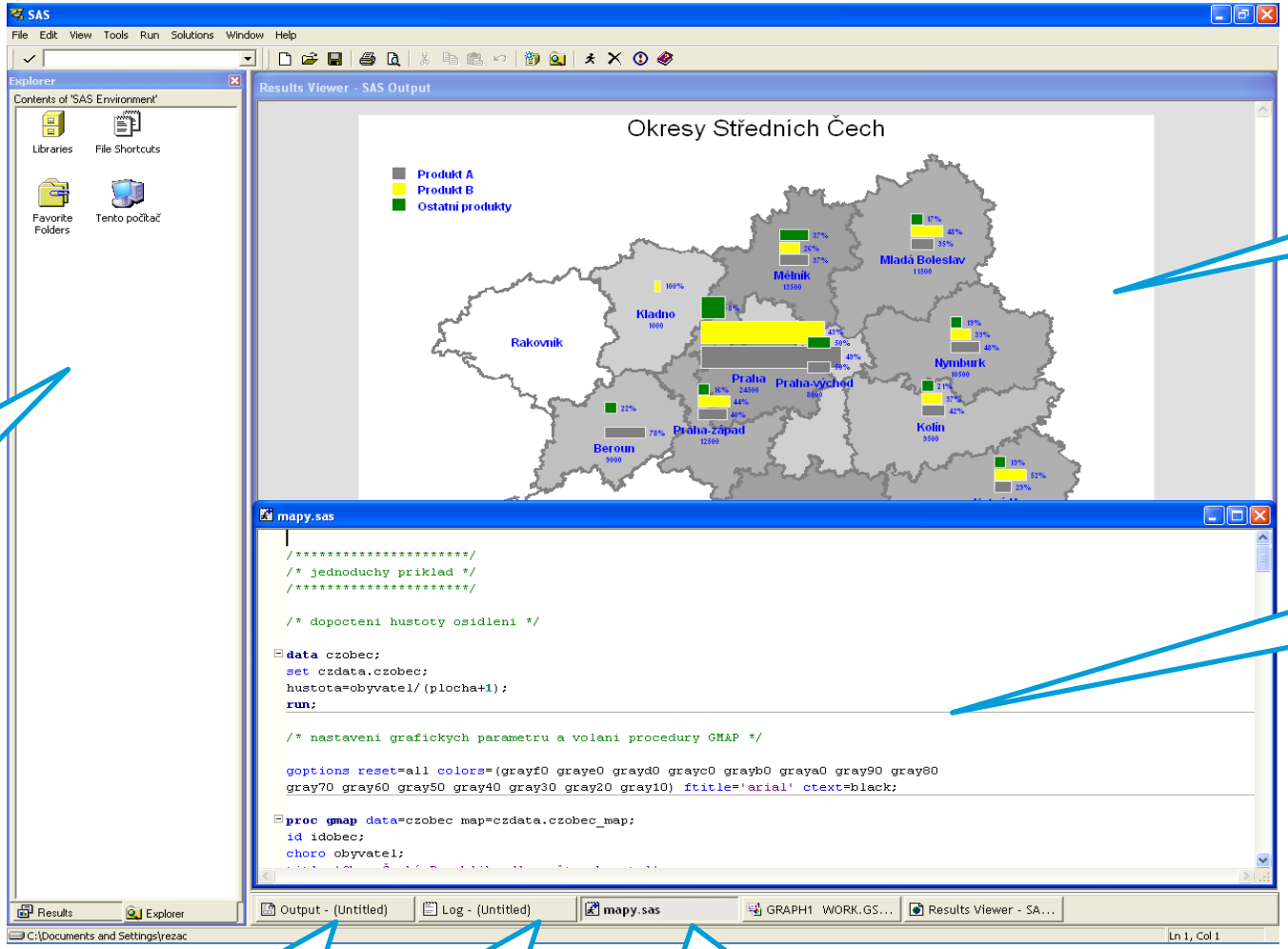
- 2 základní SAS rozhraní:
  - SAS windowing environment



  - SAS Enterprise Guide (GUI)



SAS - stručné seznámení



SAS Output

SAS Explorer window

Program editor window

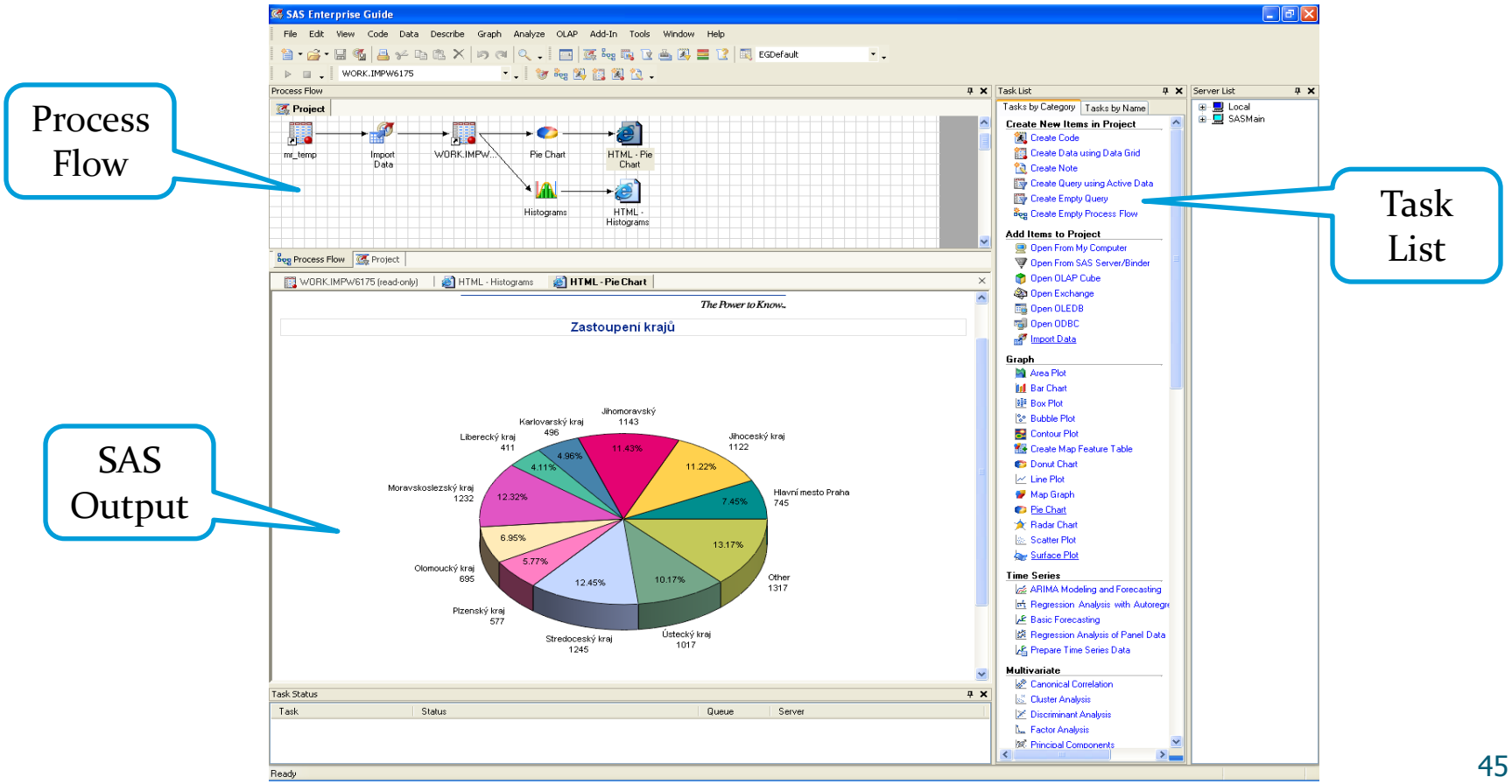
Output tab

Log tab

Editor tab

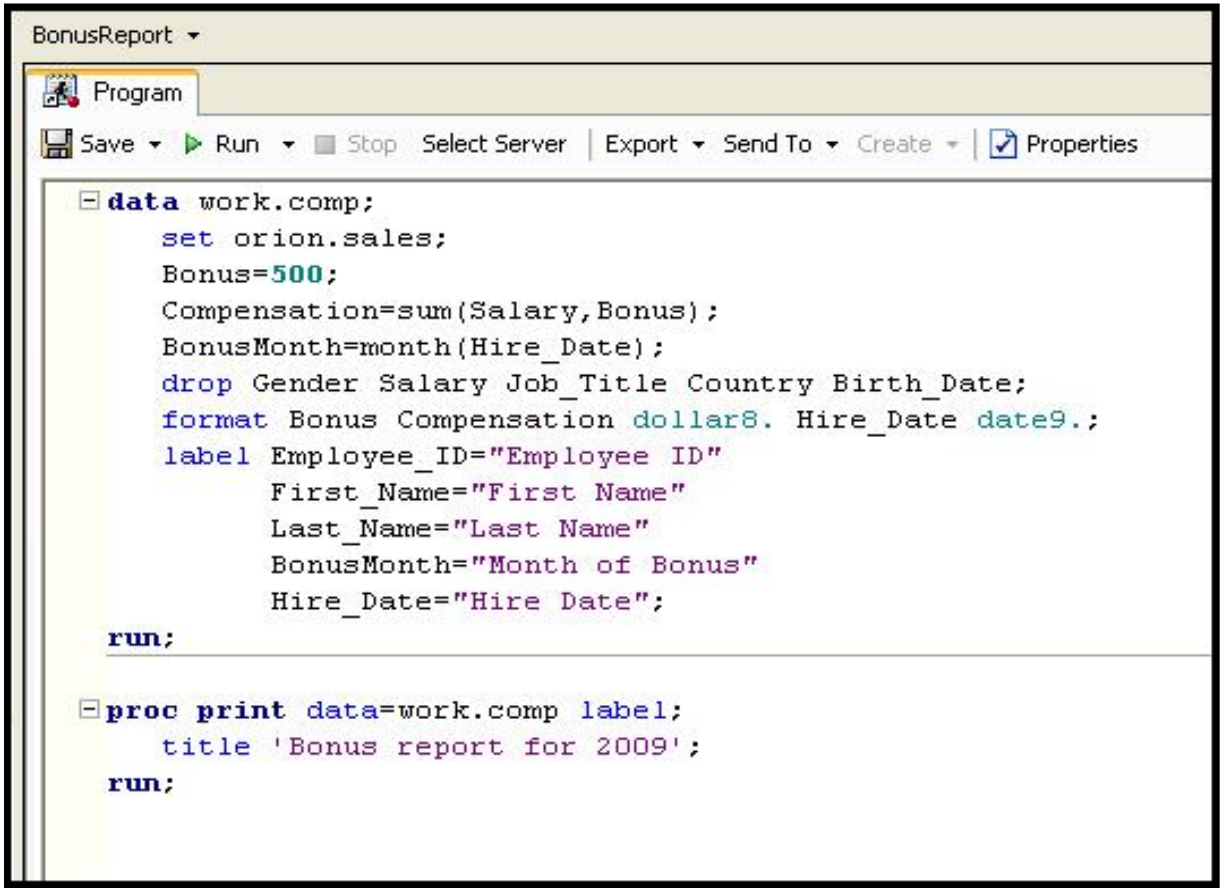
SAS - stručné seznámení

- Pomocí klikání a přetahování myší je budován procesní tok.



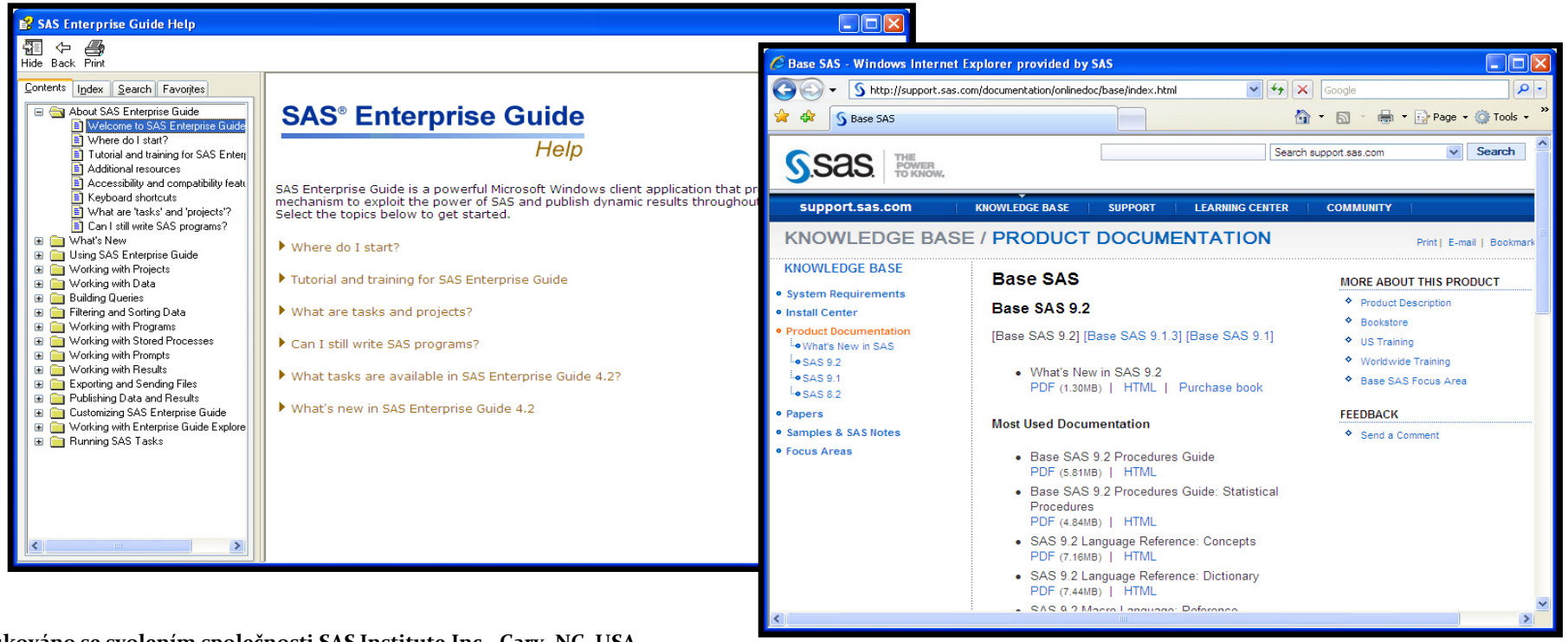
SAS Enterprise Guide (EG) Interface

- EG automaticky generuje kód, který možné dále editovat



SAS Help

- Use the SAS Enterprise Guide Help facility or SAS OnlineDoc for additional direction on SAS Enterprise Guide or the SAS programming language. Go to support.sas.com and select
- **Product Documentation ⇨ Base SAS**.



- **SAS používají např.:**

Více na http://www.sas.com/offices/europe/czech/reference/

SAS na webu

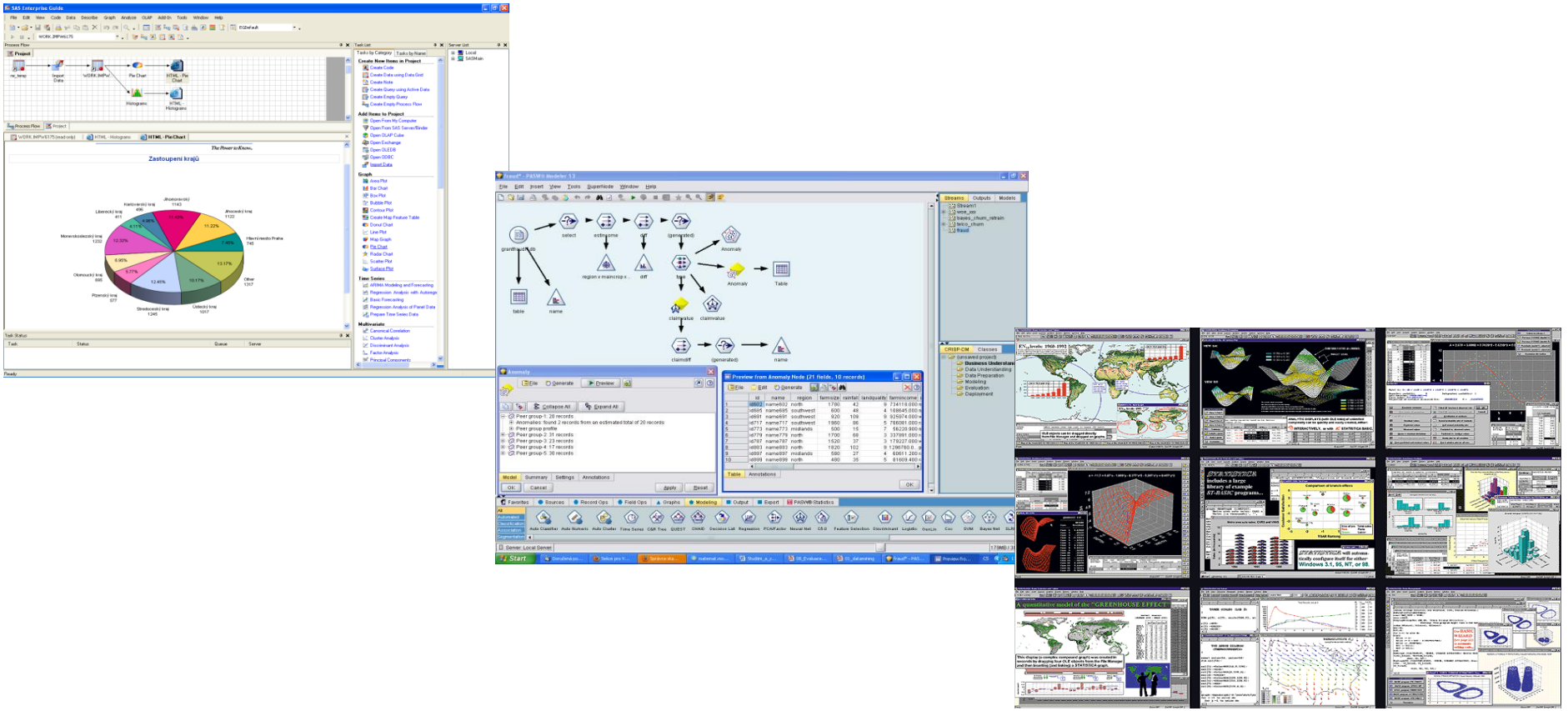
Michal Kulich: *Malý manuál uživatele SASu*
http://www.karlin.mff.cuni.cz/~kulich/sas/SASMain.html

Phil Spector: *An Introduction to the SAS System*
http://www.stat.berkeley.edu/classes/s100/sas.pdf

Patric McLeod : *Introduction to SAS 9*
http://www.unt.edu/rss/class/sas1/

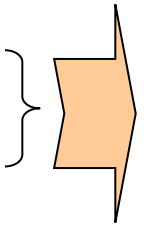
http://en.wikipedia.org/wiki/SAS_%28software%29

2. Software, základy práce v SAS



Data miningový software

- Cca 20 až 30 dodavatelů
- Hlavní hráči na trhu:
  - Clementine,
  - IBM's Intelligent Miner,
  - SGI's MineSet,
  - SAS's Enterprise Miner.



IBM SPSS Modeler
(PASW Modeler)

Více např. na:

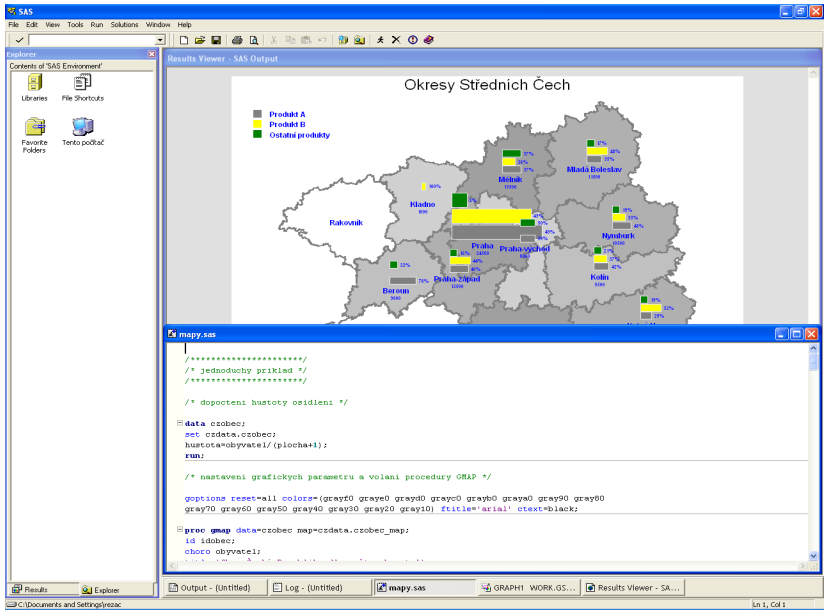
- http://www.kdnuggets.com/software/
- http://www.dmoz.org/Computers/Software/Databases/Data_Mining/Public_Domain_Software/
- http://dir.yahoo.com/Business_and_Economy/Business_to_Business/Computers/Software/Databases/Data_Mining/

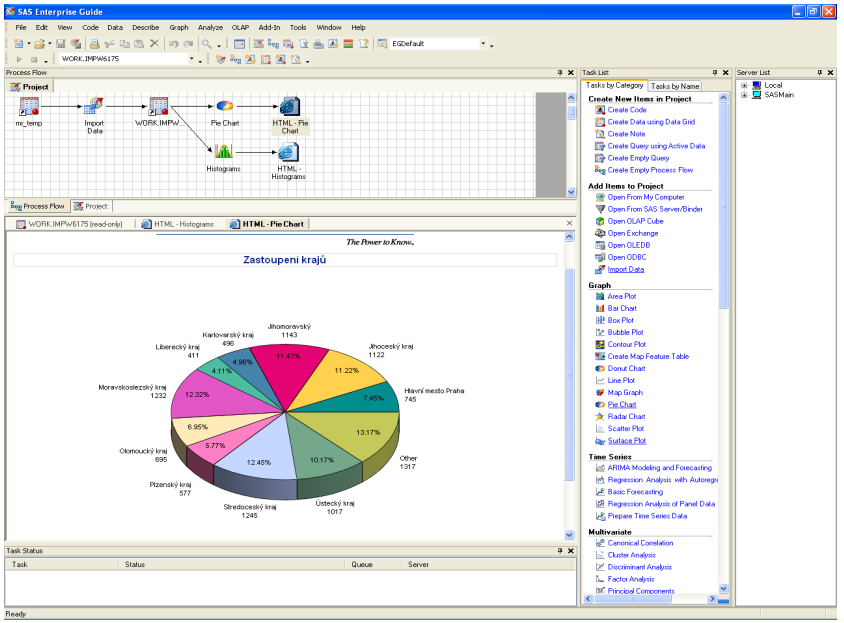
Software (další)

AcaStat	GAUSS	MRDCL	RATS	StatsDirect
ADaMSoft	GAUSS	NCSS	RKWard[4]	Statistix
Analyse-it	GenStat	OpenEpi	SalStat	SYSTAT
ASReml	Golden Helix	Origin	SAS	The Unscrambler
Auguri	gretl	Ox programming language	SOCR	UNISTAT
BioStat	JMP	OxMetrics	Stata	VisualStat
BrightStat	MacAnova	Origin	Statgraphics	Winpepi
Dataplot	Mathematica	Partek	STATISTICA	WinSPC
EasyReg	Matlab	Primer	StatIt	XLStat
Epi Info	MedCalc	PSPP	StatPlus	XploRe
EViews	modelQED	R	SPlus	
Excel	Minitab	R Commander[4]	SPSS	

Software - SAS

- sas www.sas.com





SAS

- Společnost SAS Institute
  - Vznik 1976 v univerzitním prostředí
    - Dnes:největší soukromá softwarová společnost na světě (více než 11.000 zaměstnanců)
    - přes 45.000 instalací
    - cca 9 milionů uživatelů ve 118 zemích
    - v USA okolo 1.000 akademických zákazníků (SAS používá většina vyšších a vysokých škol a výzkumných pracovišť)

SAS

Soutěž o nejlepší studentskou práci

- lze přihlásit bakalářskou, diplomovou, dizertační, semestrální nebo ročníkovou práci využívající SAS.
- **1. místo** – letenky dle vlastního výběru v hodnotě 15.000 Kč.

SAS.GLOBALFORUM
2012 Orlando Florida
April 22-25, 2012

Ročník 2010:

- **1. místo - Účast na SAS Global Forum v Las Vegas.** Výherce měl hrazenou letenku, ubytování a účastnický poplatek.

http://www.sas.com/offices/europe/czech/academic/soutez.html

http://www.sas.com/offices/europe/czech/academic/poster.html

SAS

Podpora studentů

- Možnost rozšíření licence na domácí instalace pro studenty
- SAS Fellowship Program – software zdarma pro diplomku či dizertaci
- Zadávání a vedení diplomových prací
- Sdílení informací, zkušeností či příkladů v uživatelských skupinách

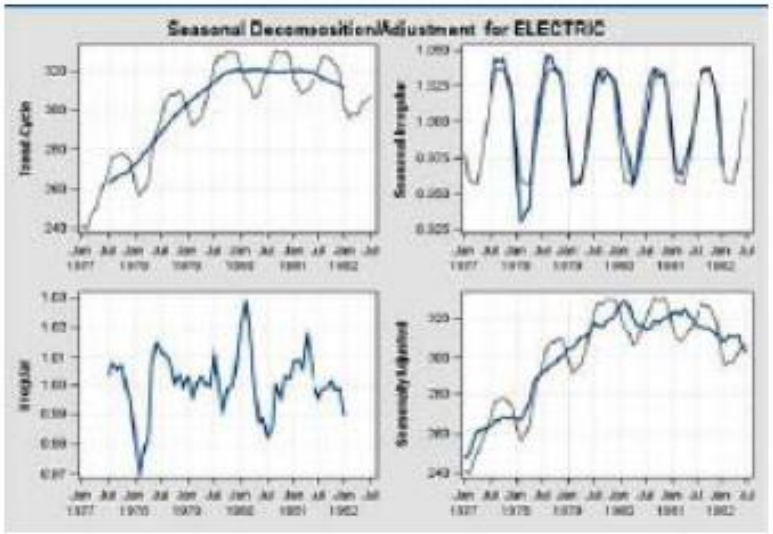
- Interaktivní moduly nebo programovací prostředí
  - Statistická analýza
  - Matice
  - Časové řady
  - Operační výzkum
  - Kontrola kvality

SAS

- Statistická analýza:
  - Popisná statistika
  - Analýza kontingenčních (frekvenčních) tabulek
  - Regresní, korelační, kovarianční analýza
  - Logistická regrese
  - Analýza rozptylu
  - Testování hypotéz
  - Diskriminační analýza
  - Shluková analýza
  - Analýza přežití
  - …

SAS

- Analýza časových řad:
  - Regresní modely
  - Modely se sezónními faktory
  - Autoregresní modely
  - ARIMA
  - Metody exponenciálního vyrovnání
  - …



SAS

- Více o SASu: http://www.sas.com/offices/europe/czech/

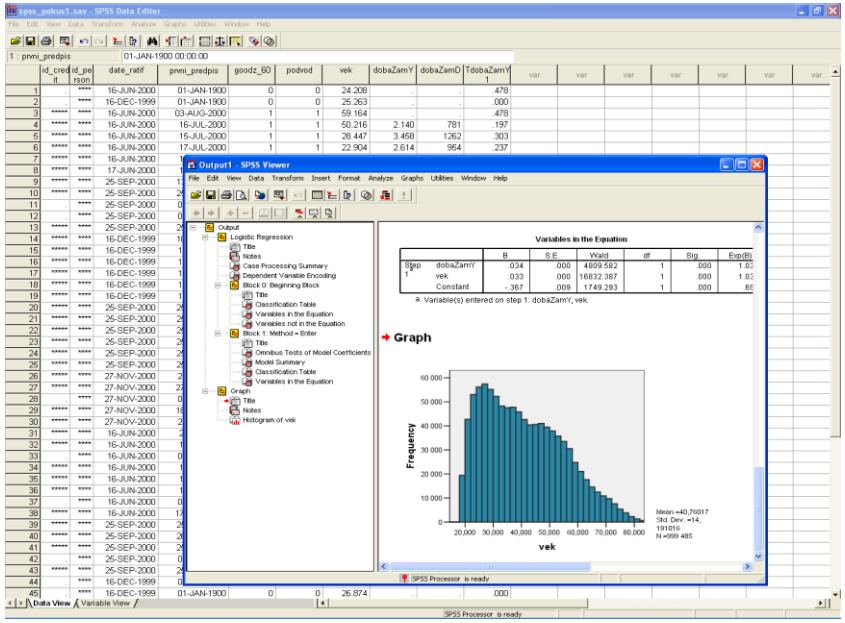
- (neúplný) seznam komerčních společností využívající SAS:
http://www.sas.com/offices/europe/czech/reference/list.html

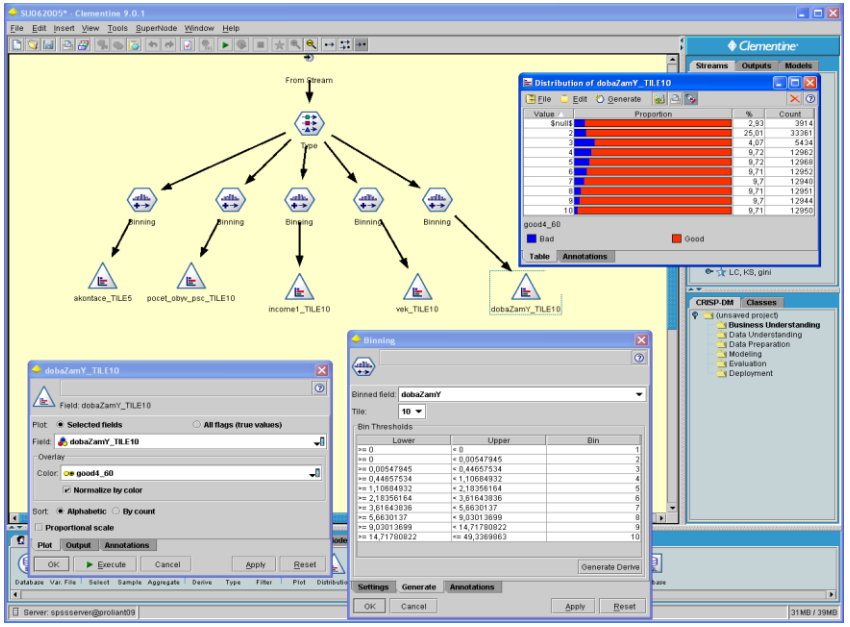
- o akademickém programu:
http://www.sas.com/offices/europe/czech/academic/index.html

- o konferenci SAS forum:
http://www.sas.com/reg/offer/cz/2010_sas_forum_2010
http://www.sas.com/reg/offer/cz/2011_sasforum

Software -SPSS

- SPSS: www.spss.cz

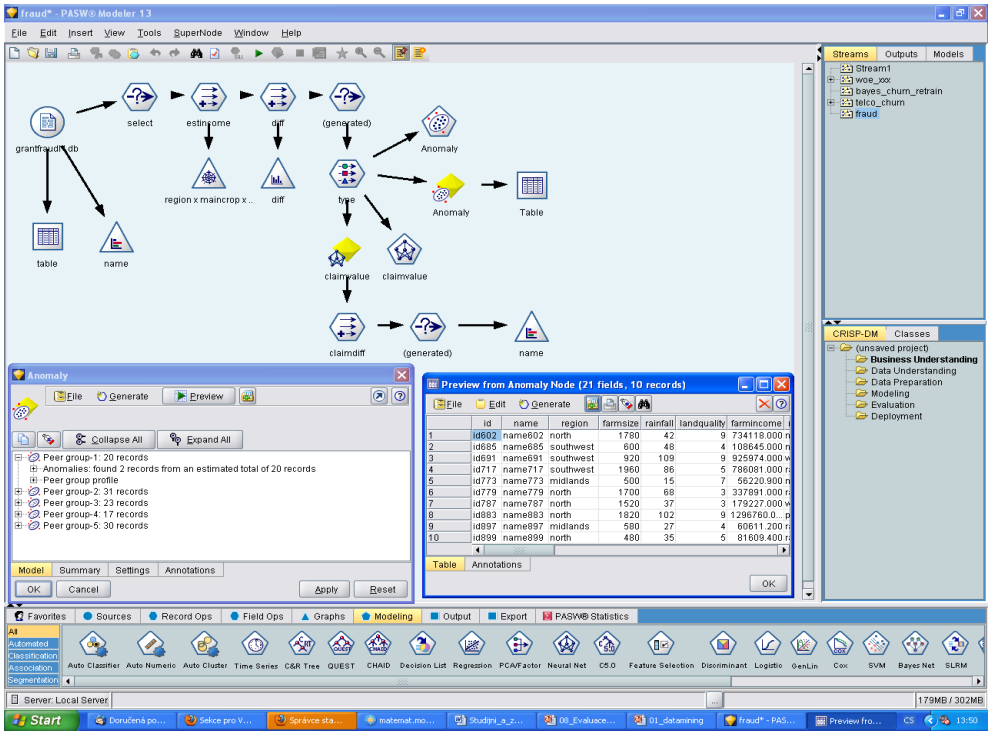


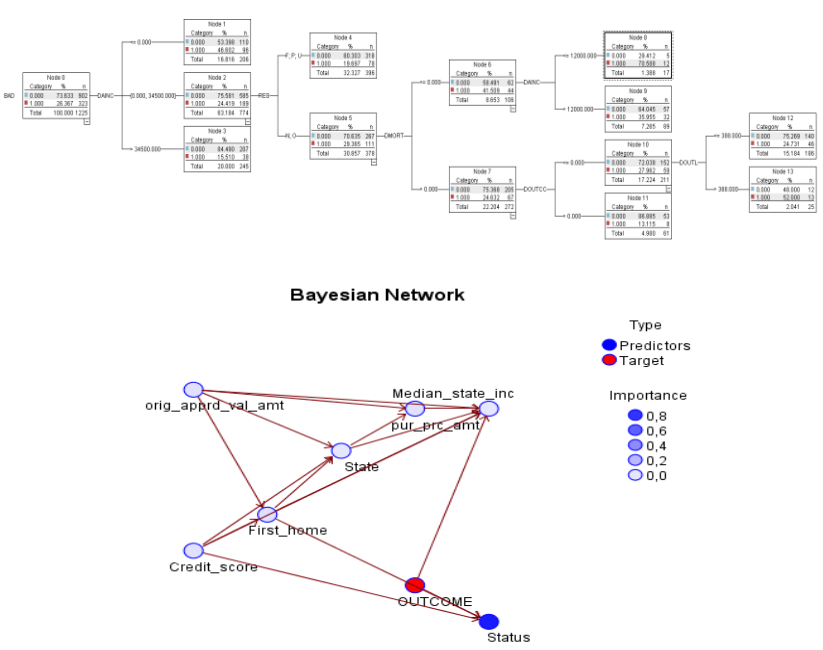


SPSS

- IBM SPSS/ PASW Modeler 13 (dříve Clementine)

  http://www.spss.cz/ibmspss_modeler.htm



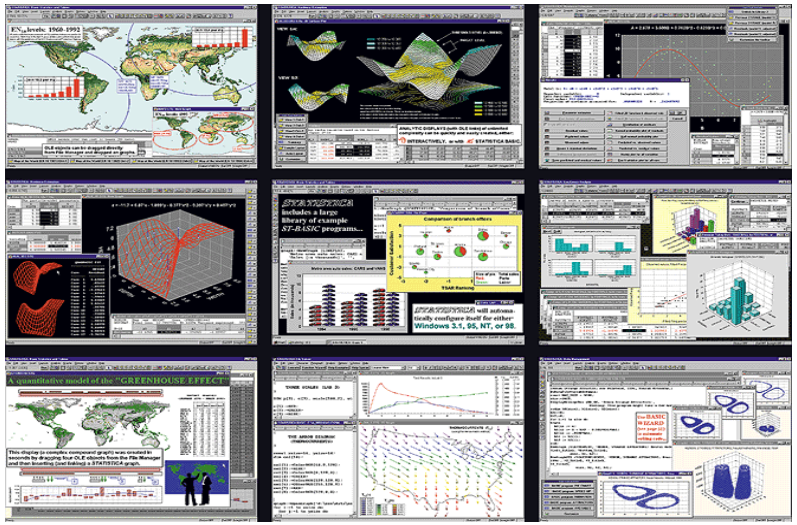


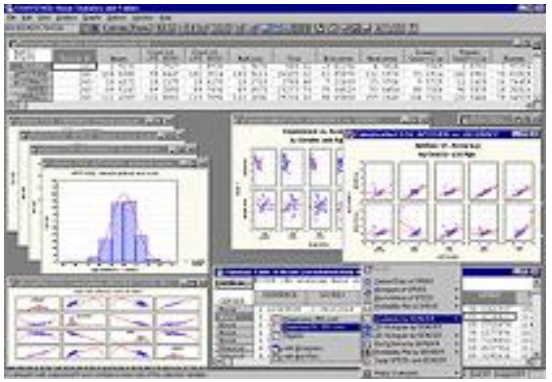
SPSS

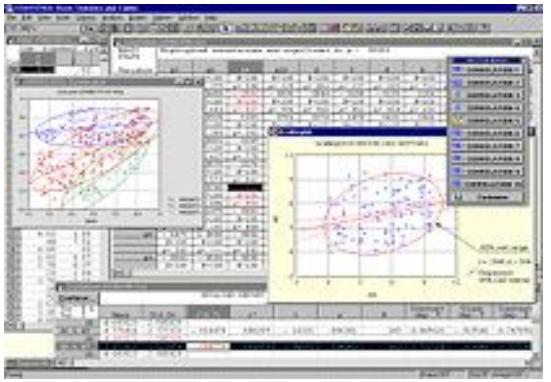
- Více o IBM SPSS Modeler 13 (dříve Clementine): http://www.spss.cz/ibmspss_modeler.htm
- (neúplný) seznam zákazníků: http://www.spss.cz/zakaznici.htm
- Akademický program: http://www.spss.com/academic/

Software -Statistica

- StatSoft www.statistica.cz

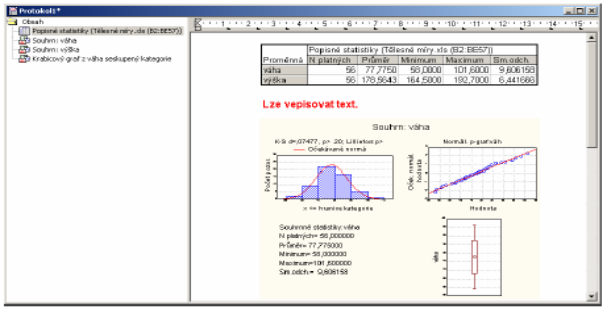






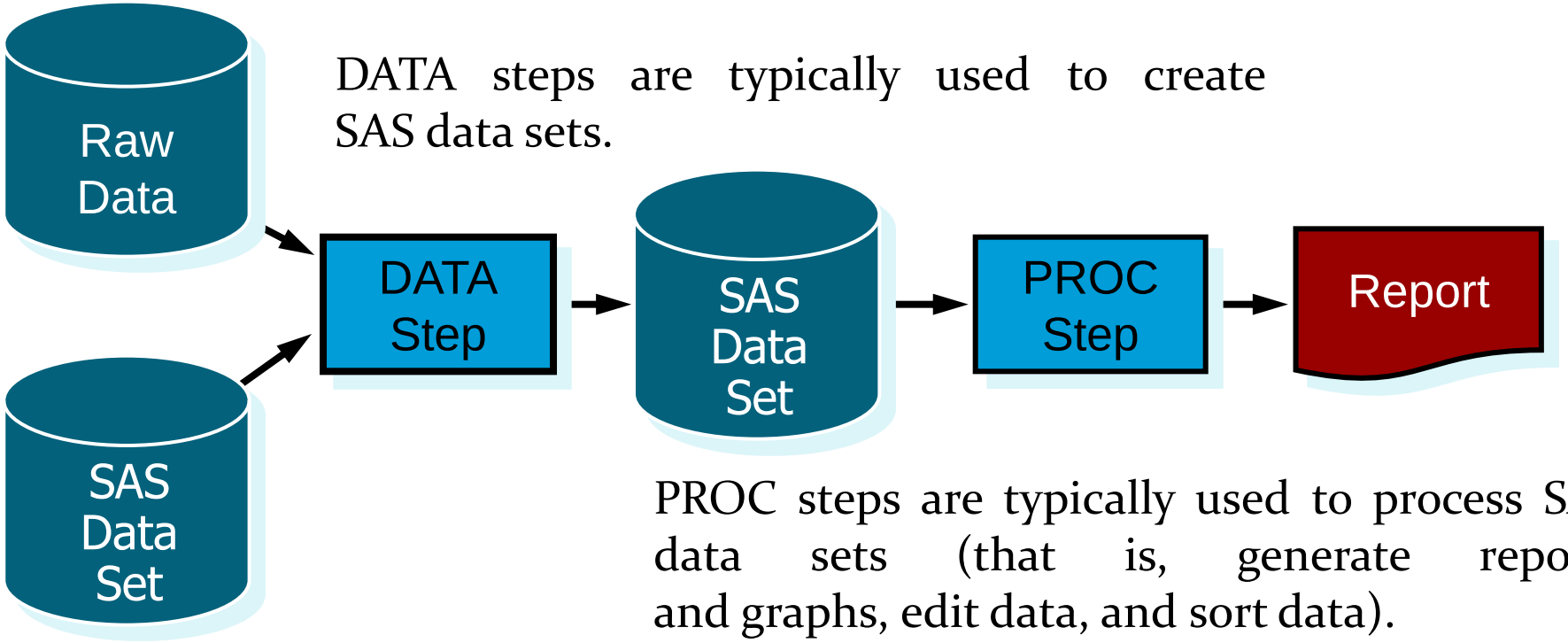
Statistica

- Více o Statistica Data Miner: http://www.statistica.cz/produkty/5-dataminingove-nastroje/21-statistica-data-miner/detail/
- (neúplný) seznam zákazníků: http://www.statsoft.com/customers/
- Akademický program: http://www.statsoft.com/academic/
- Petra Beranová – stručný manuál k ovládání programu STATISTICA: http://www.statsoft.cz/download/soubory/STATISTICA_manual.pdf



SAS Programs

- A *SAS program* is a sequence of steps that the user submits for execution.



DATA steps are typically used to create SAS data sets.

PROC steps are typically used to process SAS data sets (that is, generate reports and graphs, edit data, and sort data).

SAS Programs

```
data work.clubmembers work.nonclub;
   set orion.customer;
   if Customer_Type_ID = 3010
      then output work.nonclub;
   else output work.clubmembers;
run;

proc print data=work.nonclub;
   title "Non Club Members";
   var Country Gender Customer_Name;
run;
```

DATA Step

PROC Step

Step Boundaries

SAS steps begin with either of the following:

- DATA statement
- PROC statement

SAS detects the end of a step when it encounters one of the following:

- a RUN statement (for most steps)
- a QUIT statement (for some procedures)
- the beginning of another step (DATA statement or PROC statement)

Step Boundaries

```
data work.clubmembers work.nonclub;
   set orion.customer;
   if Customer_Type_ID = 3010
      then output work.nonclub;
   else output work.clubmembers;
run;

proc print data=work.clubmembers;

proc print data=work.nonclub;
   title "Non Club Members";
   var Country Gender Customer_Name;
run;
```

Submitting a SAS Program

- When you execute a SAS program, the results generated by SAS are divided into two major parts:

SAS log	contains information about the processing of the SAS program, including any warning and error messages.
SAS output	contains reports generated by SAS procedures and DATA steps.

- The Workspace includes tabs containing both the log and output, while the Process Flow, by default, displays icons only for the output.

SAS Log

```
18          data work.clubmembers work.nonclub;
19             set orion.customer;
20             if Customer_Type_ID = 3010
21                then output work.nonclub;
22             else output work.clubmembers;
23          run;

NOTE: There were 77 observations read from the data set ORION.CUSTOMER.
NOTE: The data set WORK.CLUBMEMBERS has 69 observations and 12 variables.
NOTE: The data set WORK.NONCLUB has 8 observations and 12 variables.
NOTE: DATA statement used (Total process time):
      real time           0.06 seconds
      cpu time            0.00 seconds


24
25          proc print data=work.nonclub noobs;
26             title "Non Club Members";
27             var Country Gender Customer_Name;
28          run;

NOTE: There were 8 observations read from the data set WORK.NONCLUB.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.09 seconds
      cpu time            0.00 seconds
```

PROC PRINT Output

sas | Enterprise Guide®

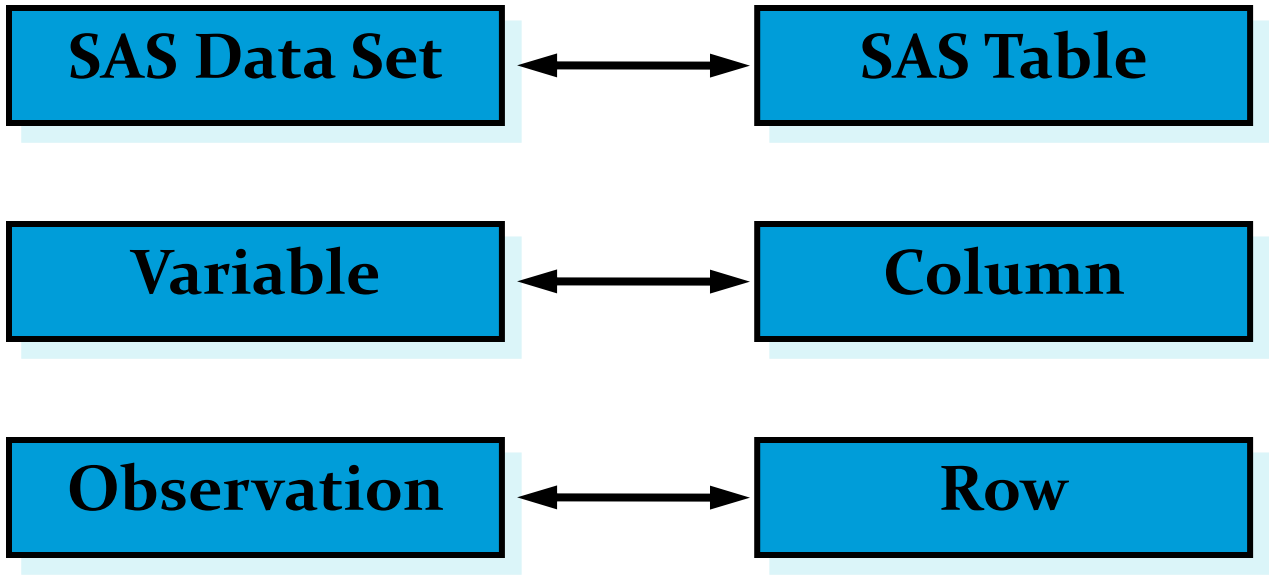
The Power to Know.

Non Club Members

Obs	Country	Gender	Customer_Name
1	DE	M	Ulrich Heyde
2	US	M	Tulio Devereaux
3	US	F	Robyn Klem
4	US	F	Cynthia Mccluney
5	AU	F	Candy Kinsey
6	US	M	Phenix Hill
7	IL	M	Avinoam Zweig
8	CA	F	Lauren Marx

SAS Terminology

- SAS documentation and text in the SAS windowing environment use the following terms interchangeably:



SAS Syntax Rules

SAS statements have these characteristics:

- usually begin with an **identifying keyword**
- always end with a **semicolon**

```
data work.clubmembers work.nonclub;
   set orion.customer;
   if Customer_Type_ID = 3010
      then output work.nonclub;
   else output work.clubmembers;
run;

proc print data=work.nonclub;
   title "Non Club Members";
   var Country Gender Customer_Name;
run;
```

SAS Syntax Rules

SAS statements are free-format.

- One or more blanks or special characters can be used to separate words.
- Statements can begin and end in any column.
- A single statement can span multiple lines.
- Several statements can be on the same line.

Unconventional Spacing

```
data work.clubmembers work.nonclub;
set orion.customer;
if Customer_Type_ID =      3010
           then output work.nonclub;
     else output work.clubmembers;run;
 proc print data=work.nonclub;          run;
```

SAS Syntax Rules

SAS statements are free-format.

- One or more blanks or special characters can be used to separate words.
- Statements can begin and end in any column.
- A single statement can span multiple lines.
- Several statements can be on the same line.

Unconventional Spacing

```
data work.clubmembers work.nonclub;
set orion.customer;
if Customer_Type_ID =      3010
            then output work.nonclub;
     else output work.clubmembers;run;
 proc print data=work.nonclub;          run;
```

SAS Syntax Rules

SAS statements are free-format.

- One or more blanks or special characters can be used to separate words.
- Statements can begin and end in any column.
- A single statement can span multiple lines.
- Several statements can be on the same line.

Unconventional Spacing

```
data work.clubmembers work.nonclub;
set orion.customer;
if Customer_Type_ID =      3010
            then output work.nonclub;
      else output work.clubmembers;run;
 proc print data=work.nonclub;          run;
```

SAS Syntax Rules

SAS statements are free-format.

- One or more blanks or special characters can be used to separate words.
- Statements can begin and end in any column.
- A single statement can span multiple lines.
- Several statements can be on the same line.

Unconventional Spacing

```
data work.clubmembers work.nonclub;
set orion.customer;
if Customer_Type_ID =      3010
            then output work.nonclub;
     else output work.clubmembers;run;
 proc print data=work.nonclub;          run;
```

SAS Syntax Rules

SAS statements are free-format.

- One or more blanks or special characters can be used to separate words.
- Statements can begin and end in any column.
- A single statement can span multiple lines.
- Several statements can be on the same line.

Unconventional Spacing

```
data work.clubmembers work.nonclub;
set orion.customer;
if Customer_Type_ID =      3010
            then output work.nonclub;
      else output work.clubmembers;run;
 proc print data=work.nonclub;              run;
```

SAS Comments

SAS comments consist of text that SAS ignores during processing. You can use comments anywhere in a SAS program to

- document the purpose of the program
- explain segments of the program
- mark SAS code as non-executing text.

Two methods of commenting are shown below:

```
/* comment */
```

```
* comment ;
```

SAS Comments: Examples

```
/* Split data based on membership */
data work.clubmembers work.nonclub;
   set orion.customer;
   if Customer_Type_ID = 3010
      then output work.nonclub;
   else output work.clubmembers;
run;
```

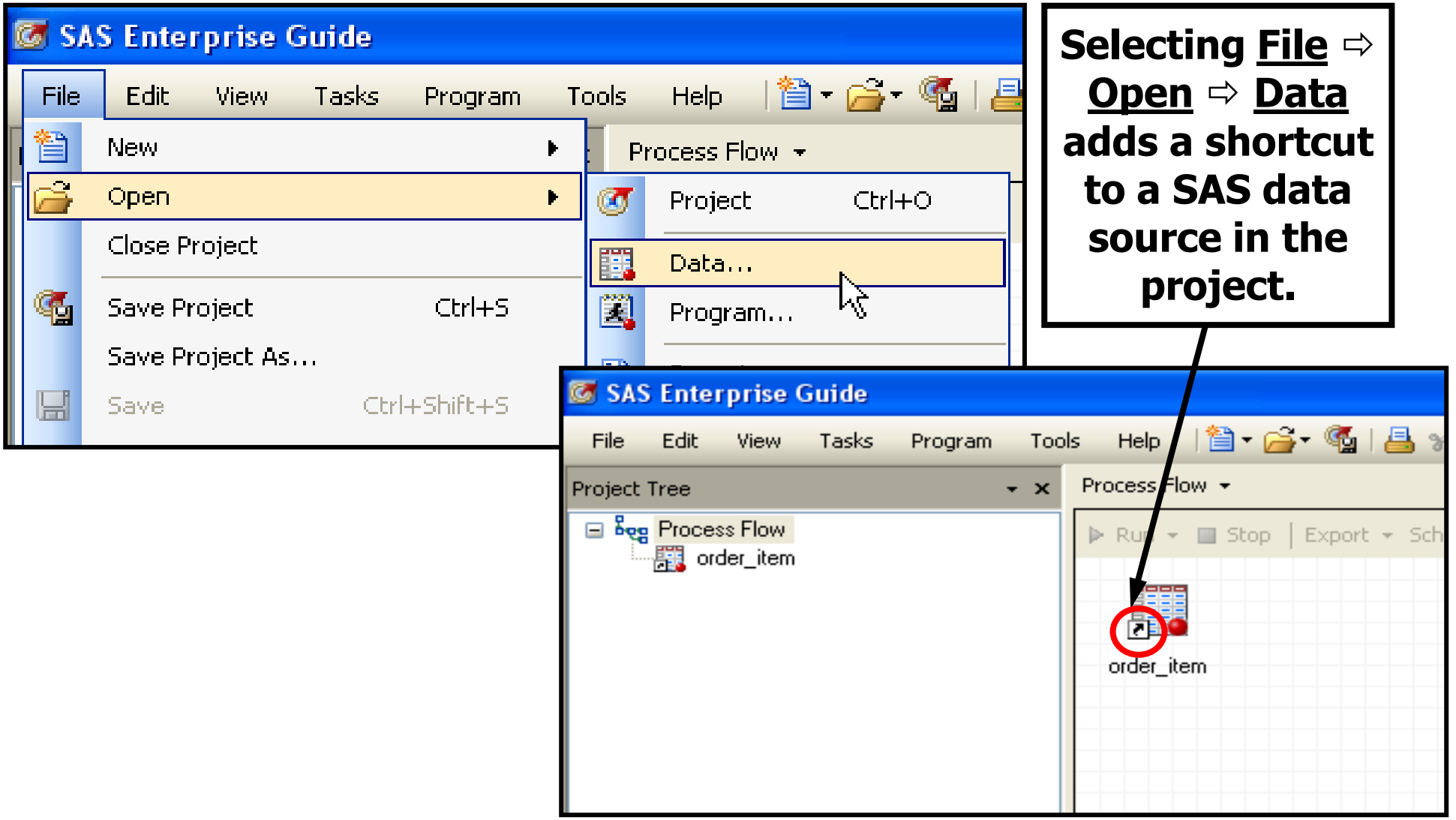
```
proc print data=work.nonclub;
   title "Non Club Members";
   *var Country Gender Customer_Name;
run;
```

Syntax Errors

- Syntax errors occur when program statements do not conform to the rules of the SAS language.
- Examples of syntax errors:
  - misspelled keywords
  - unmatched quotation marks
  - missing semicolons
  - invalid options
- When SAS encounters a syntax error, SAS prints a warning or an error message to the log.

```
ERROR 22-322: Syntax error, expecting one of the following:
              a name, a quoted string, (, /, ;, _DATA_, _LAST_,
              _NULL_.
```

How Do You Include Data in a Project?



How Do You Include Data in a Program?

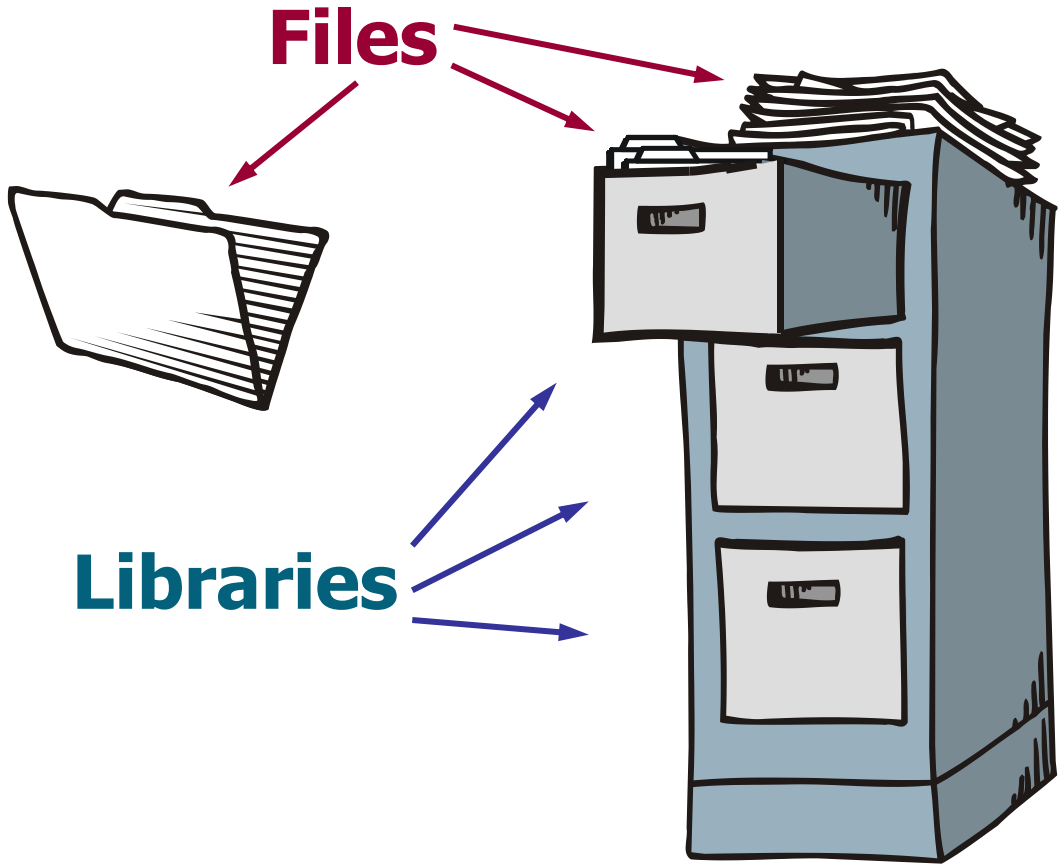
- One possibility is to include the full path and filename each time that a SAS data set is referenced.

```
data "s:\workshop\cust_age.sas7bdat";
   set "s:\workshop\customer.sas7bdat";
   /*Calculate each customer's age*/
   Age=int(yrdif(Birth_Date,today(),"actual"));
run;

proc print data="s:\workshop\cust_age.sas7bdat";
   var Customer_Name Gender Country Age;
   title "Customer Listing";
run;
```

ep02d03.sas

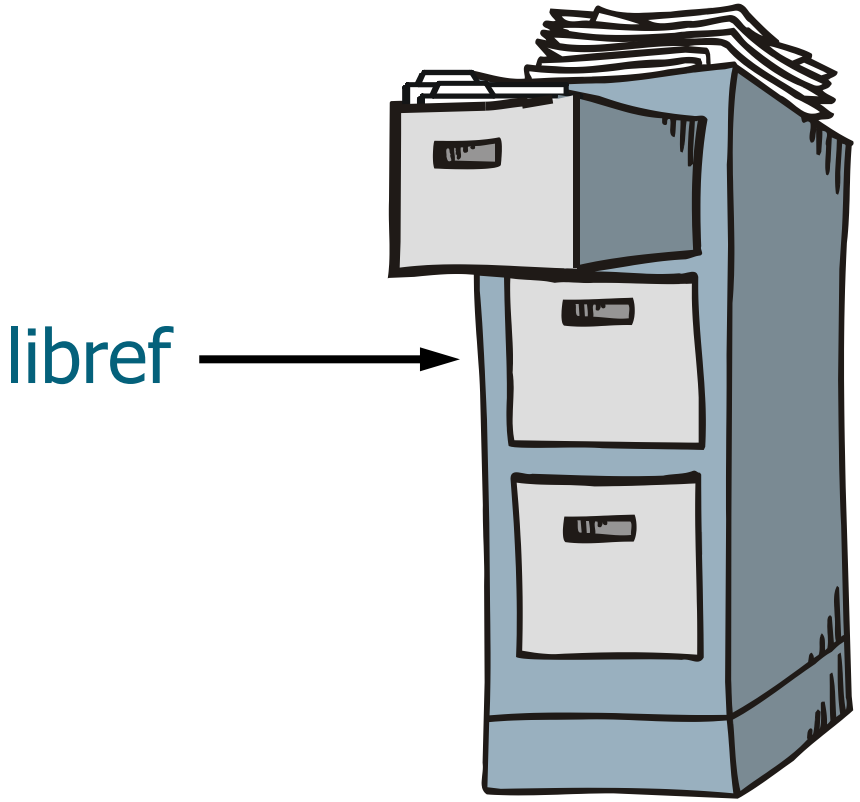
SAS Libraries

You can think of a SAS library as a drawer in a filing cabinet and a SAS data set as one of the file folders in the drawer.



Assigning a Libref

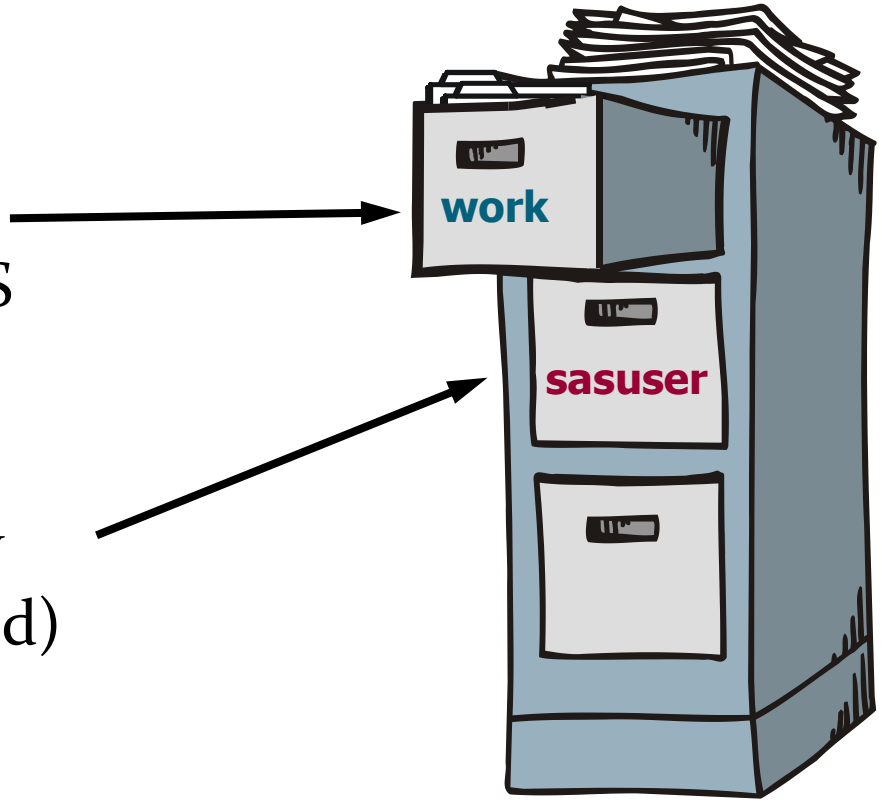
- Regardless of which host operating system you use, you identify SAS libraries by assigning a *library reference name (libref)* to each library.
- This libref can serve as a shortcut in SAS programs in place of the full path or filename.



SAS Libraries

When a SAS session starts, SAS automatically creates one temporary and at least one permanent SAS library that you can access.

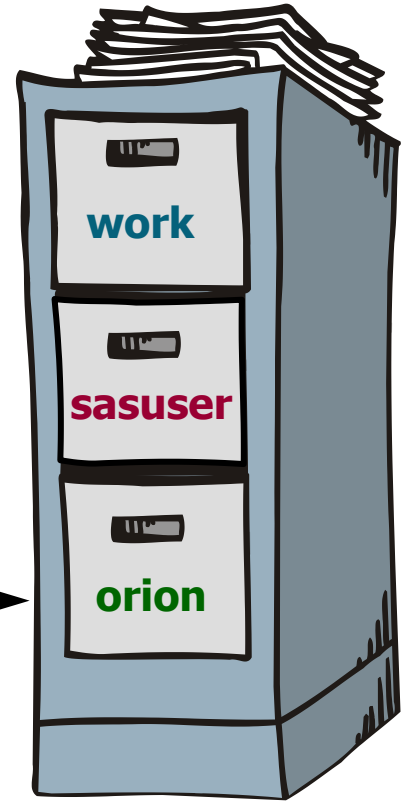
- **`work`** - temporary library (contents are deleted when SAS closes)
- **`sasuser`** - permanent library (contents are permanently saved)



SAS Libraries

- You can also create and access your own permanent libraries.

`orion` – permanent library



Assigning a Libref

- You can use the LIBNAME statement to assign a libref to a SAS library. The LIBNAME statement is a global statement.
- General form of the LIBNAME statement:

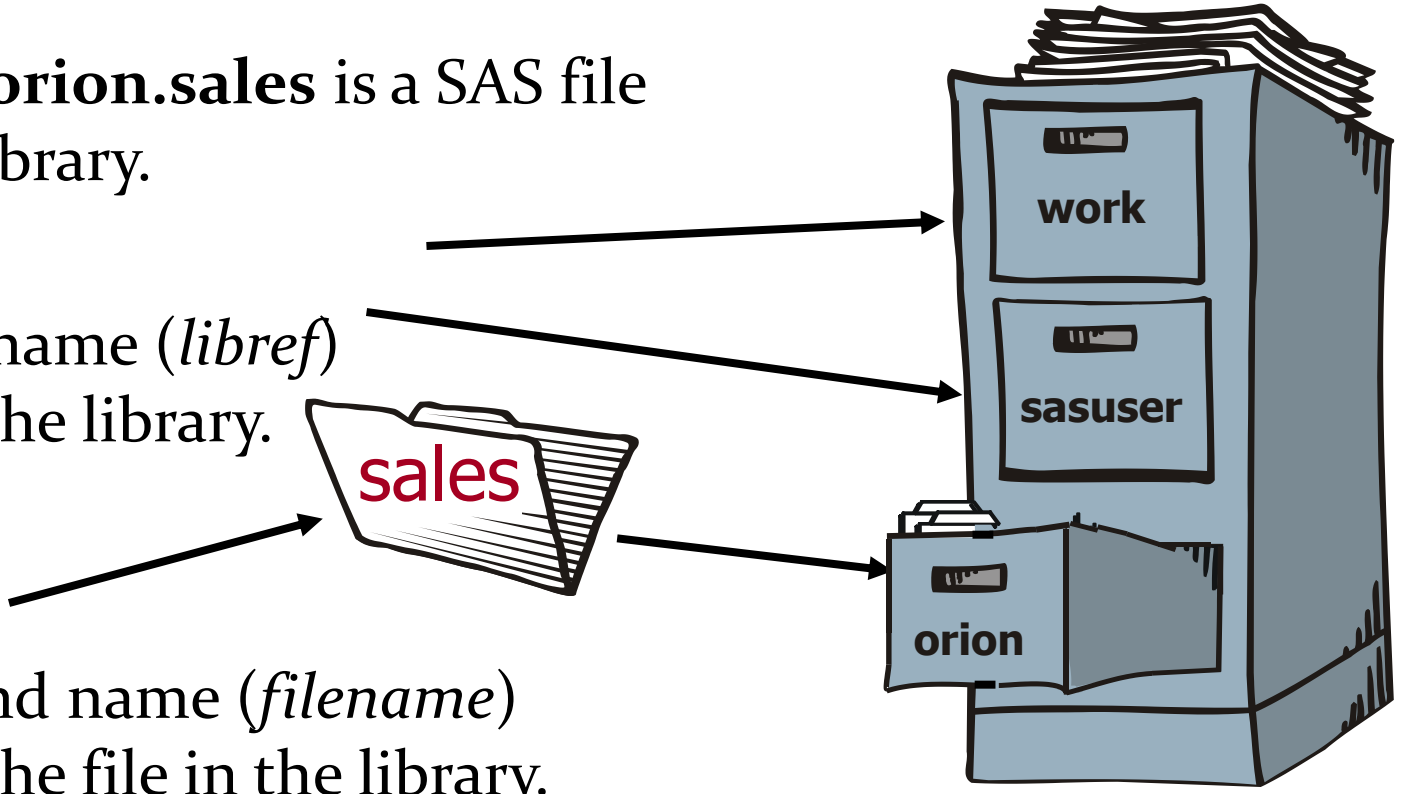
LIBNAME *libref* '*SAS-data-library*' <*options*>;

- The rules for naming a libref are as follows:
  - must be 8 or fewer characters
  - must begin with a letter or underscore
  - remaining characters are letters, numbers, or underscores

Two-Level SAS Filenames

- Every SAS file has a two-level name: *libref.filename*

- The data set **orion.sales** is a SAS file in the **orion** library.



  - The first name (*libref*) refers to the library.
  - The second name (*filename*) refers to the file in the library.

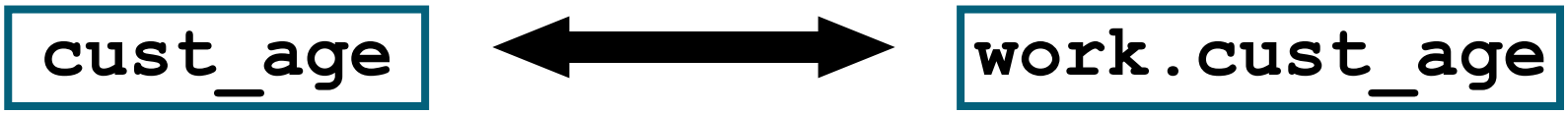
How Do You Include Data in a Program?

- využijeme knihovny (libraries)

```
libname orion "s:\workshop";
data work.cust_age;
   set orion.customer;
   /*Calculate each customer's age*/
   Age=int(yrdif(Birth_Date,today(),"actual"));
run;

proc print data=work.cust_age;
   var Customer_Name Gender Country Age;
   title "Customer Listing";
run;
```

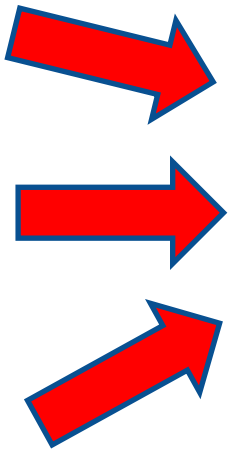
Temporary SAS Filename

- The default libref is **work** if the libref is omitted.




```
libname orion "s:\workshop";
data work.cust_age;
   set orion.customer;
   /*Calculate each customer's age*/
   Age=int(yrdif(Birth_Date,today(),"actual"));
run;

proc print data=cust_age;
   var Customer_Name Gender Country Age;
   title "Customer Listing";
run;
```

Import dat



***.sas7bdat**

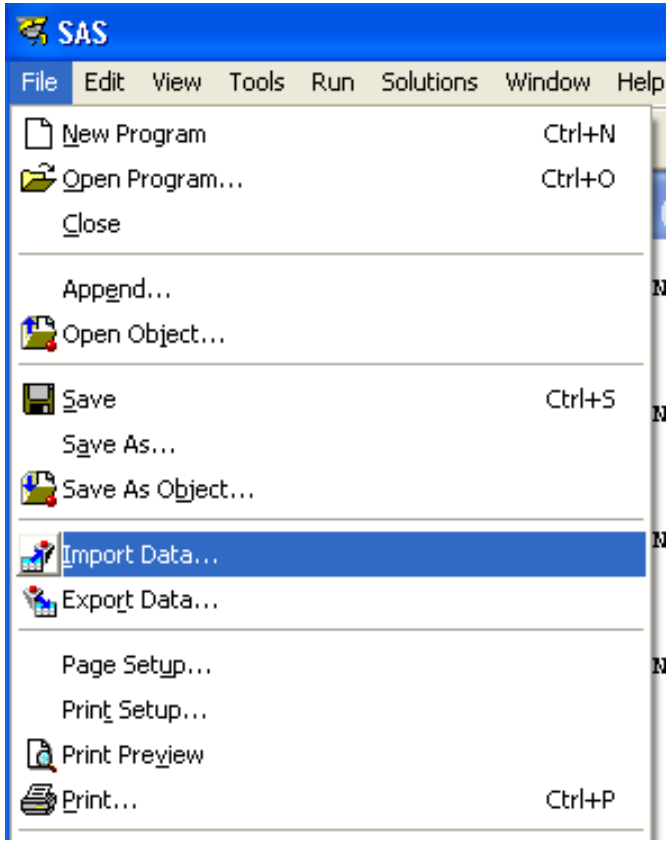
Základních pět možností importu dat:

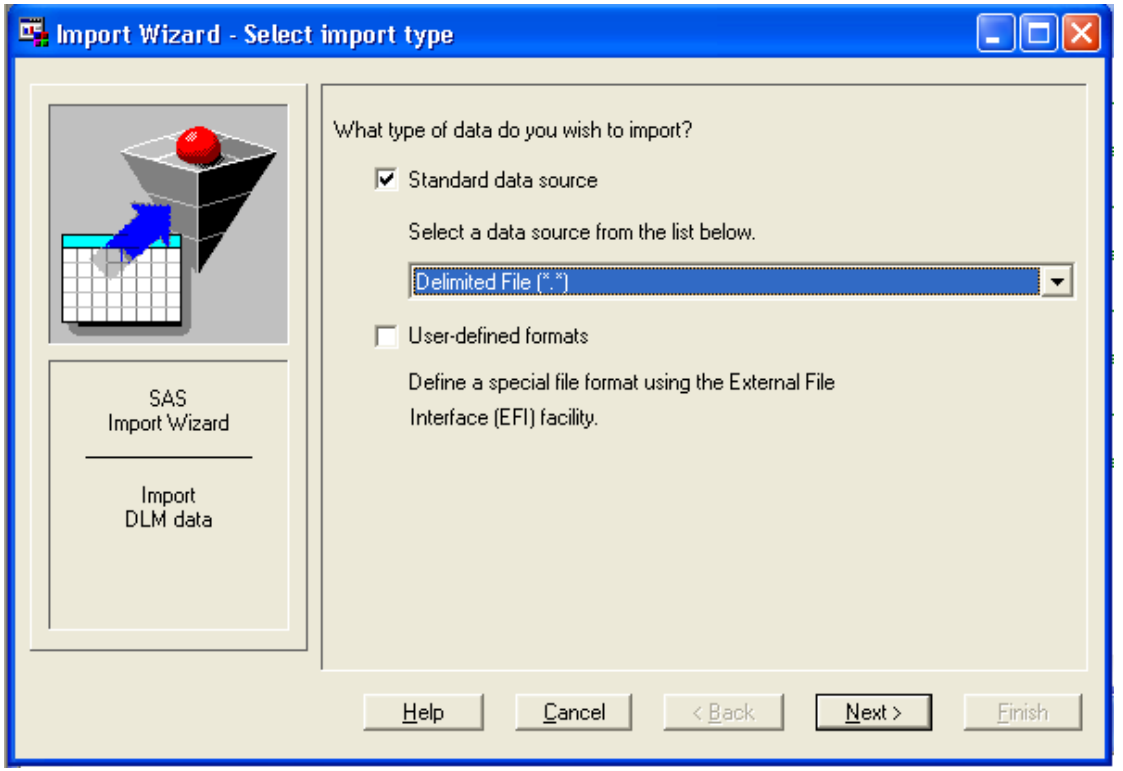
1. Import v SAS EG
2. Import wizard
3. Proc import
4. Data step
5. Proc SQL

Import Wizard

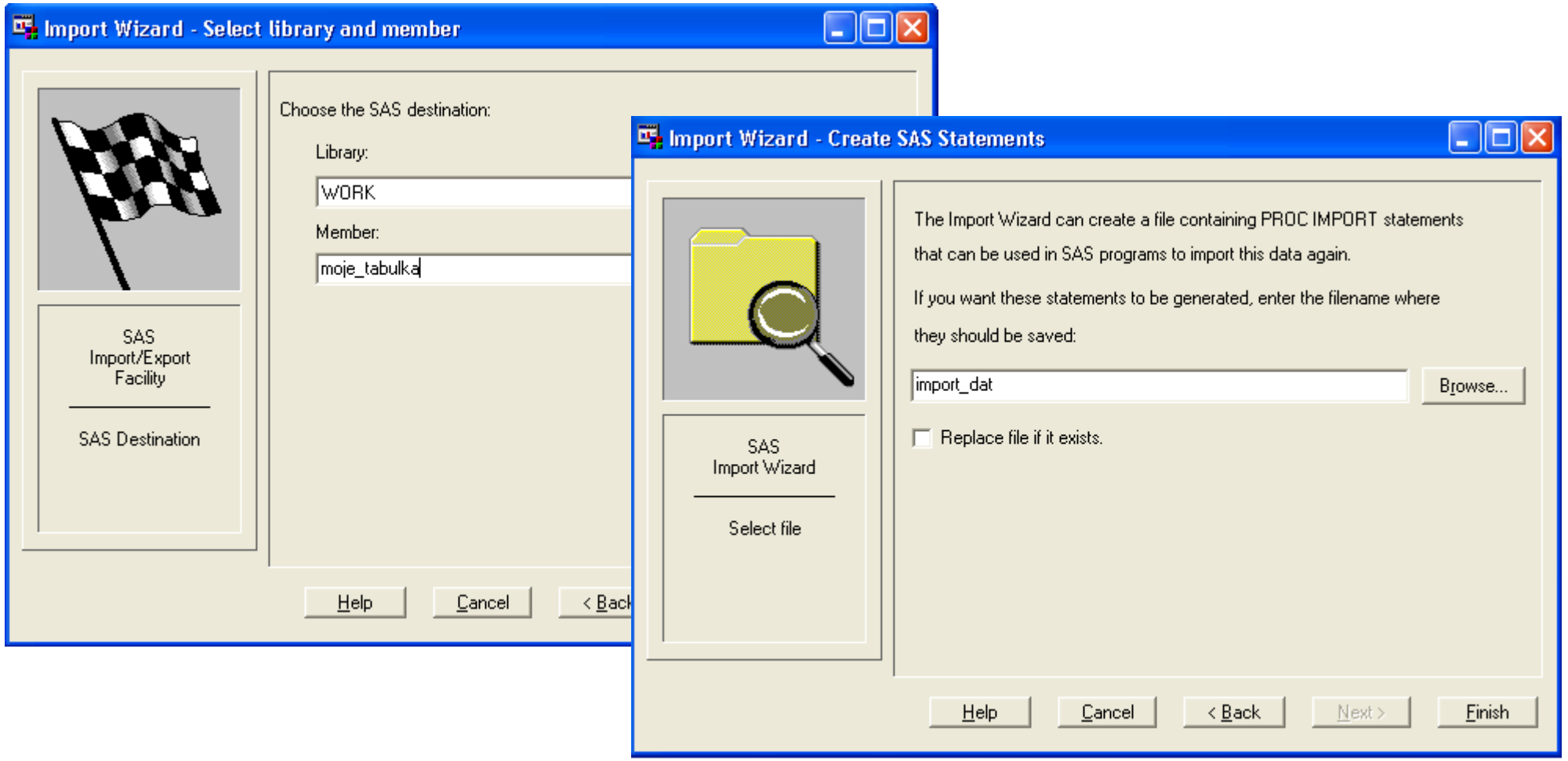
- The *Import Wizard* is a point-and-click graphical interface that enables you to create a SAS data set from several types of external files including the following:
  - dBASE files (*.DBF)
  - Excel spreadsheets (*.XLS)
  - Microsoft Access tables (.MDB)
  - delimited files (*.*)
  - comma-separated values (*.CSV)
  - …

Import Wizard





Import Wizard



PROC IMPORT

```
PROC IMPORT OUT= WORK.sales
            DATAFILE= "S:\Workshop\sales.xls"
            DBMS=EXCEL REPLACE;
     RANGE="Australia$";
     GETNAMES=YES;
     MIXED=NO;
     SCANTEXT=YES;
     USEDATE=YES;
     SCANTIME=YES;
RUN;
```

PROC IMPORT

GETNAMES=YES | NO

- determines whether SAS will use the first row of data in a Microsoft Excel worksheet or range as column names.

  YES specifies to use the first row of data in an Excel worksheet or range as column names.

  NO specifies **not** to use the first row of data in an Excel worksheet or range as column names. SAS generates and uses the variable names F1, F2, F3, and so on.

- The default is YES.

PROC IMPORT

MIXED=YES | NO

- specifies whether to import data with both character and numeric values and convert all data to character.

  YES specifies that all data values will be converted to character.

  NO specifies that numeric data will be missing when a character type is assigned. Character data will be missing when a numeric data type is assigned.

- The default is NO.

PROC IMPORT

SCANTEXT=YES | NO

specifies whether to read the entire data column and use the length of the longest string found as the SAS column width.

YES	scans the entire data column and uses the longest string value to determine the SAS column width.
NO	does not scan the column and defaults to a width of 255.

- The default is YES.

PROC IMPORT

SCANTIME=YES | NO

specifies whether to scan all row values in a date/time column and automatically determine the TIME data type if **only** time values exist.

YES specifies that a column with only time values be assigned the TIME8. format.

NO specifies that a column with only time values be assigned the DATE9. format.

- The default is NO.

PROC IMPORT

USEDATE=YES | NO

- specifies whether to use the DATE9. format for date/time values in Excel workbooks.

  YES specifies that date/time values be assigned the DATE9. format.

  NO specifies that date/time values be assigned the DATETIME16. format.

- The default is YES.

Proc import vs. Data step

```
PROC IMPORT OUT= WORK.MDATA1
            DATAFILE=
"G:\dukumenty\diplomka-data.txt"
            DBMS=CSV REPLACE;
     GETNAMES=YES;
     DATAROW=2;
RUN;
```

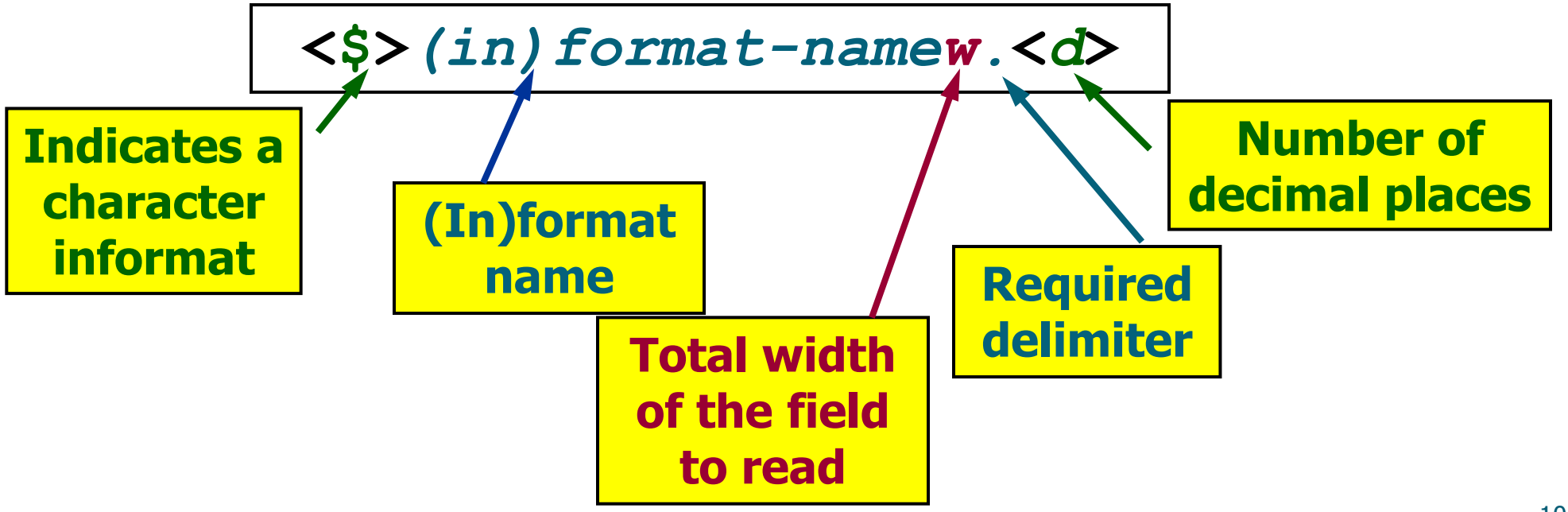
```
data work.mdata2;
length
BIRTHPLACE $ 25
AGE $ 25
.
.
.
EDUCATION $ 25
;
infile  'G:\dukumenty\diplomka-data.csv' delimiter = ';'
DSD lrecl=3276 firstobs=2 ;
input
BIRTHPLACE
AGE
.
.
.
EDUCATION
;
run;
```

Import z SQL databáze

```
libname my_data 'C:\Scoring\SASdata\';

proc sql;
connect to odbc as mssql (complete="DRIVER=SQL Server;
SERVER=sqlserv;Trusted_connection=Yes ");
create view my_data.wset_of_segments as select * from connection to mssql
(select * from db1.rezac.segmenty);
disconnect from mssql;
quit;

proc sql;
create table my_data.set_segments as
select
*
from my_data.wset_of_segments
;
quit;
```

Formats (Informats)

- An *informat* is an instruction that SAS uses to **read** data values.
- A *format* is an instruction that SAS uses to **write** data values.
- SAS (in)formats have the following form:



Formats (Informats)

InFormats by Category:

Category	***Description***
Character	instructs SAS to read character data values into character variables.
Column Binary	instructs SAS to read data stored in column-binary or multipunched form into character and numeric variables.
Date and Time	instructs SAS to read date values into variables that represent dates, times, and datetimes.
ISO 8601	instructs SAS to read date, time, and datetime values that are written in the ISO 8601 standard into either numeric or character variables.
Numeric	instructs SAS to read numeric data values into numeric variables.

http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a001239776.htm

Formats (Informats)

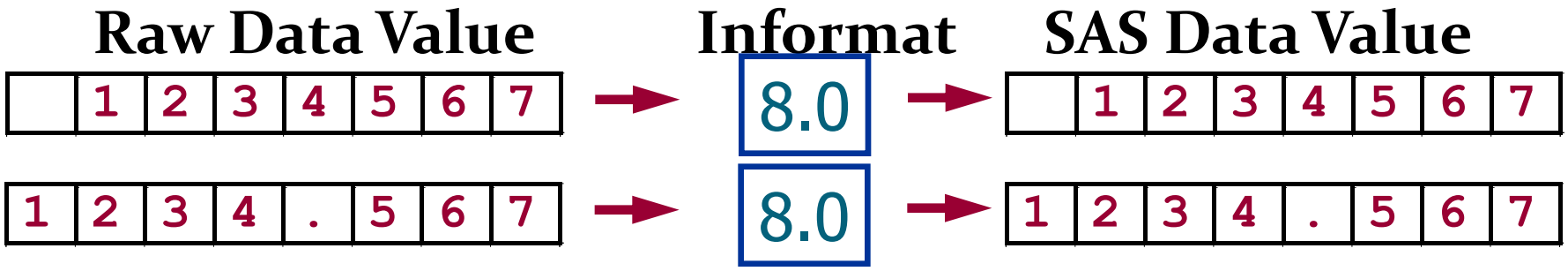
Formats by Category:

Category	***Description***
Character	instructs SAS to write character data values from character variables.
Date and Time	instructs SAS to write data values from variables that represent dates, times, and datetimes.
ISO 8601	instructs SAS to write date, time, and datetime values using the ISO 8601 standard.
Numeric	instructs SAS to write numeric data values from numeric variables.

http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a001263753.htm

Selected Informats

8. or 8.0 reads eight columns of numeric data.

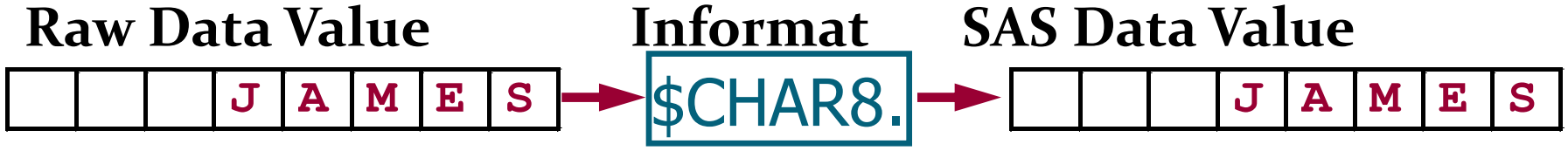


8.2 reads eight columns of numeric data and **may** insert a decimal point in the value.

Selected Informats

$8.	reads eight columns of character data and removes leading blanks.

$CHAR8.	reads eight columns of character data and preserves leading blanks.



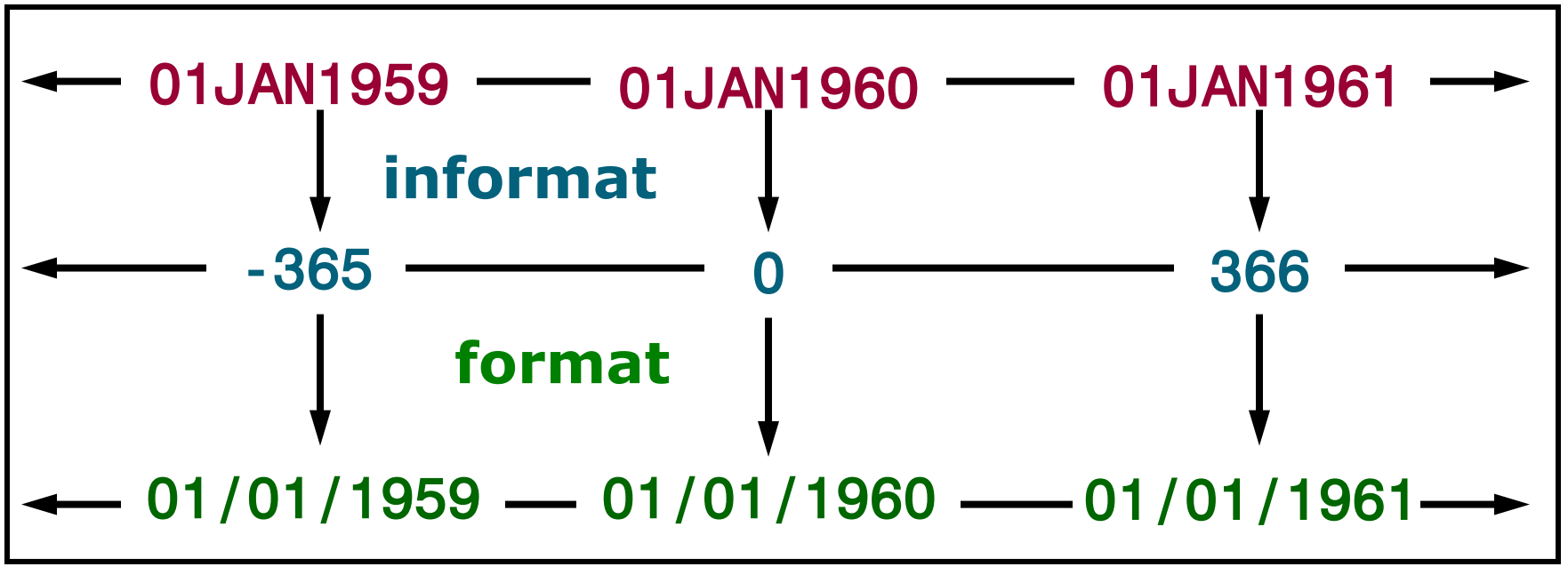
Selected Informats

COMMA7.	reads seven columns of numeric data and removes selected nonnumeric characters such as dollar signs and commas.

MMDDYY8.	reads dates of the form 10/29/01.

Datumové formáty

- Date values that are stored as SAS dates are special numeric values.
- A *SAS date value* is interpreted as the number of days between January 1, 1960, and a specific date.



Datumové formáty

- SAS uses date **informats** to **read** and **convert** dates to SAS date values.

Examples:

Raw Data Value	Informat	Converted Value
10/29/2001	MMDDYY10.	15277
10/29/01	MMDDYY8.	15277
29OCT2001	DATE9.	15277
29/10/2001	DDMMYY10.	15277

Number of days between 01JAN1960 and 29OCT2001

Optimalizace práce s daty v SAS

- Pro (velmi) velké datové soubory je vhodné použití **komprese** a **indexování** SASovských tabulek. Více na:

http://www2.sas.com/proceedings/sugi27/p023-27.pdf

http://www2.sas.com/proceedings/sugi28/003-28.pdf

http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a001288760.htm

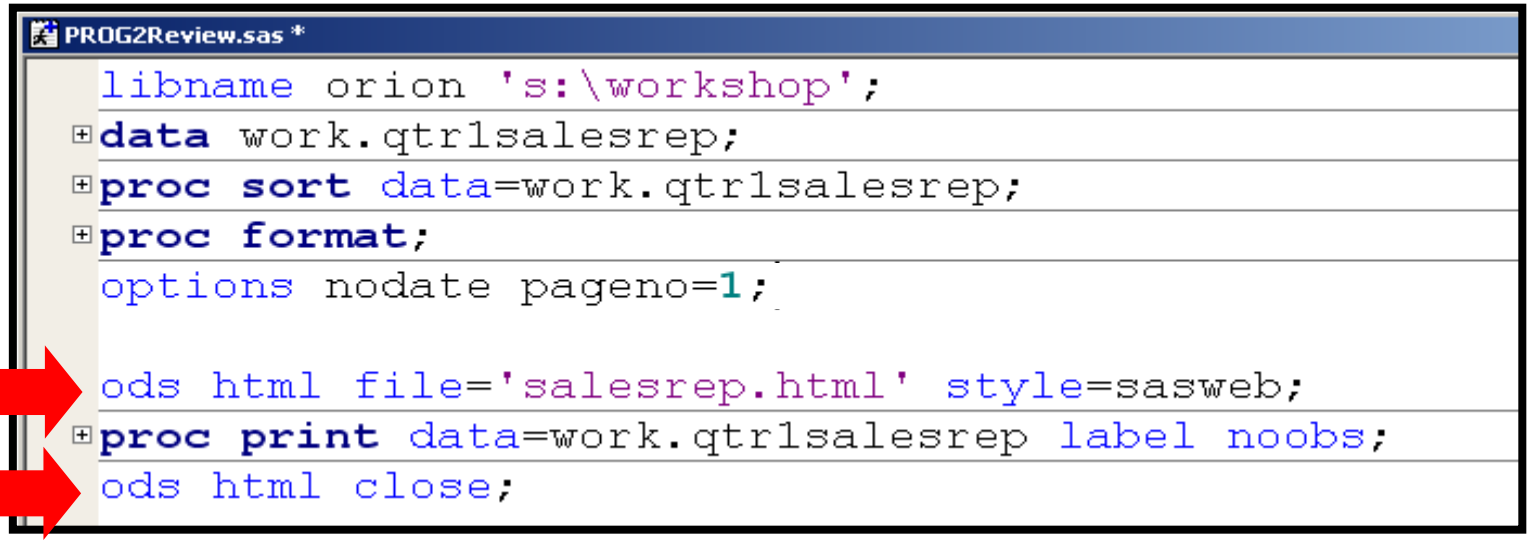
http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000131138.htm

Příklad:

```
data lib1.tab2 (compress=binary index=(var1 var2));
set lib1.tab1;
...
run;
```

ODS – The Output Delivery System

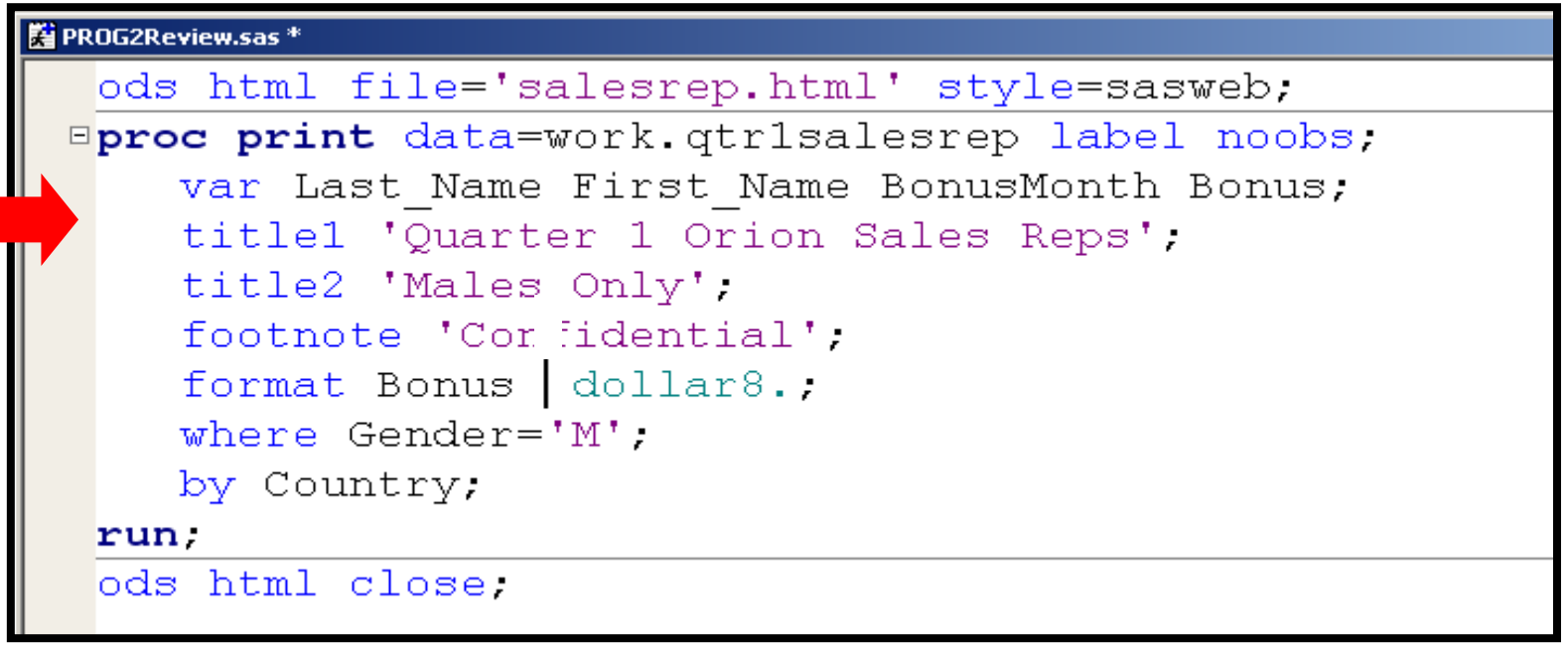
- The Output Delivery System (ODS) enables you to produce output in a variety of formats, including HTML, RTF, PDF, and the default SAS listing.



- The ODS statements above create an HTML file, salesrep.html, using the output produced by the PROC PRINT step.

The PRINT Procedure

- The PRINT procedure prints the observations in a SAS data set and uses all or some of the variables.



- The PRINT procedure above includes TITLE and FOOTNOTE statements, which are global statements and do not need to be enclosed in a DATA or PROC step.

Program Output

Partial PROC PRINT Output
(SAS Output window)

```
                    Quarter 1 Orion Sales Reps
                              Males Only

--------------------- Country=AU -----------------
                        ------

                             First        Month of
    Last Name         Name          Bonus           Bonus

    Wills             Matsuoka          1            $300
    Surawski          Marinus           1            $300
    Shannan           Sian              1            $300
    Scordia           Randal            2            $300
    Pretorius         Tadashi           3            $300
    Nowd              Fadi              1            $300
    Magrath           Brett             1            $300
```

Partial PROC PRINT Output
(HTML format)

Quarter 1 Orion Sales Reps
Males Only

Country=AU

Last Name	First Name	Month of Bonus	Bonus
Wills	Matsuoka	1	$300
Surawski	Marinus	1	$300
Shannan	Sian	1	$300
Scordia	Randal	2	$300
Pretorius	Tadashi	3	$300
Nowd	Fadi	1	$300
Magrath	Brett	1	$300

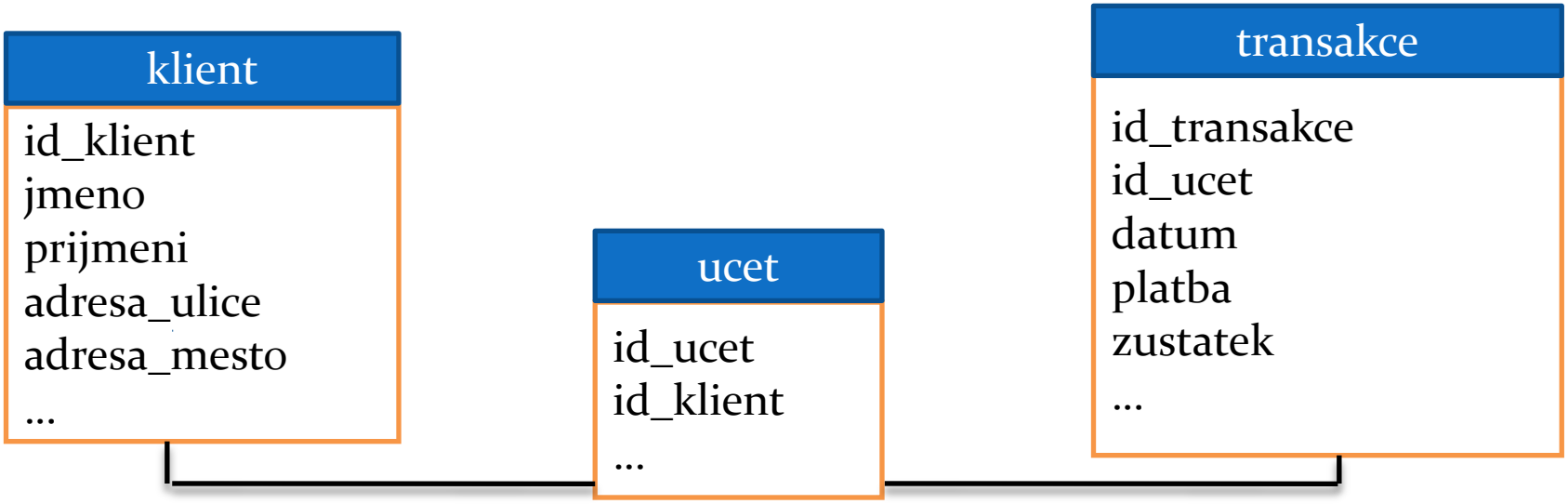
3. Organizace dat, úvod do SQL

Historie skladování dat

- V minulosti byla data ukládána v jednom velkém souboru, ke kterému se přistupovalo indexovanými sekvenčními metodami. Soubor byl indexován na základě předpokládaných způsobů dotazování. Velkou nevýhodou bylo to, že se informace v záznamech opakovaly a typy dotazů byly předurčeny.

datum	jmeno	prijmeni	adresa_ulice	adresa_mesto	cislo_uctu	platba	zustatek
980103	Jan	Novak	Dlouha 5	Praha 1	9945371	100,00	100,00
980105	Jan	Novak	Dlouha 5	Praha 1	9945371	1500,00	1600,00
980106	Jan	Novak	Dlouha 5	Praha 1	9945371	-1500,00	50,00
980106	Karel	Nemec	Lucni 4	Praha 2	24867134	3000,00	6000,00
980107	Karel	Nemec	Lucni 4	Praha 2	24867134	-4000,00	2000,00
980108	Jan	Novak	Dlouha 5	Praha 1	9945371	-150,00	-100,00
980111	Karel	Nemec	Lucni 4	Praha 2	24867134	5000,00	7000,00

Relační databáze




```
SELECT klient.jmeno, klient.prijmeni, klient.adresa_ulice,
klient.adresa_mesto, ucet.cislo_uctu, transakce.zustatek
FROM klient, ucet, transakce
WHERE klient.id_klient = ucet.id_klient;
AND transakce.id_ucet = ucet.id_ucet;
AND transakce.zustatek < 100;
GROUP BY klient.adresa_mesto;
```

Relační databáze

- **Relační databáze** je databáze založená na relačním modelu. Často se tímto pojmem označuje nejen databáze samotná, ale i její konkrétní softwarové řešení.
- Relační databáze je založena na tabulkách, jejichž řádky obvykle chápeme jako záznamy a eventuelně některé sloupce v nich (tzv. cizí klíče) chápeme tak, že uchovávají informace o relacích mezi jednotlivými záznamy v matematickém slova smyslu.
- Termín *relační databáze* definoval Edgar F. Codd v roce 1970.

- způsoby kladení dotazů:
  - QBE (query by example)
  - SQL (structured query language)

- Dle relační teorie lze pomocí základních operací (sjednocení, kartézský součin, rozdíl, selekce, projekce a spojení) uskutečnit veškeré operace s daty a ostatní operace jsou již jen kombinacemi těchto pěti.

Relační databáze

- Základem relačních databází jsou databázové tabulky. Jejich sloupce se nazývají atributy nebo pole, řádky tabulky jsou pak záznamy. Atributy mají určen svůj konkrétní datový typ - doménu. Řádek je řezem přes sloupce tabulky a slouží k vlastnímu uložení dat. Konkrétní tabulka pak realizuje podmnožinu kartézského součinu možných dat všech sloupců - relaci.
- **Primární klíč**
  - Primární klíč je jednoznačný identifikátor záznamu, řádku tabulky. Primárním klíčem může být jediný sloupec či kombinace více sloupců tak, aby byla zaručena jeho jednoznačnost. Pole klíče musí obsahovat hodnotu, tzn. nesmí se zde vyskytovat nedefinovaná prázdná hodnota NULL. V praxi se dnes často používají umělé klíče, což jsou číselné či písmenné identifikátory - každý nový záznam dostává identifikátor odlišný od identifikátorů všech předchozích záznamů (požadavek na unikátnost klíče), obvykle se jedná o celočíselné řady a každý nový záznam dostává číslo vždy o jednotku vyšší (zpravidla zcela automatizovaně) než je číslo u posledního vloženého záznamu (číselné označení záznamů s časem stoupá).
- **Cizí klíč**
  - Dalším důležitým pojmem jsou nevlastní/cizí klíče. Slouží pro vyjádření vztahů, relací, mezi databázovými tabulkami. Jedná se o pole či skupinu polí, která nám umožní identifikovat, které záznamy z různých tabulek spolu navzájem souvisí.

Relační databáze – vztahy mezi tabulkami

- Vztahy, neboli relace, slouží ke svázání dat, která spolu souvisejí a jsou umístěny v různých databázových tabulkách. V zásadě rozlišujeme čtyři typy vztahů.

- mezi daty v tabulkách není žádná spojitost, proto nedefinujeme žádný vztah.

- 1:1 používáme, pokud záznamu odpovídá právě jeden záznam v jiné databázové tabulce a naopak. Takovýto vztah je používán pouze ojediněle, protože většinou není pádný důvod, proč takovéto záznamy neumístit do jedné databázové tabulky. Jedno z mála využití je zpřehlednění rozsáhlých tabulek. Jako ilustraci je možné použít vztah řidič - automobil. V jednu chvíli (diskrétní časový okamžik) řídí jedno auto právě jeden řidič a zároveň jedno auto je řízeno právě jedním řidičem.

Relační databáze – vztahy mezi tabulkami

- 1:N přiřazuje jednomu záznamu více záznamů z jiné tabulky. Jedná se o nejpoužívanější typ relace, jelikož odpovídá mnoha situacím v reálném životě. Jako reálný příklad může posloužit vztah autobus - cestující. V jednu chvíli cestující jede právě jedním autobusem a v jednom autobuse může zároveň cestovat více cestujících.
- M:N je méně častým. Umožňuje několika záznamům z jedné tabulky přiřadit několik záznamů z tabulky druhé. V databázové praxi bývá tento vztah z praktických důvodů nejčastěji realizován kombinací dvou vztahů 1:N a 1:M, které ukazují do pomocné tabulky složené z kombinace obou použitých klíčů (třetí resp. tzv. vazební tabulka). Příkladem z reálného života by mohl být vztah výrobek - vlastnost. Výrobek může mít více vlastností a jednu vlastnost může mít více výrobků. V reálném životě nicméně existuje velké množství vztahů M : N, mimo jiné také proto, že často existuje praktická potřeba zachovávat i údaje o historii těchto vztahů z časového hlediska (jeden řidič v delším časovém období řídí více rozličných aut a jedno auto v delším časovém období může mít více různých řidičů).

Slovník pojmů

ODS	Operational Data Store
DWH	DataWareHouse
DataMart	
Meta Data	
BI	Business Intelligence
OLAP	On Line Analytical Processing
OLTP	On Line Transaction Processing
ETL	Extract, Transform, Load
ELT	Extract, Load, Transform
EAI	Enterprise Application Integration
ERP	Enterprise Resource Planning
DBMS	Database Management System
SQL	Structured Query Language

Slovník pojmů

ODS: Short for ***o****perational* ***d****ata* ***s****tore*, a type of database that serves as an interim area for a data warehouse in order to store time-sensitive operational data that can be accessed quickly and efficiently. In contrast to a data warehouse, which contains large amounts of static data, an ODS contains small amounts of information that is updated through the course of business transactions. An ODS will perform numerous quick and simple queries on small amounts of data, such as acquiring an account balance or finding the status of a customer order, whereas a data warehouse will perform complex queries on large amounts of data. An ODS contains only current operational data while a data warehouse contains both current and historical data.

DataMart: A database, or collection of databases, designed to help managers make strategic decisions about their business. Whereas a data warehouse combines databases across an entire enterprise, data marts are usually smaller and focus on a particular subject or department. Some data marts, called *dependent data marts,* are subsets of larger data warehouses.

Meta Data: Data about data. Metadata describes how and when and by whom a particular set of data was collected, and how the data is formatted. Metadata is essential for understanding information stored in data warehouses and has become increasingly important in XML-based Web applications.

SQL (někdy vyslovováno anglicky *es-kjů-el*, někdy též *síkvl*) je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. SQL je zkratka anglických slov **Structured Query Language** (strukturovaný dotazovací jazyk).

DWH: Abbreviated *DW*, a collection of data designed to support management decision making. Data warehouses contain a wide variety of data that present a coherent picture of business conditions at a single point in time.
Development of a data warehouse includes development of systems to extract data from operating systems plus installation of a warehouse database system that provides managers flexible access to the data.
The term data warehousing generally refers to the combination of many different databases across an entire enterprise. Contrast with *data mart*.

BI: Most companies collect a large amount of data from their business operations. To keep track of that information, a business and would need to use a wide range of software programs , such as Excel, Access and different database applications for various departments throughout their organization. Using multiple software programs makes it difficult to retrieve information in a timely manner and to perform analysis of the data.
The term Business Intelligence (BI) represents the tools and systems that play a key role in the strategic planning process of the corporation. These systems allow a company to gather, store, access and analyze corporate data to aid in decision-making. Generally these systems will illustrate business intelligence in the areas of customer profiling, customer support, market research, market segmentation, product profitability, statistical analysis, and inventory and distribution analysis to name a few.

A **Database Management System** (**DBMS**) is a set of computer programs that controls the creation, maintenance, and the use of a database. Details on http://en.wikipedia.org/wiki/Database_management_system

Slovník pojmů

OLAP: Short for ***O**nline **A**nalytical **P**rocessing,* a category of software tools that provides analysis of data stored in a database. OLAP tools enable users to analyze different dimensions of multidimensional data. For example, it provides time series and trend analysis views. OLAP often is used in data mining.
The chief component of OLAP is the OLAP server, which sits between a client and a database management systems (DBMS). The OLAP server understands how data is organized in the database and has special functions for analyzing the data. There are OLAP servers available for nearly all the major database systems.

ETL: Short for ***e**xtract, **t**ransform, **l**oad*, three database functions that are combined into one tool to pull data out of one database and place it into another database.

Extract -- the process of reading data from a database.

Transform -- the process of converting the extracted data from its previous form into the form it needs to be in so that it can be placed into another database. Transformation occurs by using rules or lookup tables or by combining the data with other data.

Load -- the process of writing the data into the target database.
ETL is used to migrate data from one database to another, to form data marts and data warehouses and also to convert databases from one format or type to another.

OLTP: Short for ***O**n-**L**ine **T**ransaction **P**rocessing*. Same as *transaction processing*.

Transaction processing: A type of computer processing in which the computer responds immediately to user requests. Each request is considered to be a *transaction.* Automatic teller machines for banks are an example of transaction processing.
The opposite of transaction processing is batch processing, in which a batch of requests is stored and then executed all at one time. Transaction processing requires interaction with a user, whereas batch processing can take place without a user being present.

EAI: Acronym for ***e**nterprise **a**pplication **i**ntegration*. EAI is the unrestricted sharing of data and business processes throughout the networked applications or data sources in an organization. Early software programs in areas such as inventory control, human resources, sales automation and database management were designed to run independently, with no interaction between the systems. They were custom built in the technology of the day for a specific need being addressed and were often proprietary systems. As enterprises grow and recognize the need for their information and applications to have the ability to be transferred across and shared between systems, companies are investing in EAI in order to streamline processes and keep all the elements of the enterprise interconnected.

ERP: Short for ***e**nterprise **r**esource **p**lanning,* a business management system that integrates all facets of the business, including planning, manufacturing, sales, and marketing. As the ERP methodology has become more popular, software applications have emerged to help business managers implement ERP in business activities such as inventory control, order tracking, customer service, finance and human resources.

Datový sklad (Data Warehouse)

- Definice (W.H. Inmon 1996):

Datový sklad je

- subjektově orientovaný
- integrovaný
- časově proměnný
- stálý

soubor dat, který slouží pro podporu rozhodování.

Datový sklad

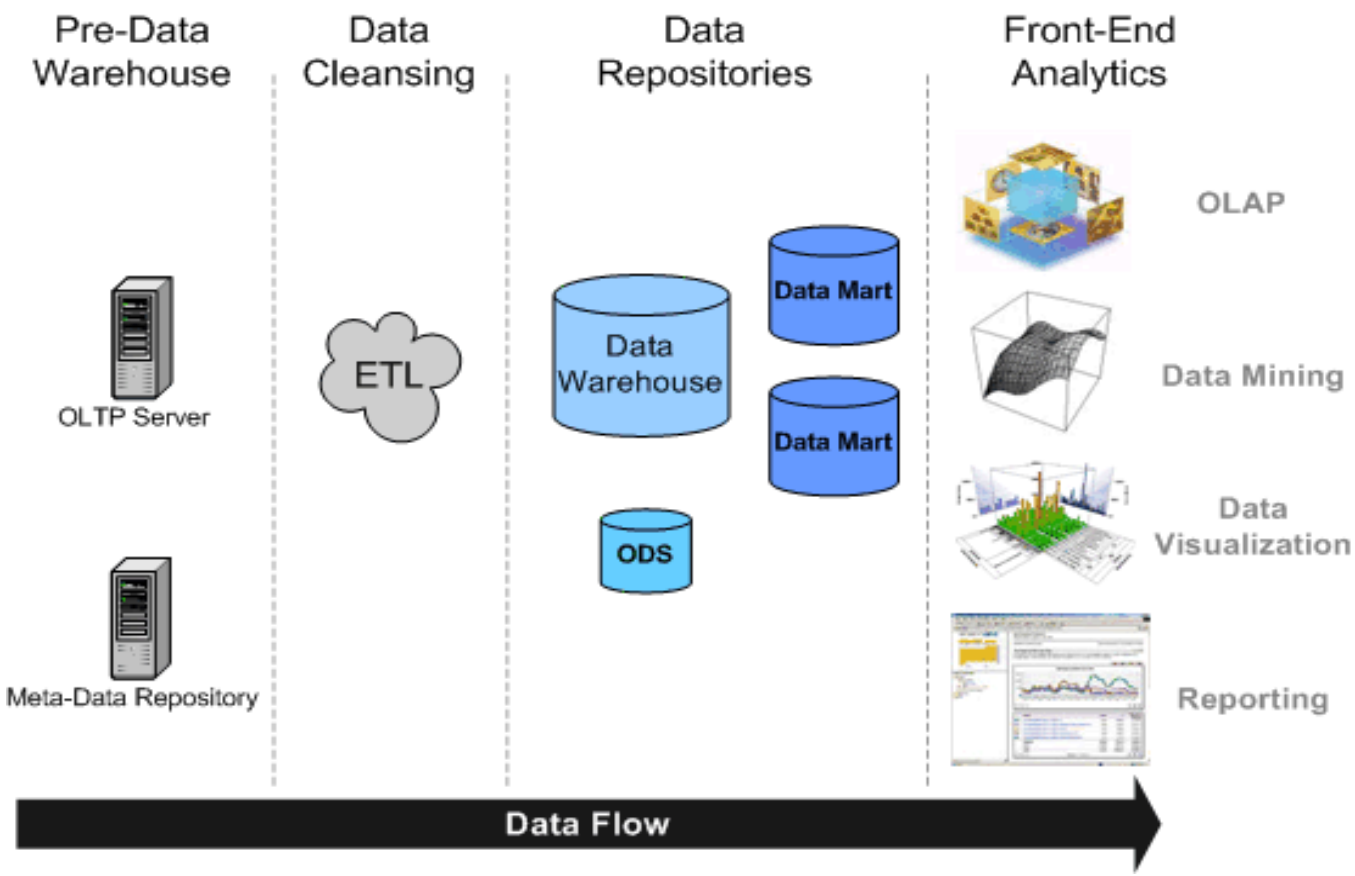
- prvotní koncepce datována počátkem 80.let
- vznik z potřeby jednoduchého přístupu ke strukturovanému úložišti kvalitních dat
- pomáhá získat odpovědi pro lepší rozhodování
- umožňuje použití dat pro dotazování, reportování a analýzu

Struktura datového skladu

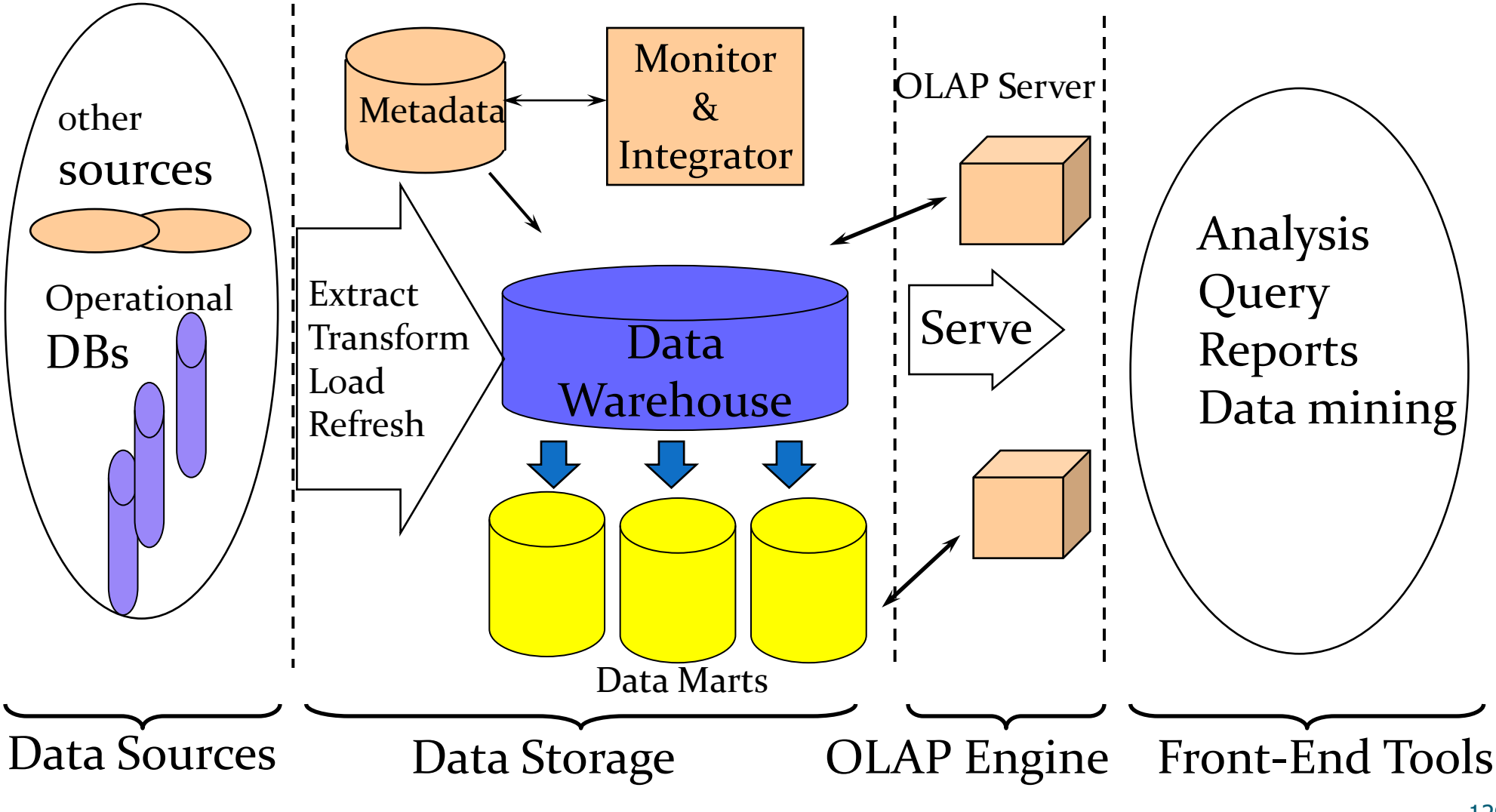
- třívrstvá architektura:
  - datový sklad
  - aplikační vrstva
  - prezentační vrstva

- fyzicky centralizovaný nebo distribuovaný

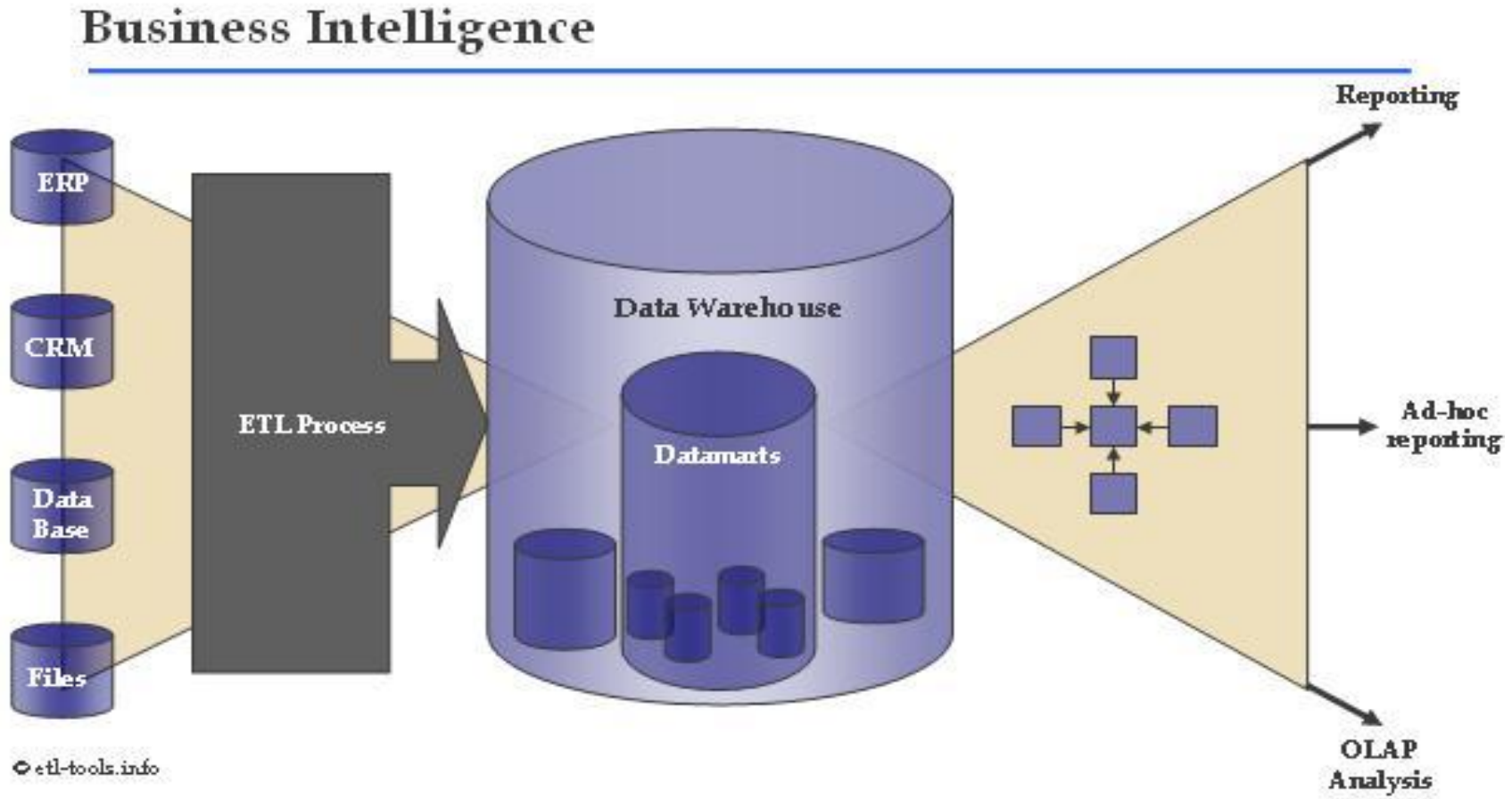
Datový sklad



Datový sklad



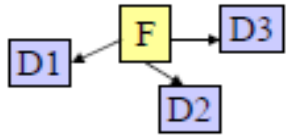
Datový sklad



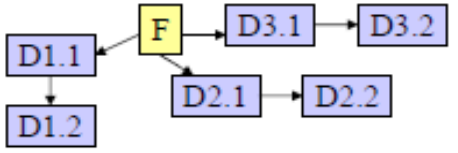
Datové Modely

- Star (hvězda)
- Snowflake (vločka)
- Starflake
- Constellation (souhvězdí)

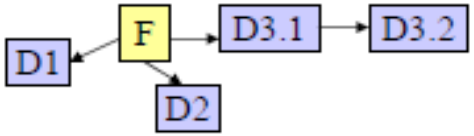
• Star Schema



• Snowflake Schema

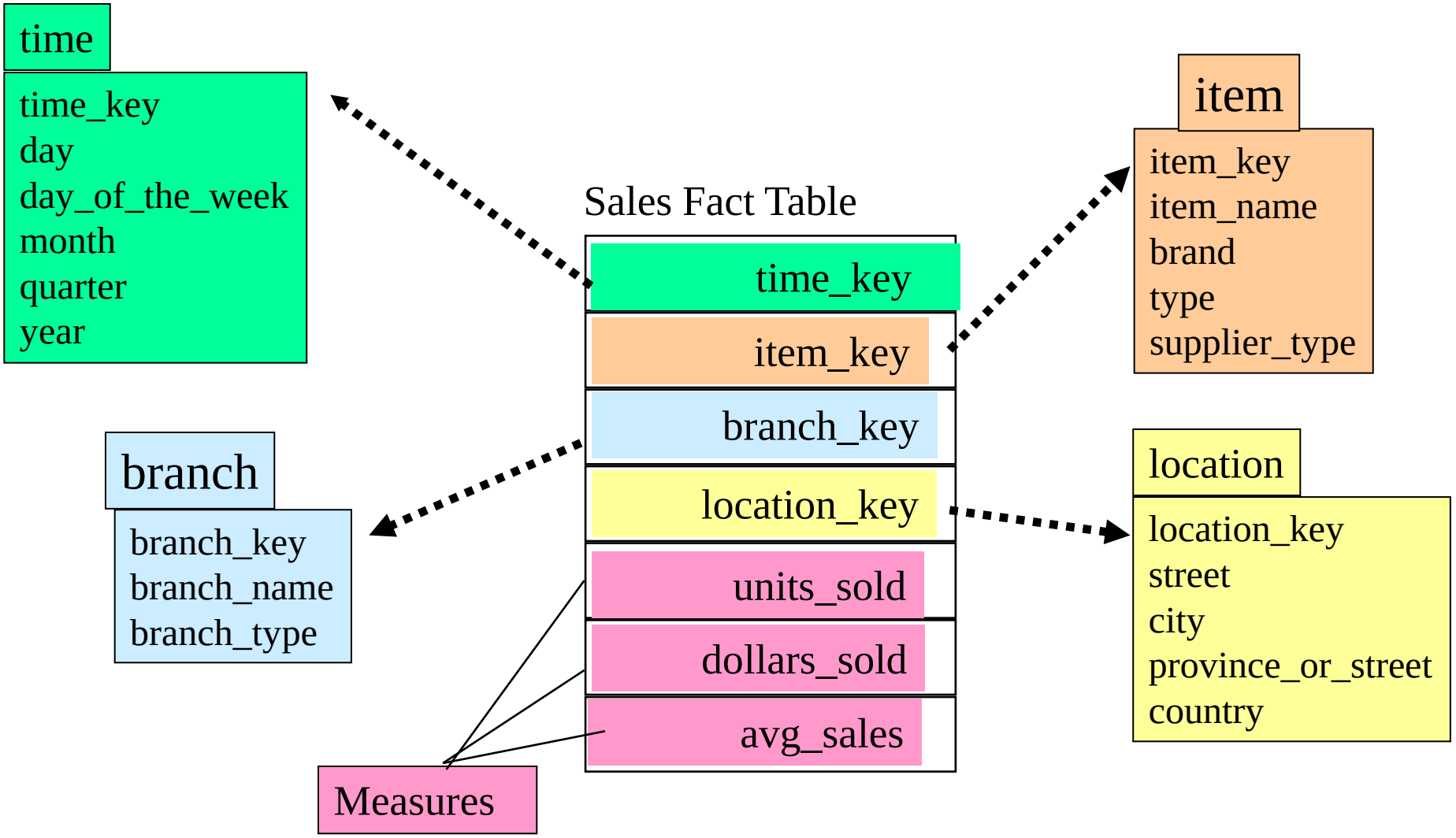


• Starflake Schema

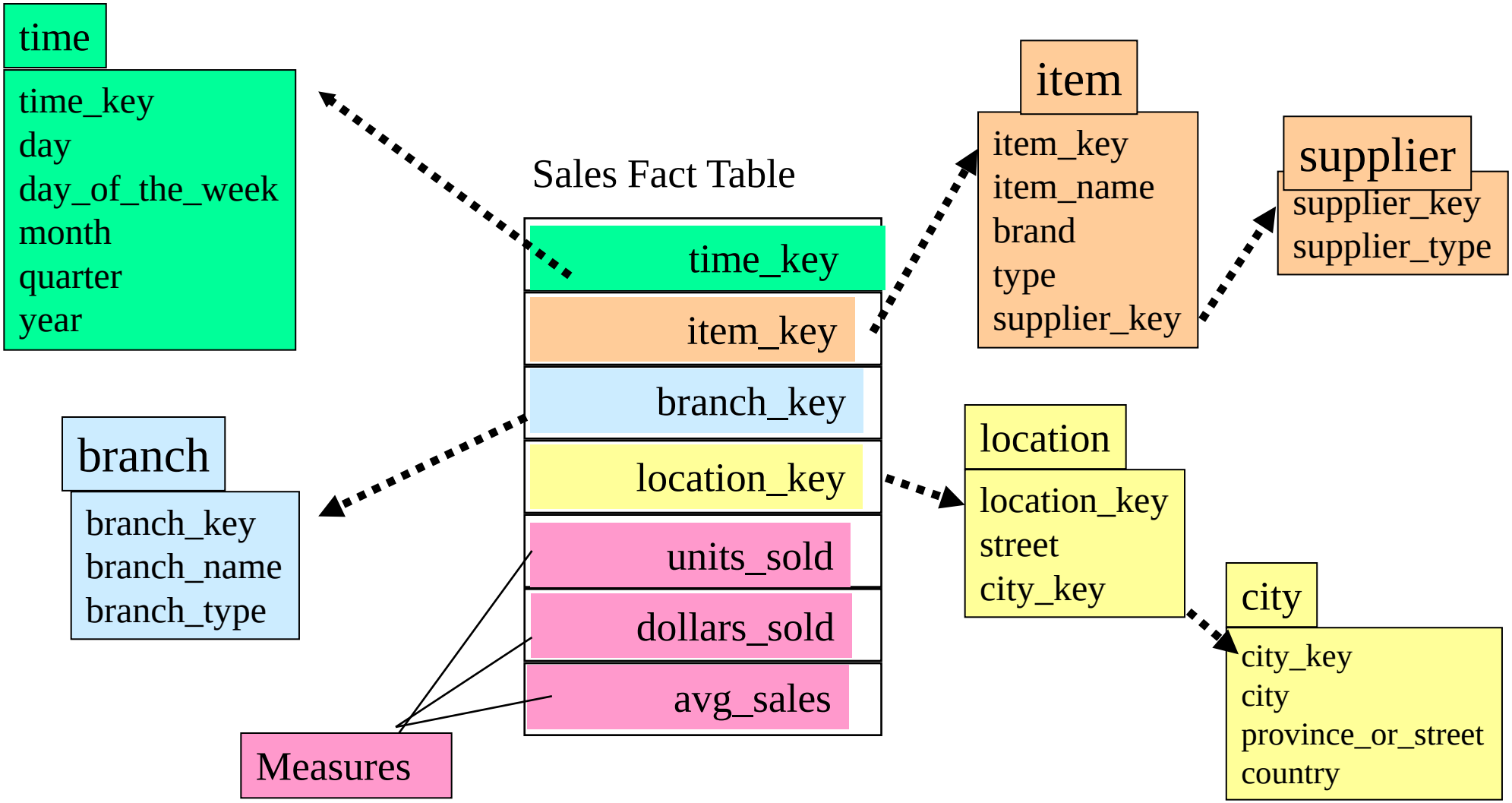


• Constellation Schema

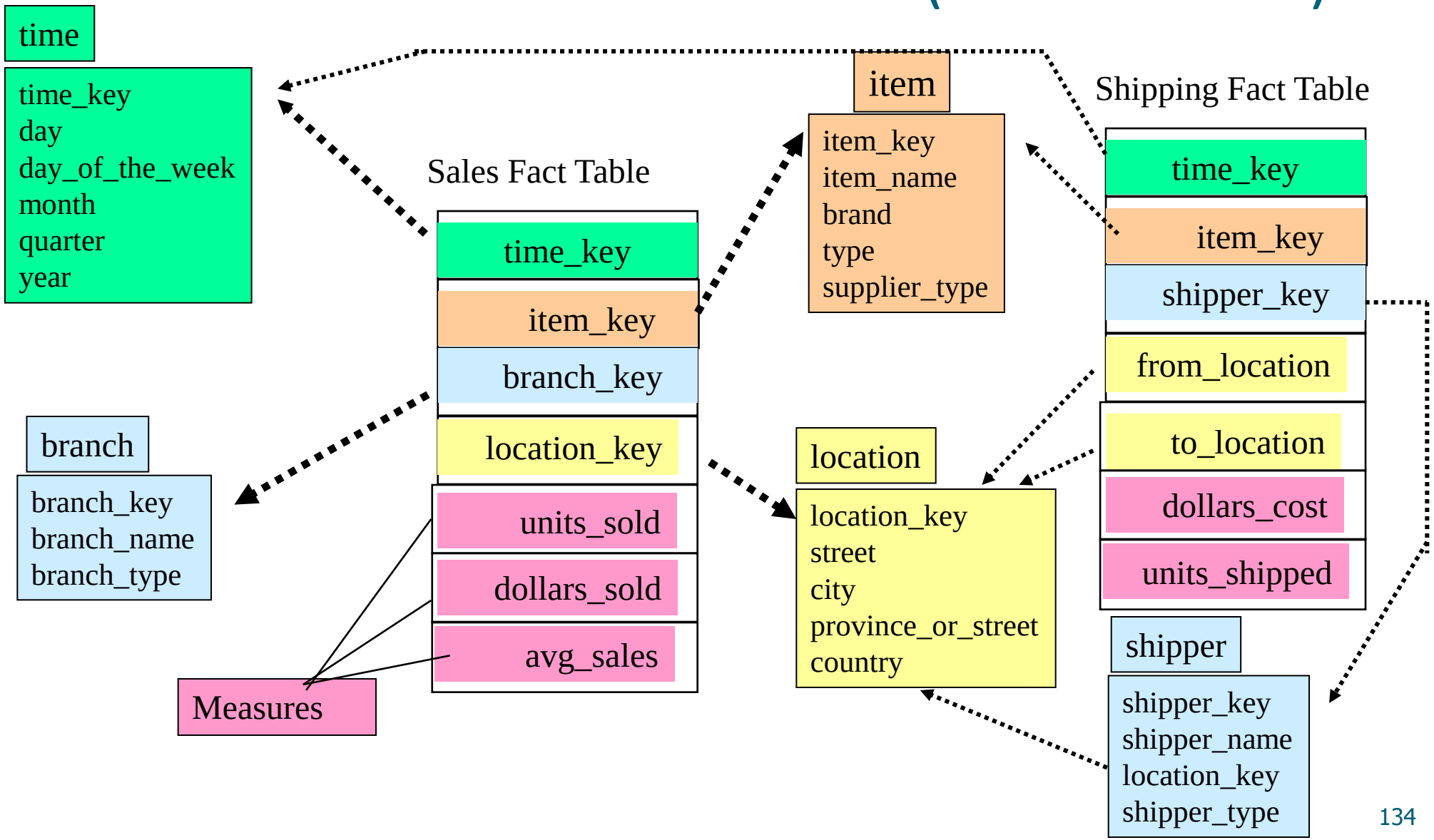
Příklad schématu hvězda (star)



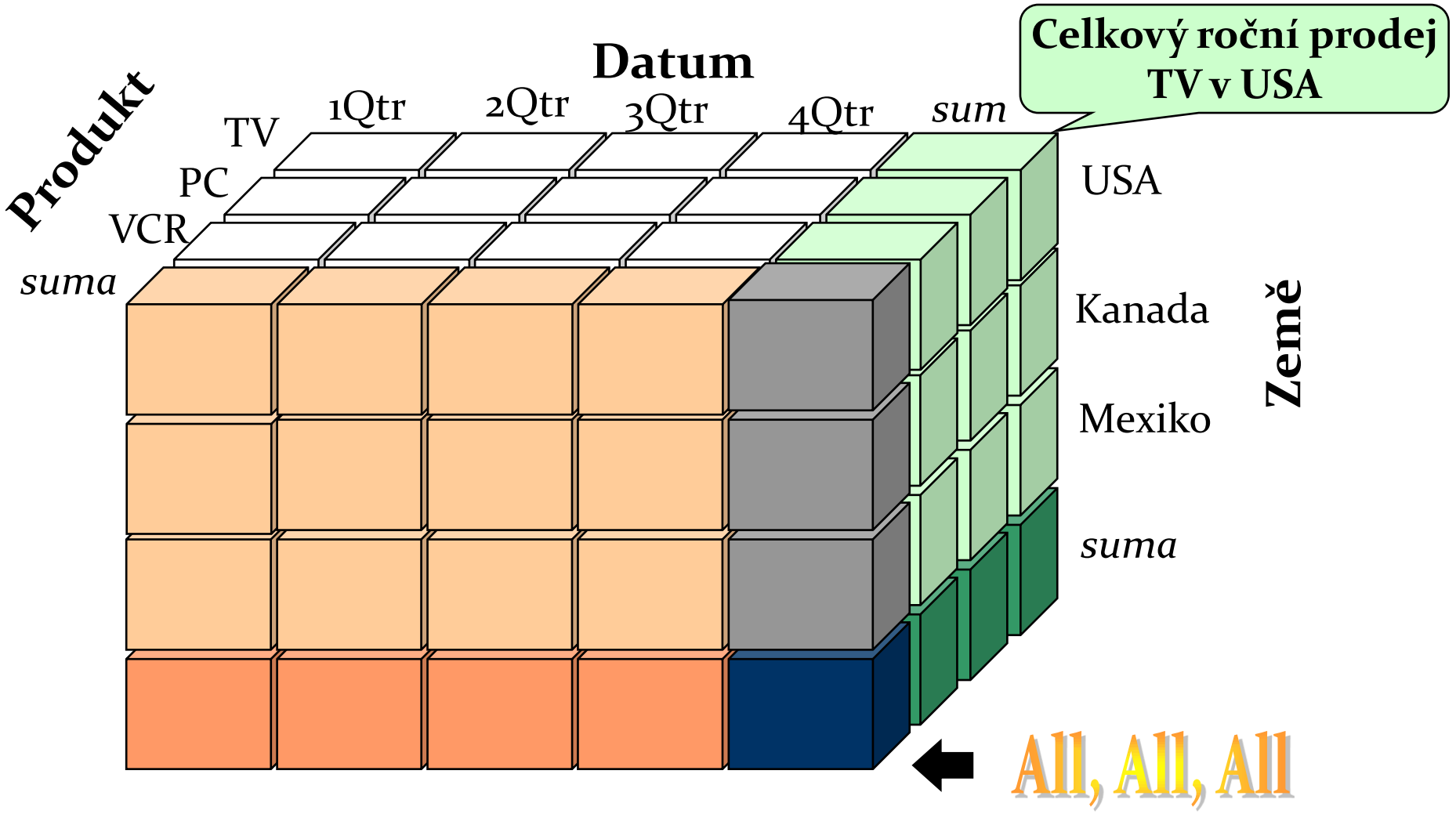
Příklad schématu vločka (Snowflake)



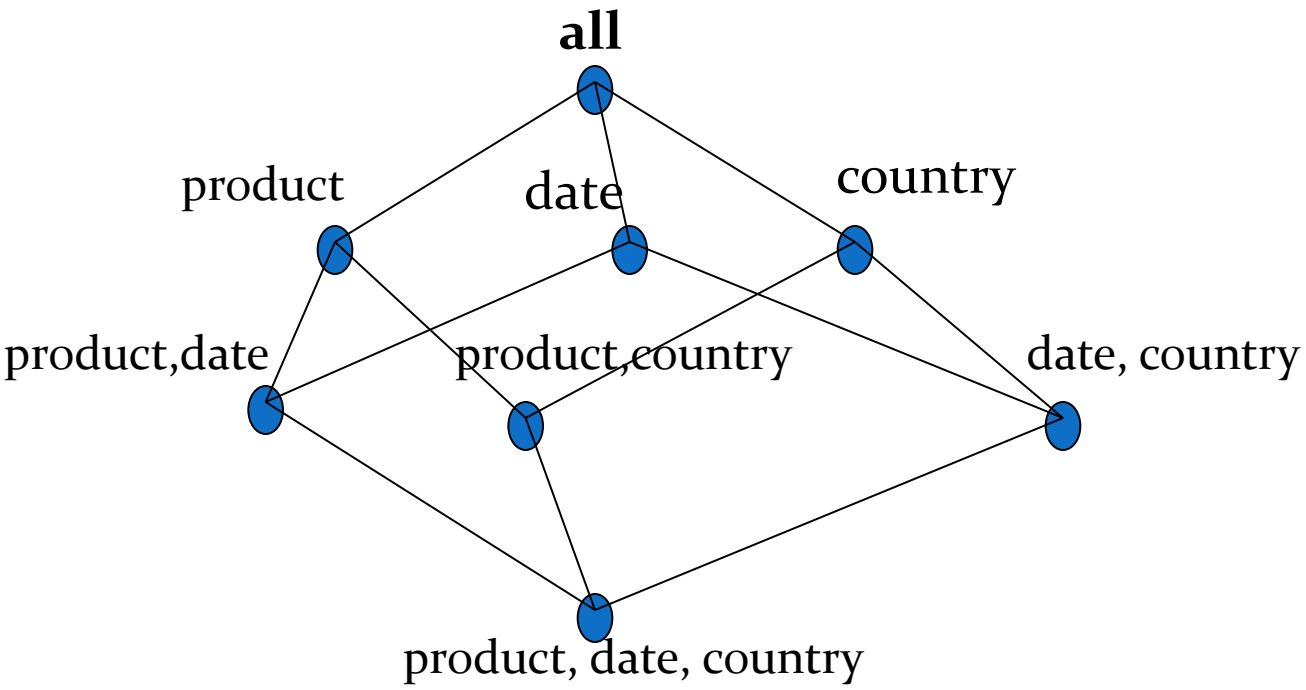
Příklad schématu souhvězdí (Constellation)



Příklad datové kostky



Datové „kvádry“ odpovídající datové kostce



0-D(apex) cuboid

1-D cuboids

2-D cuboids

3-D(base) cuboid

Typické OLAP Operace

- Roll up (drill-up): sumarizace dat
  - *Postoupení v hierarchii o úroveň výše nebo redukce dimenze (např. z kostky na čtverec).*
- Drill down (roll down): opak roll-up –zajímá nás větší detail
  - *Z vyšší úrovně sumarizace na nižší úroveň nebo zavedení nových datových dimenzí.*
- Slice and dice (krájet a kostkovat):
  - *Výběr datového podprostoru.*
- Ostatní operace:
  - *drill across: zahrnutí více datových tabulek (kostek)*
  - *drill through: přes základní úroveň datové kostky zpět k podkladovým relačním tabulkám (pomocí SQL)*

Architektura OLAP Serverů

- Relační OLAP (Relational OLAP -ROLAP)
  - Využívá relační nebo rozšířenou relační DBMS pro ukládání a správu dat datového skladu a OLAPovou střední vrstvu pro podporu chybějících částí.
  - Zahrnuje optimalizační možnosti DBMS, implementaci agregační navigační logiky a doplňkové nástroje a služby.
- Vícedimenzionální OLAP (Multidimensional OLAP - MOLAP)
  - Technologie založená na vícedimenzionálních datových polích (vč. technik pro řídké matice).
  - Rychlé indexování předem spočtených sumarizovaných dat.
- Hybridní OLAP (Hybrid OLAP - HOLAP)
  - Uživatelsky flexibilní, tj. low level: relační, high-level: pole.
- Specializované SQL servery
  - specializovaná podpora pro SQL dotazy nad star/snowflake schématy.

ROLAP

- Data uložená v relační databázi – nejsou duplikována, ovšem není k nim možný přístup bez připojení k zdrojové databázi.

- dotazy OLAP se převádějí do klasických dotazů SQL – může být nevýhodou (limitované možnosti SQL, pomalejší odezva).

- Vhodný jen pro omezené množství dat.

MOLAP

- „tradiční“ OLAP.
- Data uložena v multidimenzionálních kostkách mimo relační databázi. Jsou tudíž duplikována a je možný přístup i bez spojení s původním zdrojem dat.
- Hlavní výhodou je rychlá odezva na dotazy. Vše je předpočítáno a uloženo při tvorbě kostek.

HOLAP

- ponechává původní data v relačních tabulkách, agregace ukládá v multidimenzionálním formátu
- poskytuje propojení mezi rozsáhlými objemy dat v relačních tabulkách
- výhoda rychlejšího výkonu multidimenzionálně uložených agregací

Budování datového skladu

- metoda „velkého třesku“:
  - analýza požadavků podniku
  - vytvoření podnikového datového skladu
  - vytvoření datových tržišť

- přírůstková (evoluční) metoda

Plnění datového skladu

- počáteční plnění + pravidelná aktualizace

- plnění pomocí datových pump
- postupy ETL:
  - extrakce
  - transformace
  - loading

Co je SQL?

The SQL procedure uses Structured Query Language to perform the following tasks:

- retrieve and manipulate SAS data sets
- create and delete SAS data sets
- generate reports
- add or modify values in a SAS data set
- add, modify, or drop columns in a SAS data set

Úvod do SQL

- General form of an SQL procedure query to generate output:

```
PROC SQL;
    SELECT variables
        FROM SAS-data-set;
```

Úvod do SQL

- Create a listing report of product activity.
  - Step 1: Invoke the SQL procedure.

```
proc sql;
```

  - Step 2: Identify the variables to display on the report.

```
proc sql;
    select CustomerID, CustomerFirstName,
           CustomerLastName
```

Úvod do SQL

- Step 3: Identify the input data set.

```
proc sql;
   select CustomerID, CustomerFirstName,
          CustomerLastName
      from univ.mastercustomers;
```

- Step 4: End the procedure with a QUIT statement.

```
proc sql;
   select CustomerID, CustomerFirstName,
          CustomerLastName
      from univ.mastercustomers;
quit;
```

Úvod do SQL

- SQL joins have the following characteristics:
  - They do not require sorted data.
  - They can be performed on up to 32 data sets at one time.
  - They allow complex matching criteria using the WHERE clause.

Úvod do SQL

- General form of an SQL procedure join to generate output:

```
PROC SQL;
      SELECT variables
           FROM SAS-data-set1 AS alias1,
                        SAS-data-set2 AS alias2
          WHERE alias1.variable=alias2.variable;
```

Úvod do SQL

- Create a listing report by joining data sets **univ.mastercustomers** and **univ.customerorders** by **CustomerID**.

  - Step 1: Invoke the SQL procedure and list the variables to display.

```
proc sql;
   select CustomerID, CustomerFirstName,
          CustomerLastName, OrderID,
          UnitPrice, Quantity
```

Úvod do SQL

- Step 2: Identify the data sets to join and provide a table alias for each.

- Because **CustomerID** exists in both data sets, identify which **CustomerID** to use.

```
proc sql;
   select m.CustomerID, CustomerFirstName,
          CustomerLastName, OrderID,
          UnitPrice, Quantity
      from univ.mastercustomers as m,
         univ.customerorders as c
```

Úvod do SQL

- Step 3: State the condition on which observations are matched and terminate the query.

```
proc sql;
   select m.CustomerID, CustomerFirstName,
          CustomerLastName, OrderID,
          UnitPrice, Quantity
      from univ.mastercustomers as m,
           univ.customerorders as c
      where m.CustomerID=c.CustomerID;
quit;
```

Úvod do SQL

Create a new variable named **TotSale** by multiplying **Quantity** by **UnitPrice**. Name the new variable **TotSale**.

```
proc sql;
   select m.CustomerID, CustomerFirstName,
          CustomerLastName, OrderID,
          UnitPrice, Quantity,
          Quantity * UnitPrice as TotSale
      from univ.mastercustomers as m,
           univ.customerorders as c
      where m.CustomerID=c.CustomerID;
quit;
```

Úvod do SQL

- General form of a PROC SQL query to create a SAS data set:

```
PROC SQL;
    CREATE TABLE SAS-data-set AS
        SELECT ...
        other SQL clauses;
```

Úvod do SQL

- Join the tables **univ.mastercustomers** and **univ.customerorders** to create a new data set.

```
proc sql;
   create table work.ordertotals as
      select m.CustomerID,
             CustomerFirstName,
             CustomerLastName, OrderID,
             UnitPrice, Quantity,
             Quantity*UnitPrice as TotSale
         from univ.mastercustomers as m,
              univ.customerorders as c
         where m.CustomerID=c.CustomerID;
quit;
```

Úvod do SQL

- General form of an SQL procedure query using labels and formats:

```
PROC SQL;
      SELECT variable LABEL='column-header'
                        FORMAT=format.
        FROM SAS-data-set ;
```

Úvod do SQL

- Enhance the previous report.

```
proc sql;
   select m.CustomerID,
          CustomerFirstName format=$10.,
          CustomerLastName format=$15.,
          OrderID,
            UnitPrice format=dollar7.2,
          Quantity,
          Quantity * UnitPrice as TotSale
               format=dollar8.2
               label='Total Sale Amount'
      from univ.mastercustomers as m,
             univ.customerorders as c
      where m.CustomerID=c.CustomerID;
quit;
```

Úvod do SQL

- Partial Output

Customer ID	Customer First Name	Customer Last Name	OrderID	Unit Price	Quantity	Sale Amount
062096	Craig	Knapmeyer	1240062267	$36.00	3	$108.00
062096	Craig	Knapmeyer	1240832690	$27.00	4	$108.00
062284	Robert	Britt	1238409388	$15.00	1	$15.00
062284	Robert	Britt	1238409388	$33.00	1	$33.00
064810	Randall	Goodman	1238248877	$175.00	4	$700.00
064810	Randall	Goodman	1238248877	$283.00	1	$283.00
064810	Randall	Goodman	1238273875	$220.00	1	$220.00
064810	Randall	Goodman	1238768955	$52.00	1	$52.00
064810	Randall	Goodman	1238842450	$24.00	1	$24.00
064810	Randall	Goodman	1239353817	$59.00	2	$118.00
064810	Randall	Goodman	1239489696	$11.00	2	$22.00
064810	Randall	Goodman	1239608721	$22.00	3	$66.00
064810	Randall	Goodman	1239608721	$46.00	3	$138.00
064810	Randall	Goodman	1240590287	$21.00	2	$42.00

Úvod do SQL

- General form of an SQL procedure query to generate summary output:

```
PROC SQL;
      SELECT group-variable,
                  SUM(analysis-variable)
            FROM SAS-data-set
            GROUP BY group-variable;
```

- If a summary function is used in the SELECT clause with only one argument, then an overall statistic is calculated down the column.

Úvod do SQL

- Step 1: Identify the variables to display, the input data sets, and the matching criteria.

```
proc sql;
   select m.CustomerID,
          CustomerFirstName format=$10.,
          CustomerLastName format=$15.,
          sum(Quantity) label= 'Total Quantity',
          sum(Quantity*UnitPrice) as TotSale
                 format=dollar12.2
                 label='Total Sale Amount'
      from univ.mastercustomers as m,
           univ.customerorders as c
      where m.CustomerID=c.CustomerID;
```

Úvod do SQL

- Step 2: Identify the grouping variable(s).

```
proc sql;
   select m.CustomerID,
          CustomerFirstName format=$10.,
          CustomerLastName format=$15.,
          sum(Quantity) label='Total Quantity',
          sum(Quantity*UnitPrice) as TotSale
               format=dollar12.2
               label='Total Amount Purchased'
     from univ.mastercustomers as m,
          univ.customerorders as c
     where m.CustomerID=c.CustomerID
     group by m.CustomerID, CustomerFirstName,
              CustomerLastName;
quit;
```

Úvod do SQL

- General form of an SQL procedure query to generate ordered output:

```
PROC SQL;
      SELECT group-variable,
                        SUM(analysis-variable)
           FROM SAS-data-set
            GROUP BY group-variable
            ORDER BY variable1 <, variable2> ;
```

- The default is ascending order.

Úvod do SQL

- Order the report by total sale.

```
proc sql;
   select m.CustomerID,
          CustomerFirstName format=$10.,
          CustomerLastName format=$15.,
          sum(Quantity) label='Total Quantity',
          sum(Quantity*UnitPrice) as TotSale
               format=dollar12.2
               label='Total Amount Purchased'
     from univ.mastercustomers as m,
          univ.customerorders as c
     where m.CustomerID=c.CustomerID
     group by m.CustomerID, CustomerFirstName,
              CustomerLastName
     order by TotSale;
quit;
```

Úvod do SQL

- Order the report by total sale – **v sestupném pořadí**

```
proc sql;
   select m.CustomerID,
          CustomerFirstName format=$10.,
          CustomerLastName format=$15.,
          sum(Quantity) label='Total Quantity',
          sum(Quantity*UnitPrice) as TotSale
               format=dollar12.2
               label='Total Amount Purchased'
     from univ.mastercustomers as m,
          univ.customerorders as c
     where m.CustomerID=c.CustomerID
     group by m.CustomerID, CustomerFirstName,
              CustomerLastName
     order by TotSale desc;
quit;
```

Inner JOIN

- The INNER JOIN keywords can be used to join tables. The ON clause replaces the WHERE clause for specifying columns to join. PROC SQL provides these keywords primarily for compatibility with the other joins (OUTER, RIGHT, and LEFT JOIN). Using INNER JOIN with an ON clause provides the same functionality as listing tables in the FROM clause and specifying join columns with a WHERE clause.

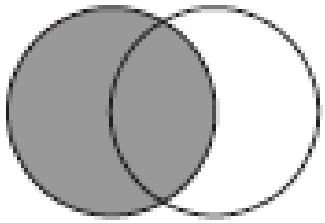
```
proc sql outobs=6;
title 'Oil Production/Reserves
of Countries';
select p.country, barrelsperday
'Production', barrels
'Reserves'
from sql.oilprod p,
sql.oilrsrvs r
where p.country = r.country
order by barrelsperday desc;
```

=

```
proc sql ;
select p.country,
barrelsperday
'Production', barrels
'Reserves'
from sql.oilprod p inner
join sql.oilrsrvs r
on p.country = r.country
order by barrelsperday
desc;
```

Left JOIN

- *Outer joins are inner joins that are augmented with rows from one table that do not* match any row from the other table in the join. The resulting output includes rows that match and rows that do not match from the join's source tables. Nonmatching rows have null values in the columns from the unmatched table. Use the ON clause instead of the WHERE clause to specify the column or columns on which you are joining the tables. However, you can continue to use the WHERE clause to subset the query result.

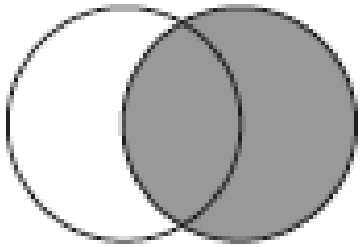


- A left outer join lists matching rows and rows from the left-hand table (the first table listed in the FROM clause) that do not match any row in the right-hand table. A left join is specified with the keywords LEFT JOIN and ON.

```
proc sql;
select Capital format=$20., Name 'Country'
format=$20.,
Latitude, Longitude
from sql.countries a left join sql.worldcitycoords b
on a.Capital = b.City and
a.Name = b.Country;
```

Right JOIN

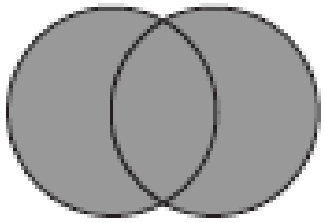
- A right join, specified with the keywords RIGHT JOIN and ON, is the opposite of a left join: nonmatching rows from the right-hand table (the second table listed in the FROM clause) are included with all matching rows in the output.



```
proc sql outobs=10;
select City format=$20., Country
'Country' format=$20., Population
from sql.countries right join
sql.worldcitycoords
on Capital = City and
Name = Country
order by City;
```

Inner/Full Outer/Left/Right JOIN

• A full outer join, specified with the keywords FULL JOIN and ON, selects all matching and nonmatching rows.



```
proc sql outobs=10;
select City '#City#(WORLDCITYCOORDS)'
format=$20.,
Capital '#Capital#(COUNTRIES)'
format=$20.,
Population, Latitude, Longitude
from sql.countries full join
sql.worldcitycoords
on Capital = City and
Name = Country;
```

Speciální typy JOIN

- Cross Join
  - A cross join is a Cartesian product; it returns the product of two tables.
- Union Join
  - A union join combines two tables without attempting to match rows. All columns and rows from both tables are included.
- Natural Join
  - A natural join automatically selects columns from each table to use in determining matching rows. With a natural join, PROC SQL identifies columns in each table that have the same name and type; rows in which the values of these columns are equal are returned as matching rows. The ON clause is implied.

Úvod do SQL

- Další filtrování výstupu pomocí „having“

```
proc sql;
   select CustomerGroup,
          sum(Quantity*UnitPrice) as TotSale
             format=dollar12.2
             label='Total Amount Purchased',
   from customers as m left join
        customerorders as c
   on m.CustomerID=c.CustomerID
   group by m.CustomerGroup
   having TotSale ge 10000 and TotSale ne .
   order by TotSale desc
;
quit;
```

Větší nebo rovno 10000

Různé od „missing“,
tj. chybějící hodnoty

Úvod do SQL

- Zjištění počtu rozdílných (distinct) klientů v daných skupinách:

```
proc sql;
   select CustomerGroup,
          count(distinct m.CustomerID) as pocet,
from customers as m left join
        customerorders as c
   on m.CustomerID=c.CustomerID
   group by m.CustomerGroup
order by TotSale desc
;
quit;
```

Úvod do SQL

- Výpis prvních deseti (obecně n) řádků a všech sloupců:

```
proc sql outobs=10;
    select
    *
    from customers
;
quit;
```

Functions and CALL Routines

- cca 500 „funkcí“
  - textové
  - datumové
  - matematické
  - statistické
  - pravděpodobnostní
  - finanční
  - …

Více na:
http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000245860.htm

Math operators:

- Arithmetic Operators:

Symbol	Definition	Example	Result
**	exponentiation	a**3	raise A to the third power
*	Multiplication	2*y	multiply 2 by the value of Y
/	division	var/5	divide the value of VAR by 5
+	addition	num+3	add 3 to the value of NUM
-	subtraction	sale-discount	subtract the value of DISCOUNT from the value of SALE

Math operators:

- Comparison Operators :

Symbol	Mnemonic Equivalent	Definition	Example
=	EQ	equal to	a=3
^=	NE	not equal to (table note 1)	a ne 3
¬=	NE	not equal to	
~=	NE	not equal to	
>	GT	greater than	num>5
<	LT	less than	num<8
>=	GE	greater than or equal to (table note 2)	sales>=300
<=	LE	less than or equal to (table note 3)	sales<=100
	IN	equal to one of a list	num in (3, 4, 5)

Math operators:

- Logical Operators:

Symbol	Mnemonic Equivalent	Example
&	AND	(a>b & c>d)
|	OR	(a>b or c>d)
!	OR	
¦	OR	
¬	NOT	not(a>b)
^	NOT	
~	NOT	

Více na:
http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a000780367.htm

Special WHERE Operators

- *Special WHERE operators* are operators that can only be used in a WHERE expression.

Symbol	Mnemonic	Definition
	BETWEEN-AND	an inclusive range
	IS NULL	missing value
	IS MISSING	missing value
?	CONTAINS	a character string
	LIKE	a character pattern

BETWEEN-AND Operator

- The *BETWEEN-AND operator* selects observations in which the value of a variable falls within an inclusive range of values.

- Examples:

```
where salary between 50000 and 100000;
```

```
where salary not between 50000 and 100000;
```

- Equivalent Expressions:

```
where salary between 50000 and 100000;
```

```
where 50000 <= salary <= 100000;
```

IS NULL and IS MISSING Operators

- The *IS NULL* and *IS MISSING operators* select observations in which the value of a variable is missing.
  - The operator can be used for both character and numeric variables.
  - You can combine the NOT logical operator with IS NULL or IS MISSING to select nonmissing values.
- Examples:

```
where Employee_ID is null;
```

```
where Employee_ID is missing;
```

CONTAINS Operator

- The *CONTAINS (?) operator* selects observations that include the specified substring.
  - The position of the substring within the variable's values does not matter.
  - The operator is case sensitive when comparisons are made.
- Example:

```
where Job_Title contains 'Rep';
```

LIKE Operator

- The *LIKE operator* selects observations by comparing character values to specified patterns.

- There are two special characters available for specifying a pattern:
  - A percent sign (%) replaces any number of characters.
  - An underscore (_) replaces one character.

- Consecutive underscores can be specified.
- A percent sign and an underscore can be specified in the same pattern.
- The operator is case sensitive.

LIKE Operator

- Examples:

```
where Name like '%N';
```

- This WHERE statement selects observations that begin with any number of characters and end with an N.

```
where Name like 'T_M%';
```

- This WHERE statement selects observations that begin with a T, followed by a single character, followed by an M, and followed by any number of characters.
- Possible values include **Tom** and **Tammy**.

Which WHERE statement will return the observations that have a first name starting with the letter M for the given values?

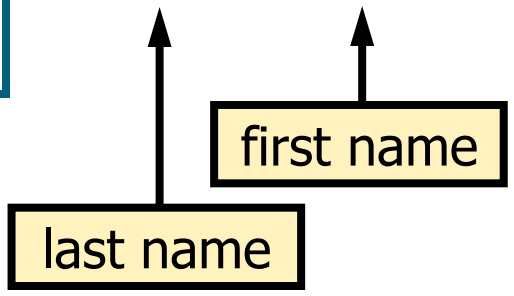
a.
```
where Name like '_, M_';
```

b.
```
where Name like '%, M%';
```

c.
```
where Name like '_, M%';
```

d.
```
where Name like '%, M_';
```

Name
Elvish, Irenie
Ngan, Christina
Hotstone, Kimiko
Daymond, Lucian
Hofmeister, Fong
Denny, Satyakam
Clarkson, Sharryn
Kletschkus, Monica



Which WHERE statement will return the observations that have a first name starting with the letter M for the given values?

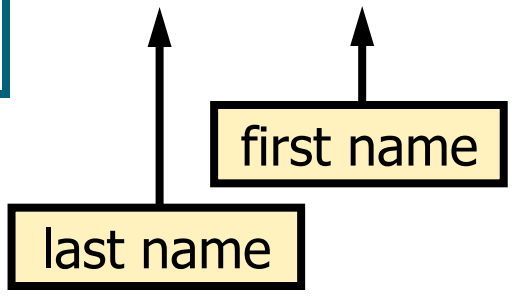
a.
```
where Name like '_, M_';
```

b.
```
where Name like '%, M%';
```

c.
```
where Name like '_, M%';
```

d.
```
where Name like '%, M_';
```

Name
Elvish, Irenie
Ngan, Christina
Hotstone, Kimiko
Daymond, Lucian
Hofmeister, Fong
Denny, Satyakam
Clarkson, Sharryn
Kletschkus, Monica



Odkazy na právě vypočtené sloupce

- When you use a column alias to refer to a calculated value, you must use the CALCULATED keyword with the alias to inform PROC SQL that the value is calculated within the query. The following example uses two calculated values, LowC and HighC, to calculate a third value, Range:

```
proc sql outobs=12;
title 'Range of High and Low
Temperatures in Celsius';
select City, (AvgHigh - 32) * 5/9
as HighC format=5.1,
(AvgLow - 32) * 5/9 as LowC
format=5.1,
(calculated HighC - calculated
LowC)
as Range format=4.1
from sql.worldtemps;
```

- You can specify a calculated column only in a SELECT clause or a WHERE clause

Podmíněné přiřazení hodnoty

- You can use conditional logic within a query by using a CASE expression to conditionally assign a value. You can use a CASE expression anywhere that you can use a column name. You must close the CASE logic with the END keyword.

```
proc sql outobs=12;
   title 'Climate Zones of World Cities';
   select City, Country, Latitude,
      case
        when Latitude gt 67 then 'North Frigid'
        when 67 ge Latitude ge 23 then 'North Temperate'
        when 23 gt Latitude gt -23 then 'Torrid'
        when -23 ge Latitude ge -67 then 'South Temperate'
        else 'South Frigid'
      end as ClimateZone
   from sql.worldcitycoords
   order by City;
```

Více o proceduře SQL

SAS® 9.2 SQL Procedure
User's Guide

http://support.sas.com/documentation/cdl/en/sqlproc/62086/PDF/default/sqlproc.pdf

SAS(R) 9.2 SQL Procedure User's Guide – Knowlwdge Base:

http://support.sas.com/documentation/cdl/en/sqlproc/62086/HTML/default/a002536894.htm

Datová matice

- Nutný formát dat pro modelování.
- 2-rozměrná matice $n \times p$.
- Řádky reprezentují n statistických jednotek (klientů)
- Sloupce reprezentují p statistických proměnných.

f:\apps\spsswin\bank.sav

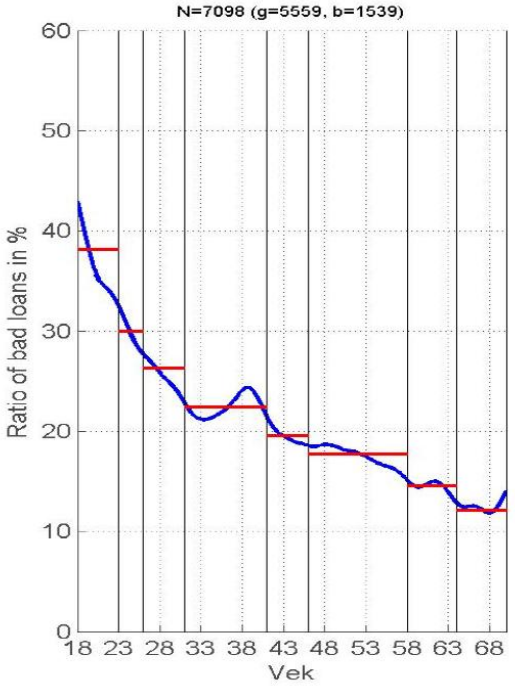
1:id 628

	id	salbeg	sex	time	age	salnow	edlevel	work	jobcat	minority	sexr
1	628	8400	0	81	28.50	16080	16	.25	4	0	
2	630	24000	0	73	40.33	41400	16	12.50	5	0	
3	632	10200	0	83	31.08	21960	15	4.08	5	0	
4	633	8700	0	93	31.17	19200	16	1.83	4	0	
5	635	17400	0	83	41.92	28350	19	13.00	5	0	
6	637	12996	0	80	29.50	27250	18	2.42	4	0	
7	641	6900	0	79	28.00	16080	15	3.17	1	0	
8	649	5400	0	67	28.75	14100	15	.50	1	0	

Zásady tvorby datové matice

- Replikovatelnost tvorby dat. matice
  - žádné manuální úpravy dat
- Srozumitelnost tvorby dat. matice
  - podrobné komentáře
- Zpětná konektivita dat. matice
  - primární klíče (id) všech podkladových datových tabulek

4. Příprava dat –čištění, kategorizace, agregace, transformace dat (WOE), úvod do SAS Data Step



Čištění dat: Praktické zkušenosti

- Pokud vaše nová data obsahují více než 30 čísel, tak je v nich skoro jistě nějaká chyba.
- **Čištění a příprava dat zabírá obvykle 80 – 90 % analytikova času.**
- Pokud budete VELMI pečliví v této fázi, ušetříte si daleko víc času a nervů později – jinak stavíte dům na písku.
- GIGO…Garbage in, Garbage out (smetí dovnitř, smetí ven)
  - sebelepší model (proces) nevyrobí ze smetí nic jiného než opět smetí.

Co způsobí nekvalitní data

- Správa nekvalitních/nadbytečných dat
- Nedoručené zásilky (marketing, fakturace)
- Nesprávné výsledky zpracování (reporting, analýzy, data mining)
- Špatné fungování systému (nekompatibilita)
- Ztráta image, nespokojení klienti

Co způsobí nekvalitní data

- Při mailingové kampani jedné britské **maloobchodní společnosti** se ukázalo, že jedna pětina oslovených už zemřela. Přesto (nebo pro to?) byli obesláni s pozdravným oslovením „**Drahý pane Zesnulý**“. [1)]
- Jistá **pojišťovna** zjistila, že většina jejich zákazníků má **zaměstnání „Astronaut**“ – další pátrání ukázalo, že „Astronaut“ je první volba v seznamu v jejich CRM systému. [1)]
- 44 000-98 000 Američanů ročně umírá na základě **odvratitelné medicínské chyby** jako přepsání při psaní receptu, špatně popsaný výsledek krevní zkoušky, nečitelná informace v pacientských záznamech atd. Je to **osmá nejčastější příčina úmrtí** v USA [2)]
- 7.5.1999 bombardovaly **ozbrojené síly USA** čínské velvyslanectví v Jugoslávii. Vyšetřování zjistilo: CIA používá zastaralý mapový materiál; ještě k tomu pracovník předložil v důsledku chyby v datech **špatnou adresu** – „Doslovně nakreslil X na nesprávné místo“ [3)]

1) Peel, M: Letters to the dead and other data dereliction. © 2007 Financial Times Deutschland. http://www.ftd.de, vydání z 2.10.2007
2) Oash, J. (1999): IT Can Reduce Medical Errors. Obsaženo v: Wang, Pierce, Madnick: Information Quality, 2005
3) BBC: Americas chinese embassy warning ignored. © 1999 BBC. http://news/bbc.co.uk/1/hi/world/americas/37775.stm, vydání z 2.10.2007

Čištění dat: Ověření souboru

- Ověření souboru s daty / zdrojů dat
  - Jsou to správná data (čas vzniku, výzkum…)?
  - Jsou kompletní, bez duplicit, umím je číst…
- Zkoumání případů
  - Mají identifikátory?
    - Jsou tyto ID správné?
  - Neopakují se (duplicity)?
    - Existují i „skoro“ duplicity – dva podobné, ale ne přesně totožné záznamy o tomtéž subjektu.
  - Nejsou vynechány?

Čištění dat: Ověření proměnných

- Zkoumání metadat o proměnných
  - Jsou tam všechny proměnné a správně značené?
  - Je jasné, co znamenají (kódovníky, definice…)? Dokumentace OK?
    - Pozor na mezinárodní studie, produkty konsorcií agentur a opakované vlny výzkumů. Jemné nuance metody mohou způsobit hrubý nesoulad !
  - Neopakuje se některá proměnná vícekrát?

Čištění dat: Průzkum proměnných

- Nabývá přípustných hodnot (x out of range)?
- „Divné“ kódy („xxx“, „9999“…)
- Duplicitní kódy pro stejnou věc („Ž“, „ž“, „žena“, „zena“…)
- Kódování češtiny/ruštiny/…
- Překlepy apod.
  - Editovací distance (Levenshteinova (Владимир Иосифович Левенштейн), …) pomohou odhalit překlep
  - Editovací distance = počet elementárních editovacích kroků potřebných pro změnu jednoho řetězce na druhý. Viz http://www.merriampark.com/ld.htm k Levenshteinově distanci
    - Je zde aplet, který ji umí počítat
  - Shlukování řetězců podle ED

Čištění dat: Průzkum proměnných

- Slučování podobných kategorií (prodavač – prodejce – prodavačka);
- Málo četné kategorie (národnost brazilská…) – je třeba sloučit/přiřadit k nějaké(kým) více četné(ným) kategorii(ím) na základě nějakého vhodného kriteria.
- Je distribuce přiměřená našemu očekávání (interval hodnot, rozptyl, šikmost, špičatost, modální hodnoty…)? Není např. příliš „ořezaná“ či naopak „roztažená“?
  - Někdy se obtížně poznává: Např. věk v části dat může být kódován jako poslední dvojčíslí roku narození, a v jiné části dat jako *2007 – rok narození*.

Čištění dat: Průzkum proměnných

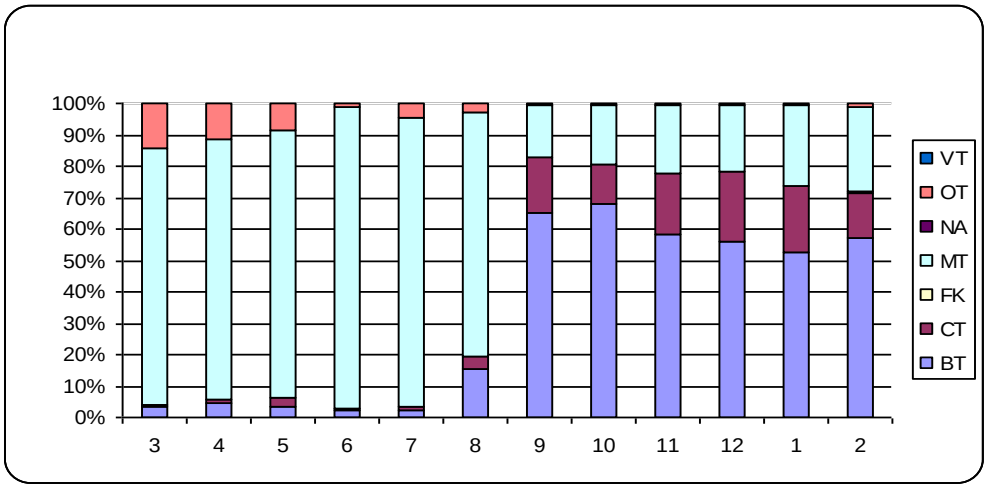
- Shluky (clumping), typicky kolem zaokrouhlených hodnot
  - Příjem –lidé rádi zaokrouhlují směrem nahoru.
  - Nebo třeba kolem hranic věkových kvót, vzniklé tím, jak tazatelé „upravují“ věky respondentů, aby se vešli do kvót.
- Chybějící hodnoty (příčiny vzniku, zastoupení,…)!!!
- Pozor na kódy časů (amer. x evrop. konvence), regionů apod.!

Čištění dat: Vazby mezi daty

- Více proměnných
  - Kontingenční tabulky, box ploty s kategoriemi, bodové grafy a jejich matice, korelační koeficienty
  - Logické vazby (např. 10tiletý nemůže být ženatý, 30tiletý nemůže pracovat 20let,…)
    - Hledání pomocí programu/kódu – podmínky vyjádříme pomocí prostředků matematické logiky a necháme počítač, aby vyhledal případy, kde nejsou splněny.
  - Extrémní hodnoty vícerozměrného rozdělení
    - Bodový graf
    - Mahalanobisova vzdálenost od těžiště: $[(\boldsymbol{x}-\boldsymbol{t})^T \boldsymbol{S}^{-1} (\boldsymbol{x}-\boldsymbol{t})]^{-1/2}$, kde $\boldsymbol{t}$ je vektor těžiště, $\boldsymbol{x}$ zkoumaný bod a $\boldsymbol{S}$ kovarianční matice
      - např. P. Filzmoser (2004) A multivariate outlier detection method, http://www.statistik.tuwien.ac.at/public/filz/papers/minsko4.pdf
  - Další vlastnosti; např. existují očekávané korelace?

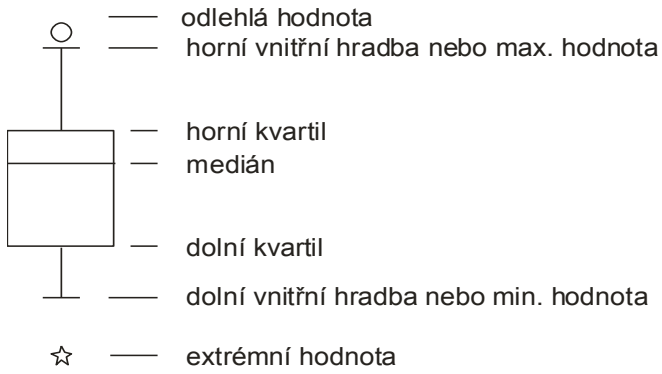
Čištění dat: Vazby mezi daty

- korektní vkládání dat do DB
  - text. pole s názvem zboží vs. rolovací seznam s typem zboží



  - pořadí hodnot v rolovacím seznamu – problém první (defaultní) hodnoty

Čištění dat: Odlehlé hodnoty



- kvartilová odchylka: $q = x_{0.75} - x_{0.25}$
- vnitřní hradby: $x_{0.25} - 1.5q$, $x_{0.75} + 1.5q$
- vnější hradby: $x_{0,25} - 3q$, $x_{0,75} + 3q$

- **Odlehlá hodnota** leží mezi vnějšími a vnitřními hradbami, tj. v intervalu $(x_{0,75} + 1{,}5q, x_{0,75} + 3q)$ či v intervalu $(x_{0,25} - 3q, x_{0,25} - 1{,}5q)$.
- **Extrémní hodnota** leží za vnějšími hradbami, tj. v intervalu $(x_{0,75} + 3q, \infty)$
- či v intervalu $(-\infty, x_{0,25} - 3q)$.

Čištění dat: Opravy chyb

- Zpět k pramenům!
- Vyřazení podezřelých případů:
  - Záměrné podvody, např. nespolehliví tazatelé (shluková analýza!).
  - Neověřitelná data.
- Vyřazení podezřelých hodnot.
- Rekódování na správné hodnoty (imputace hodnot):
  - imputace – průměrem, mediánem, max./min. hodnotou, pomocí modelu.

Transformace dat

- Binarizace (dummy proměnné)
  - Dummy proměnné představují techniku využívající dichotomické proměnné (kódované 0 nebo 1) pro vyjádření jednotlivých hodnot nominálních proměnných.
  - Název „dummy“ poukazuje na fakt, že přítomnost znaku označeného kódem 1 reprezentuje faktor, nebo soubor faktorů, který není měřitelný žádným lepším způsobem v rámci dané analýzy.

Dummy proměnné

- Dummy proměnná přiřazuje hodnotu 1 danému pozorování vybrané proměnné a hodnotu 0 ve zbývajících případech.
- Pro pohlaví (2 kategorie), např. přiřadí 1 pro ženu a 0 pro muže. V tomto případě je postačující vytvoření právě jedné dummy proměnné.
- Pro rasu (4 kategorie), je třeba vytvořit více dummy proměnných.

  P1=1, pokud rasa=„běloch“ a 0 jinak.
  P2=1, pokud rasa=„černoch“ a 0 jinak.
  P3=1, pokud rasa=„asiat“ a 0 jinak.
  P4=1, pokud rasa=„ostatní“ a 0 jinak.
- Důležité: Všechny 4 proměnné nejsou zahrnuty do regrese (způsobilo by to perfektní multikolinearitu, P4=1-P3-P2-P1).
- Počet dummy proměnných=počet kategorií -1.
- Vynechaná proměnná je „referenční“ proměnnou.
- Konstanta obsahuje informaci o této referenční proměnné.
- Koeficienty zahrnutých proměnných jsou brány ve vztahu ke konstantě.

Transformace dat

- Kategorizace spojitých proměnných
  - pokud chceme využít WOE, je nutná
  - decily, ale lze tvořit i tak aby byla maximalizována např. informační hodnota

- Agregace
  - podobných hodnot (věcná podobnost, podobnost hodnot ve vztahu k cílové proměnné,...)
  - málo četných hodnot

- Segmentace
  - není nezbytná, ale může výrazně zvýšit predikční sílu modelu, případně je vhodná z něj. business důvodů.

Categorization of predictors

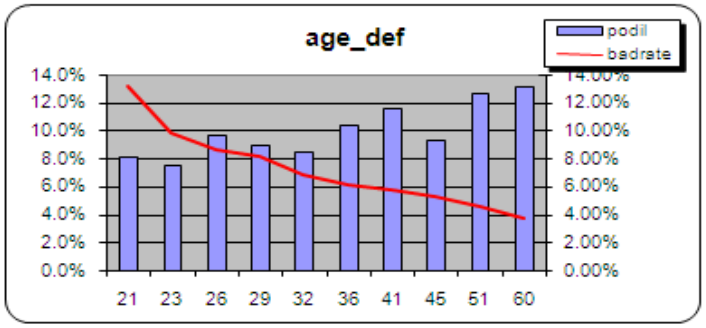
- Every variable should be categorized (divided to reasonable number of categories

  - Best separation (default rates within categories are different as much as possible)

  - Time stability (ordering in categories by default rate is the same in different periods of development sample)

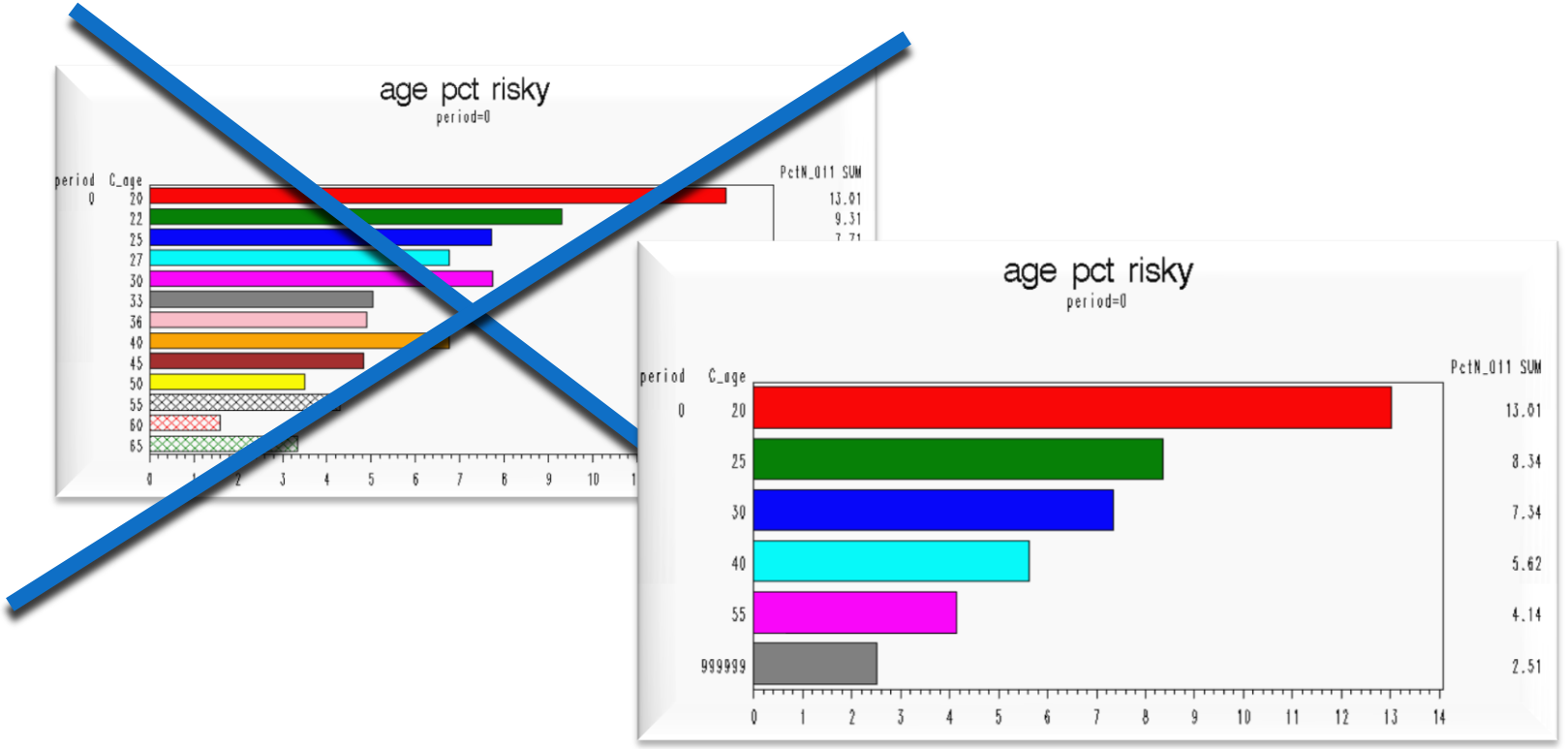
age_def	Gini:	0.2212	Info.Value:	0.1558

	pocet	podil	badrate
21	35 059	8.2%	13.11%
23	32 401	7.5%	9.81%
26	41 807	9.7%	8.61%
29	38 510	9.0%	8.07%
32	36 271	8.4%	6.79%
36	44 648	10.4%	6.11%
41	50 015	11.6%	5.74%
45	40 099	9.3%	5.21%
51	54 526	12.7%	4.52%
60	56 551	13.2%	3.71%
Total	429 887	100.0%	6.79%



Categorization of predictors

- We want to find out real statistical dependencies, not random differences in default.



Transformace dat - WOE

- **Good** celkový počet dobrých klientů ve vzorku
- **Bad** celkový počet špatných klientů ve vzorku

- $\mathbf{good_i^s}$, $\mathbf{bad_i^s}$ počet dobrých, resp. špatných klientů v *i*-té kategorii příslušné *s-té* proměnné.
- celková šance

$$odds\_all = \frac{good}{bad}$$

- šance i-té kategorie s-té proměnné

$$odds_i^{\ s} = \frac{good_i^{\ s}}{bad_i^{\ s}}$$

- poměr šancí (OR)

$$odds\_ratio_i^{\ s} = \frac{odds_i^{\ s}}{odds\_all}$$

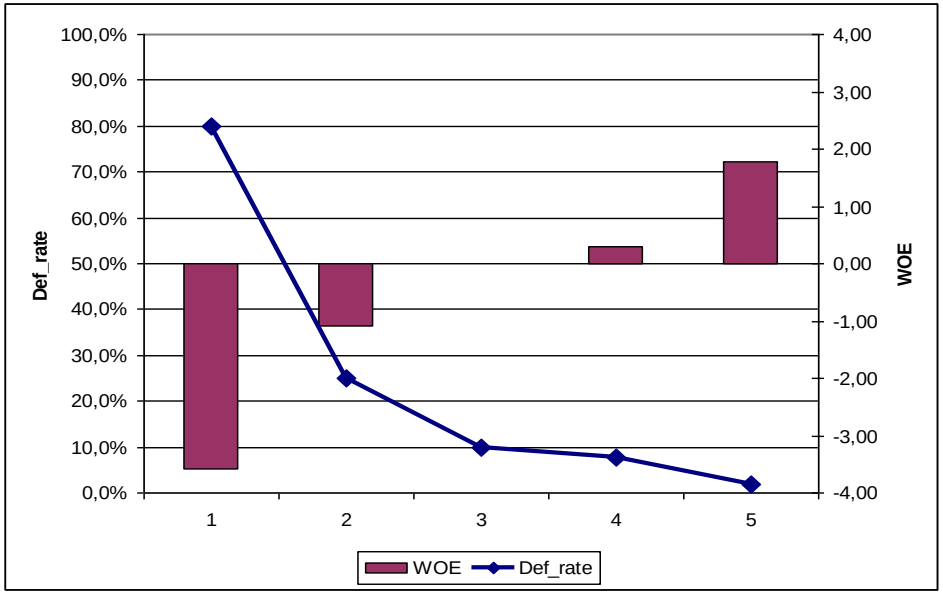
- WOE (weights of evidence)

$$WOE_i^{\ s} = \ln\left(odds\_ratio_i^{\ s}\right) = \ln\left(\frac{\frac{good_i^{\ s}}{bad_i^{\ s}}}{\frac{good}{bad}}\right) = \ln\left(\frac{\frac{good_i^{\ s}}{good}}{\frac{bad_i^{\ s}}{bad}}\right)$$

Transformace dat -WOE

cat.	# bad clients	#good clients	Def_rate	odds	OR	% bad [1]	% good [2]	[3] = [2] / [1]	WOE = ln[3]
1	4	1	80,0%	0,25	0,03	40,0%	1,1%	0,03	-3,58
2	2	6	25,0%	3,00	0,33	20,0%	6,7%	0,33	-1,10
3	2	18	10,0%	9,00	1,00	20,0%	20,0%	1,00	0,00
4	1	12	7,7%	12,00	1,33	10,0%	13,3%	1,33	0,29
5	1	53	1,9%	53,00	5,89	10,0%	58,9%	5,89	1,77
All	10	90	10,0%	9,00					
ALL	100								

80% = 4/ (4+1)
0,25 = 1/4
0,03 = 0,25 / 9
40% = 4 / 10
1,1% = 1 / 90



Citlivost WOE

1. Zachování Def_rate, ale zmenšení četnosti 1. kategorie (změna %all)

2. Změna Def_rate při zachování %all.

3. Jako u 2, navíc změna def_rate u další kategorie.

cat.	# bad clients	#good clients	Def_rate	odds	OR	% all	% bad [1]	% good [2]	[3] = [2] / [1]	WOE = ln[3]
1	400	100	80,0%	0,25	0,03	5,0%	40,0%	1,1%	0,03	-3,58
2	200	600	25,0%	3,00	0,33	8,0%	20,0%	6,7%	0,33	-1,10
3	200	1800	10,0%	9,00	1,00	20,0%	20,0%	20,0%	1,00	0,00
4	100	1200	7,7%	12,00	1,33	13,0%	10,0%	13,3%	1,33	0,29
5	100	5300	1,9%	53,00	5,89	54,0%	10,0%	58,9%	5,89	1,77
All	1000	9000	10,0%	9,00						
ALL	10000									

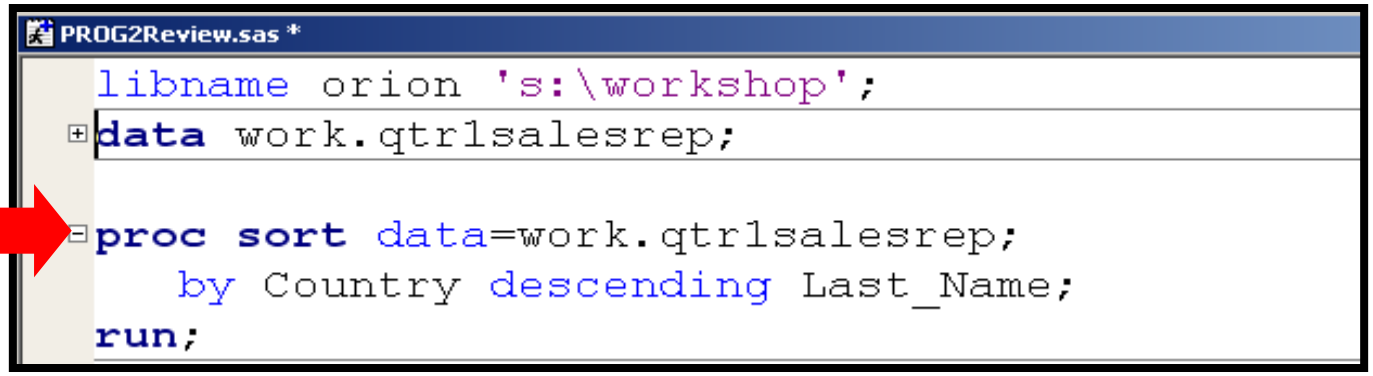
cat.	# bad clients	#good clients	Def_rate	odds	OR	% all	% bad [1]	% good [2]	[3] = [2] / [1]	WOE = ln[3]
1	200	50	80,0%	0,25	0,03	2,6%	25,0%	0,6%	0,02	-3,80
2	200	600	25,0%	3,00	0,33	8,2%	25,0%	6,7%	0,27	-1,32
3	200	1800	10,0%	9,00	1,00	20,5%	25,0%	20,1%	0,80	-0,22
4	100	1200	7,7%	12,00	1,33	13,3%	12,5%	13,4%	1,07	0,07
5	100	5300	1,9%	53,00	5,89	55,4%	12,5%	59,2%	4,74	1,56
All	800	8950	8,2%	11,19						
ALL	9750									

cat.	# bad clients	#good clients	Def_rate	odds	OR	% all	% bad [1]	% good [2]	[3] = [2] / [1]	WOE = ln[3]
1	300	200	60,0%	0,67	0,07	5,0%	33,3%	2,2%	0,07	-2,72
2	200	600	25,0%	3,00	0,33	8,0%	22,2%	6,6%	0,30	-1,22
3	200	1800	10,0%	9,00	1,00	20,0%	22,2%	19,8%	0,89	-0,12
4	100	1200	7,7%	12,00	1,33	13,0%	11,1%	13,2%	1,19	0,17
5	100	5300	1,9%	53,00	5,89	54,0%	11,1%	58,2%	5,24	1,66
All	900	9100	9,0%	10,11						
ALL	10000									

cat.	# bad clients	#good clients	Def_rate	odds	OR	% all	% bad [1]	% good [2]	[3] = [2] / [1]	WOE = ln[3]
1	300	200	60,0%	0,67	0,07	5,0%	30,0%	2,2%	0,07	-2,60
2	300	500	37,5%	1,67	0,19	8,0%	30,0%	5,6%	0,19	-1,69
3	200	1800	10,0%	9,00	1,00	20,0%	20,0%	19,8%	0,99	-0,01
4	100	1200	7,7%	12,00	1,33	13,0%	10,0%	13,2%	1,32	0,28
5	100	5300	1,9%	53,00	5,89	54,0%	10,0%	58,2%	5,82	1,76
All	1000	9000	10,0%	9,00						
ALL	10000									

The SORT Procedure

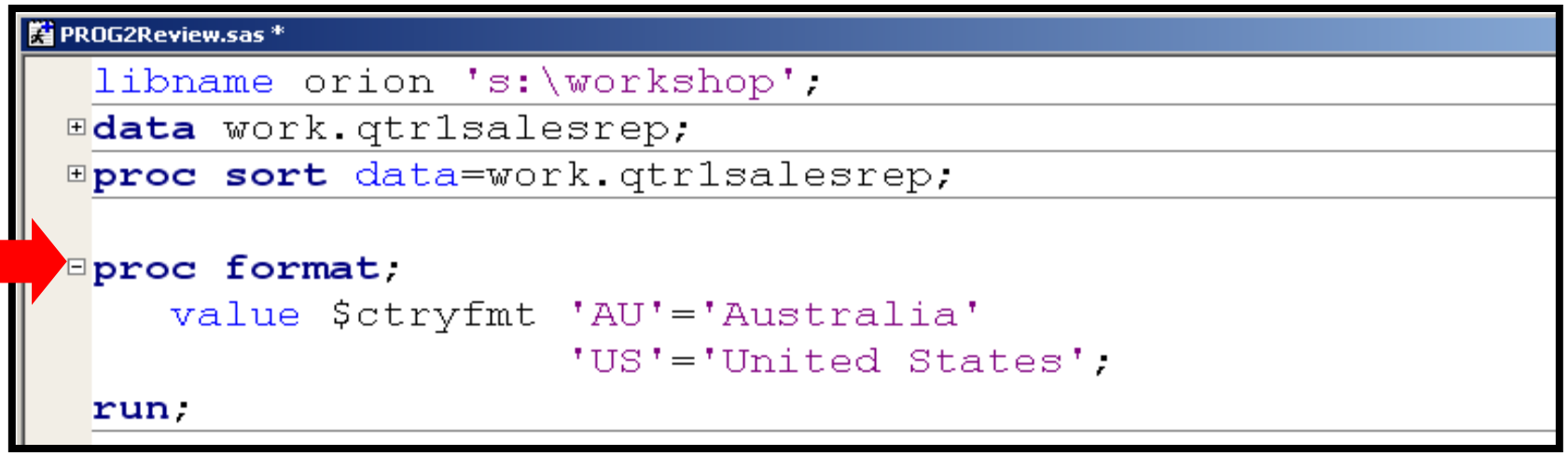
- The SORT procedure rearranges the observations in **work.qtr1salesrep** and places them in order by descending **Last_Name** within **Country**.



- The OUT= option in the SORT procedure can be used to create an output data set, instead of overwriting the input data set.

The FORMAT Procedure

- The FORMAT procedure creates user-defined formats and informats, and stores them in the SAS catalog **work.formats** by default.



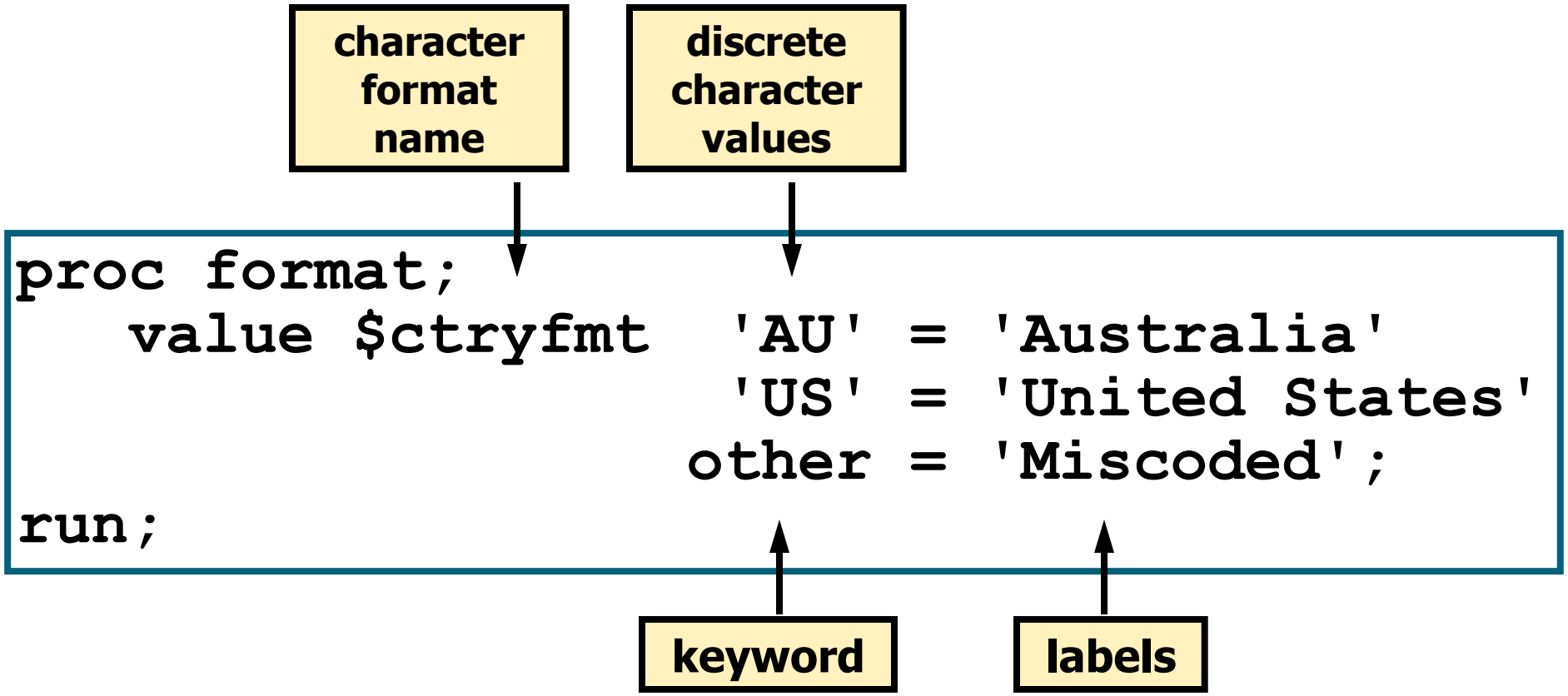
- Více na: http://www2.sas.com/proceedings/sugi27/p056-27.pdf

The FORMAT Procedure

- *Range(s)* can be
  - single values
  - ranges of values
  - lists of values.

- *Labels*
  - can be up to 32,767 characters in length
  - are typically enclosed in quotation marks, although it is not required.

Character User-Defined Format



- The OTHER keyword matches all values that do not match any other value or range.

Character User-Defined Format

Part 1

```
proc format;
   value $ctryfmt   'AU' = 'Australia'
                    'US' = 'United States'
                  other = 'Miscoded';
run;

proc print data=orion.sales label;
   var Employee_ID Job_Title Salary
       Country Birth_Date Hire_Date;
   label Employee_ID='Sales ID'
         Job_Title='Job Title'
         Salary='Annual Salary'
         Birth_Date='Date of Birth'
         Hire_Date='Date of Hire';
```

Part 2

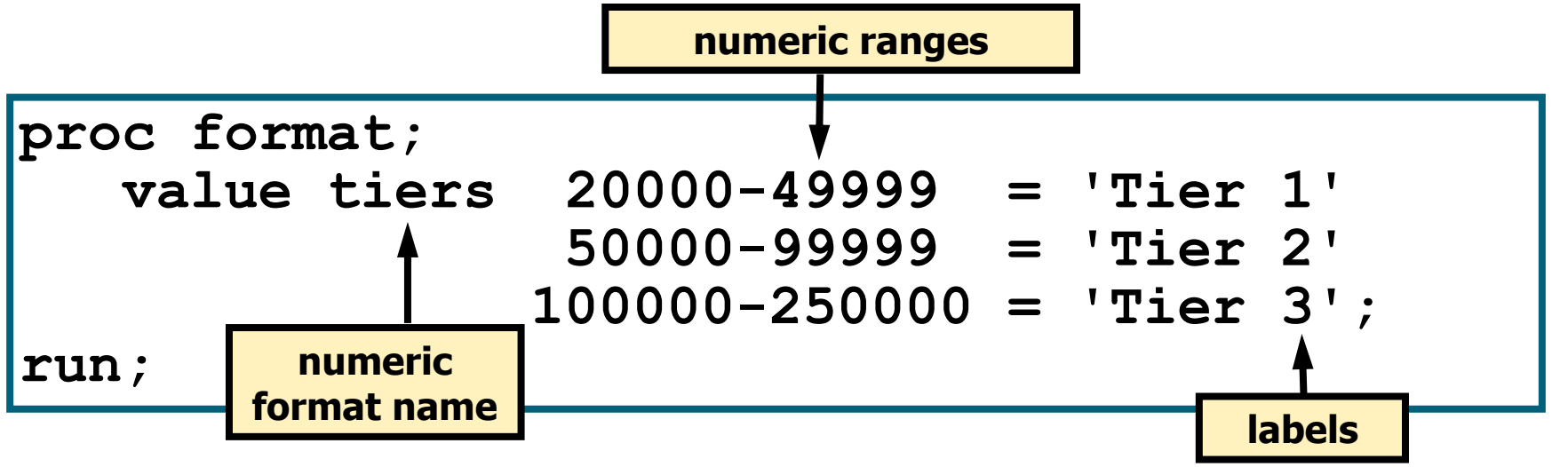
```
   format Salary dollar10.0
          Birth_Date Hire_Date monyy7.
          Country $ctryfmt.;
run;
```

Character User-Defined Format

- Partial PROC PRINT Output

Obs	Sales ID	Job Title	Annual Salary	Country	Date of Birth	Date of Hire
60	120178	Sales Rep. II	$26,165	Australia	NOV1954	APR1974
61	120179	Sales Rep. III	$28,510	Australia	MAR1974	JAN2004
62	120180	Sales Rep. II	$26,970	Australia	JUN1954	DEC1978
63	120198	Sales Rep. III	$28,025	Australia	JAN1988	DEC2006
64	120261	Chief Sales Officer	$243,190	United States	FEB1969	AUG1987
65	121018	Sales Rep. II	$27,560	United States	JAN1944	JAN1974
66	121019	Sales Rep. IV	$31,320	United States	JUN1986	JUN2004
67	121020	Sales Rep. IV	$31,750	United States	FEB1984	MAY2002
68	121021	Sales Rep. IV	$32,985	United States	DEC1974	MAR1994
69	121022	Sales Rep. IV	$32,210	United States	OCT1979	FEB2002
70	121023	Sales Rep. I	$26,010	United States	MAR1964	MAY1989
71	121024	Sales Rep. II	$26,600	United States	SEP1984	MAY2004
72	121025	Sales Rep. II	$28,295	United States	OCT1949	SEP1975

Numeric User-Defined Format




```
proc format;
   value tiers    low-<50000  = 'Tier 1'
                 50000- 100000 = 'Tier 2'
                100000<-high   = 'Tier 3';
run;
```

LOW encompasses the lowest possible value.
HIGH encompasses the highest possible value.

Numeric User-Defined Formats

- The less than (<) symbol excludes values from ranges.
  - Put < after the value if you want to exclude the first value in a range.
  - Put < before the value if you want to exclude the last value in a range.

50000 - 100000	Includes 50000	Includes 100000
50000 - < 100000	Includes 50000	Excludes 100000
50000 < - 100000	Excludes 50000	Includes 100000
50000 < - < 100000	Excludes 50000	Excludes 100000

Multiple User-Defined Formats

- Multiple VALUE statements can be in a single PROC FORMAT step.

```
proc format;
   value $ctryfmt   'AU' = 'Australia'
                    'US' = 'United States'
                  other = 'Miscoded';
   value tiers      low-<50000  = 'Tier 1'
                 50000- 100000 = 'Tier 2'
                100000<-high   = 'Tier 3';
run;
```

The FORMAT Procedure

```
proc format;
value $goods_t
'BT'='A'
'BZ'='D,
''='missing'
'  '='missing'
'.'='missing'
;
run;
```

```
proc tabulate data=lib1.tab1
missing;
title "D vs. goods_type";
class goods_type D;
table (goods_type all),(D
all)*(n colpctn='c%'
rowpctn='r%');
format goods_type $goods_t.;
run;
```

The FORMAT Procedure

```
proc format;
value good_typ
1=1
2=3
3=10
;
run;
```

```
data lib1.tab1;
set lib1.tab1;
  goods_type3=goods_type2;
  format goods_typen3n good_typ.;
run;
```

```
proc format;
invalue good_t2e
'BT'=4
'BZ'=5
'CK'=5
other=-1
;
run;
```

```
data lib1.tab1;
set lib1.tab1;
goods_type1=upcase(goods_type);
goods_type3n=input(goods_type1,good_t2e.)
;
evid_id=put(evid_id,z10.);
;
run;
```

Replacing Missing Values

The COALESCE function enables you to replace missing values in a column with a new value that you specify. For every row that the query processes, the COALESCE function checks each of its arguments until it finds a nonmissing value, then returns that value. If all of the arguments are missing values, then the COALESCE function returns a missing value. For example, the following query replaces missing values in the LowPoint column in the SQL.CONTINENTS table with the words **Not Available:**

```
proc sql;
title 'Continental Low
Points';
select Name,
coalesce(LowPoint,
'Not Available') as
LowPoint
from sql.continents;
```

Output 2.14 Using the COALESCE Function to Replace Missing Values

Continental Low Points

Name	LowPoint
Africa	Lake Assal
Antarctica	Not Available
Asia	Dead Sea
Australia	Lake Eyre
Central America and Caribbean	Not Available
Europe	Caspian Sea
North America	Death Valley
Oceania	Not Available
South America	Valdes Peninsula

The DATA Step

- The SAS DATA step
  - is the original SAS programming language for data manipulation
  - can be used as a complete **programming language**
  - is generated by SAS Enterprise Guide when data is imported or in support of other tasks.

Advantages of the DATA Step over SQL

DATA Step	SQL
Can read data from many different sources	Can only read from SAS database tables
Can create multiple tables in a single pass of the data	Can only output one table at a time
Has comprehensive conditional processing	Only has the CASE clause
Can deal with repetitive programming using loops and arrays	Does not support loops or arrays

Advantages of SQL over the DATA Step

SQL	DATA Step
Is very flexible when joining multiple tables with non-common key variables	Can require several steps to join multiple tables with different key variables
Can, in some cases, replace multiple SAS steps	Can require several steps
Is the native language of databases	Might need to generate SQL to get to data that is not SAS data

Choose the right tool for the task to be completed.

The DATA Statement

- The *DATA statement* begins a DATA step and provides the name of the SAS data set being created.
- General form of the DATA statement:

```
DATA output-SAS-data-set;
   SET input-SAS-data-set;
<additional SAS statements>
RUN;
```

- The DATA statement can create temporary or permanent data sets.

The SET Statement

- The *SET statement* reads observations from a SAS data set for further processing in the DATA step.
- General form of the SET statement:

```
DATA output-SAS-data-set;
   SET input-SAS-data-set;
<additional SAS statements>
RUN;
```

- By default, the SET statement does the following:
  - names the SAS data set(s) to be read
  - reads all observations and all variables from the input data set
  - can read temporary or permanent data sets

Business Scenario: Reading a SAS Data Set

This program does the following:

- reads all the rows and all the columns from the **sales** data set in the **orion** library
- writes all the rows and all the columns to a data set named **comp** in the Work library

```
data work.comp;
   set orion.sales;
run;
```

Partial Listing of **comp**

Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title	Country	Birth_Date	Hire_Date
120102	Tom	Zhou	M	108255	Sales Manager	AU	3510	10744
120103	Wilson	Dawes	M	87975	Sales Manager	AU	-3996	5114
120121	Irenie	Elvish	F	26600	Sales Rep. II	AU	-5630	5114
120122	Christina	Ngan	F	27475	Sales Rep. II	AU	-1984	6756
120123	Kimiko	Hotstone	F	26190	Sales Rep. I	AU	1732	9405
120124	Lucian	Daymond	M	26480	Sales Rep. I	AU	-233	6999

Selecting Variables

- You can control the variables **written out** to SAS data sets using the following:
  - the DROP statement to specify the variables that you want **excluded**
  - the KEEP statement to specify the variables that you want **included**
- General form of DROP and KEEP statements:

DROP *variable1 variable2* …**;**

KEEP *variable1 variable2* …**;**

Business Scenario: Selecting Variables

```
data work.comp;
   set orion.sales;
   drop Gender Salary Job_Title
        Country Birth_Date Hire_Date;
run;
```

This program can do these tasks:

- read all the rows and columns from **orion.sales**
- write all the rows and the three columns not excluded via the DROP statement to a data set called **comp** in the Work library

Partial Listing of **comp**

Employee_ID	First_Name	Last_Name
120102	Tom	Zhou
120103	Wilson	Dawes
120121	Irenie	Elvish
120122	Christina	Ngan
120123	Kimiko	Hotstone
120124	Lucian	Daymond

Selecting Rows

Partial Listing of `austemp`

	Employee_ID	First_Name	Last_Name	Gender	Salary	Job_Title	Country
1	120102	Tom	Zhou	M	108255	Sales Manager	AU
2	120103	Wilson	Dawes	M	87975	Sales Manager	AU
3	120125	Fong	Hofmeister	M	32040	Sales Rep. IV	AU
4	120128	Monica	Kletschkus	F	30890	Sales Rep. IV	AU
5	120129	Alvin	Roebuck	M	30070	Sales Rep. III	AU
6	120135	Alexei	Platts	M	32490	Sales Rep. IV	AU
7	120144	Viney	Barbis	M	30265	Sales Rep. III	AU
8	120154	Caterina	Hayawardhana	F	30490	Sales Rep. III	AU
9	120158	Daniel	Pilgrim	M	36605	Sales Rep. III	AU
10	120159	Lynelle	Phoumirath	F	30765	Sales Rep. IV	AU
11	120161	Rosette	Martines	F	30785	Sales Rep. III	AU
12	120166	Fadi	Nowd	M	30660	Sales Rep. IV	AU

Orion wants to subset the data to only include Australian employees with a salary greater than $30,000.

Selecting Rows with the WHERE Statement

You can control which rows are read from a SAS data set by using the WHERE statement.

General form of the WHERE statement:

WHERE *expression*;

- Only one WHERE statement can be included in a DATA step.
- The expressions that can be used are the same as expressions built in the Filter Data tab using either the Edit Filter window or the Advanced Expression Editor.

Comparison Operators -examples

```
where Gender = 'M';
```

```
where Gender eq ' ';
```

```
where Salary ne .;
```

```
where Salary >= 50000;
```

```
where Country in ('AU','US');
```

```
where Country in ('AU' 'US');
```

Values must be separated by commas or blanks.

Arithmetic Operators - examples

```
where Salary / 12 < 6000;
```

```
where (Salary / 12 ) * 1.10 >= 7500;
```

```
where Salary + Bonus <= 10000;
```

Logical Operators - examples

```
where Gender ne 'M' and Salary >=50000;
```

```
where Gender ne 'M' or Salary >= 50000;
```

```
where Country = 'AU' or Country = 'US';
```

```
where Country not in ('AU' 'US');
```

Multiple Choice Poll – Correct Answer

- Which WHERE statement correctly subsets for numeric months May, June, or July and character names with a missing value?

a. `where Months in (5 - 7) and Names = . ;`

b. `where Months in (5 , 6 , 7) and Names = ' ';`

c. `where Months in ('5' , '6' , '7') and Names = '.';`

Creating New Variables

- Assignment statements are used in the DATA step to update existing variables or create new variables.
- An assignment statement does the following:
  - evaluates an expression
  - assigns the resulting value to a variable

General form of an assignment statement:

```
variable=expression;
```

```
DATA output-SAS-data-set;
     SET input-SAS-data-set;
     variable = expression;
RUN;
```

SAS Expressions

- An *expression* contains *operands* and *operators* that form a set of instructions that produce a value.

Operands are

- variable names
- constants.

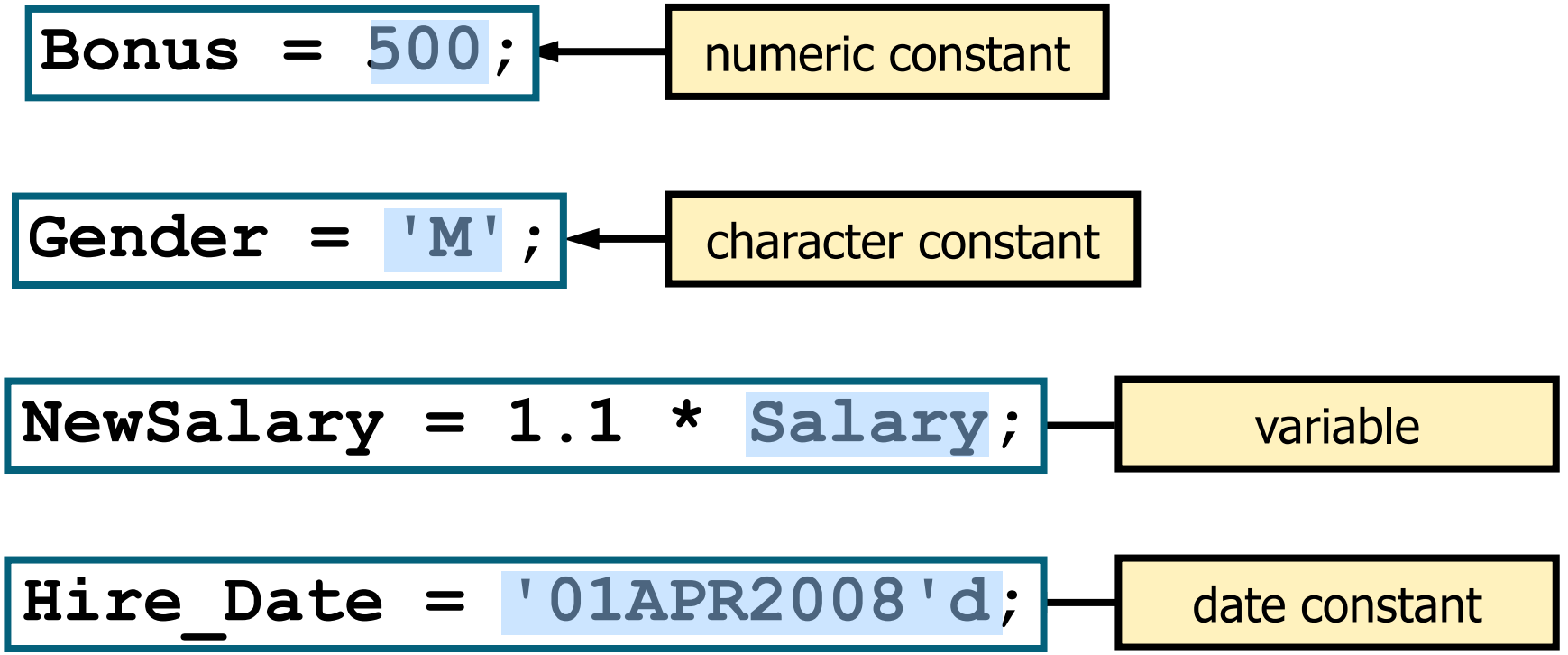
Operators are

- symbols that request arithmetic calculations
- SAS functions.

- *An expression* entered in an assignment statement is identical to an expression built using the SAS Enterprise Guide Advanced Expression Editor.

Operands

- Operands are constants (character, numeric, or date) and variables (character or numeric).
- Examples:



SAS Date Constants

The constant '***ddMMMyyyy***'**d** (example: '14dec2000'd) creates a SAS date value from the date enclosed in quotation marks.

dd	is a one- or two-digit value for the day.
MMM	is a three-letter abbreviation for the month (JAN, FEB, MAR, and so on).
yyyy	is a four-digit value for the year.
d	is required to convert the quoted string to a SAS date.

Operators

- Operators are symbols that represent an arithmetic calculation and SAS functions.

- Examples:

```
Revenue = Quantity * Price;
```

```
NewCountry = upcase(Country);
```

Arithmetic Operators

- *Arithmetic operators* indicate that an arithmetic calculation is performed.

Symbol	Definition	Priority
**	exponentiation	I
-	negative prefix	I
*	multiplication	II
/	division	II
+	addition	III
-	subtraction	III

- If a missing value is an operand for an arithmetic operator, the result is a missing value.

Multiple Choice Poll – Correct Answer

- What is the result of the assignment statement given the values of **`var1`** and **`var2`**?

a. . (missing)
b. 0
c. 5
d. 10

var1	var2
.	10

```
num = var1 + var2 / 2;
```

If an operand is missing for an arithmetic operator, the result is missing.

Using SAS Functions

- SAS functions can do the following:
  - perform arithmetic operations
  - compute sample statistics (for example: sum, mean, and standard deviation)
  - manipulate SAS dates
  - process character values
  - perform many other tasks

Sample statistics functions ignore missing values.

- SAS functions can be used in the DATA step or in the Advanced Expression Editor of the Query Builder to create new columns or filter data.

Multiple Choice Poll – Correct Answer

- What is the result of the assignment statement given the values of **`var1`**, **`var2`**, and **`var3`**?

a. . (missing)

b. 0

c. 4

d. 6

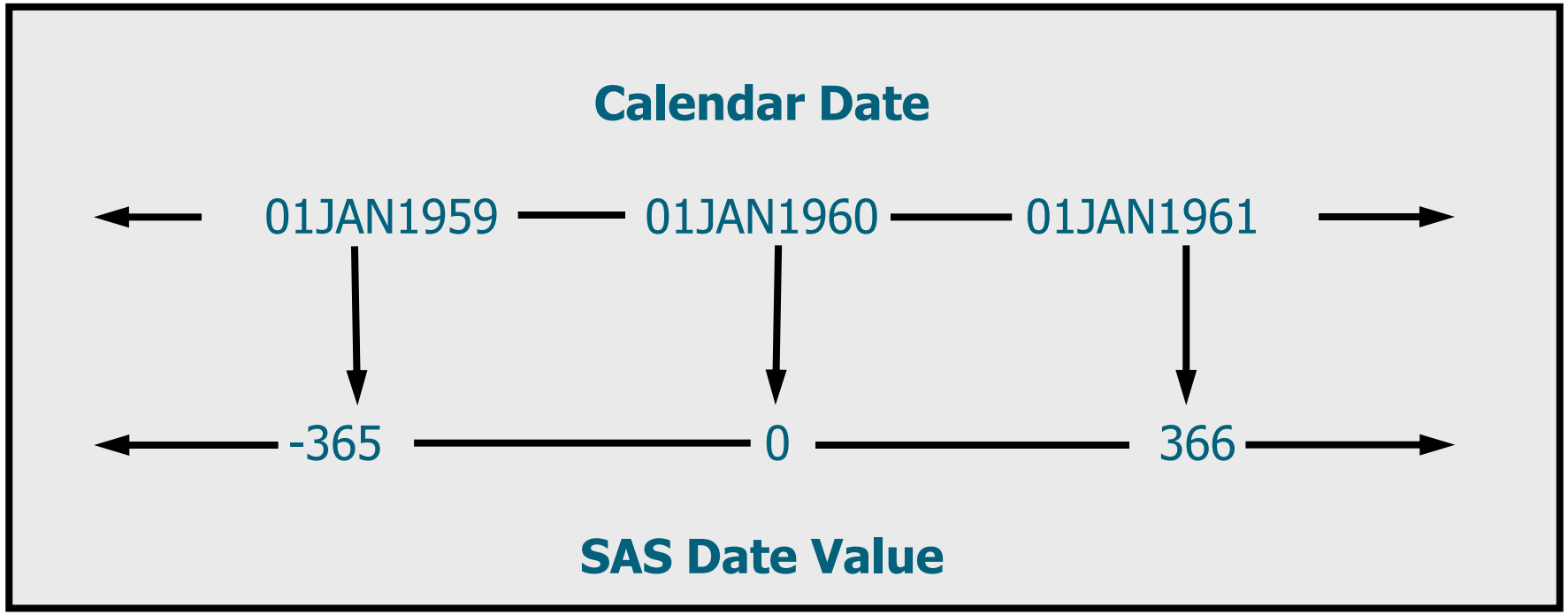
Var1	Var2	Var3
9	.	3

```
Average = mean(Var1,Var2,Var3);
```

Using Date Functions

You can use SAS date functions to do the following:

- create SAS date values
- extract information from SAS date values



Date Functions: Creating SAS Dates

TODAY()	obtains the date value from the system clock.
MDY(*month,day,year*)	uses numeric *month*, *day*, and *year* values to return the corresponding SAS date value.

- Example:

```
Days_Since_Order = today() - Order_Date;
```

Date Functions: Extracting Information

YEAR(*SAS-date*)	extracts the year from a SAS date and returns a four-digit value for year.
QTR(*SAS-date*)	extracts the quarter from a SAS date and returns a number from 1 to 4.
MONTH(*SAS-date*)	extracts the month from a SAS date and returns a number from 1 to 12.
DAY(*SAS-date*)	extracts the day of the month from a SAS date and returns a number from 1 to 31.
WEEKDAY(*SAS-date*)	extracts the day of the week from a SAS date and returns a number from 1 to 7, where 1 represents Sunday, and so on.

- Example:

```
BonusMonth = month(Hire_Date);
```

The LABEL Statement

- Permanent labels can also be assigned in the DATA step.
- General form of the LABEL statement:

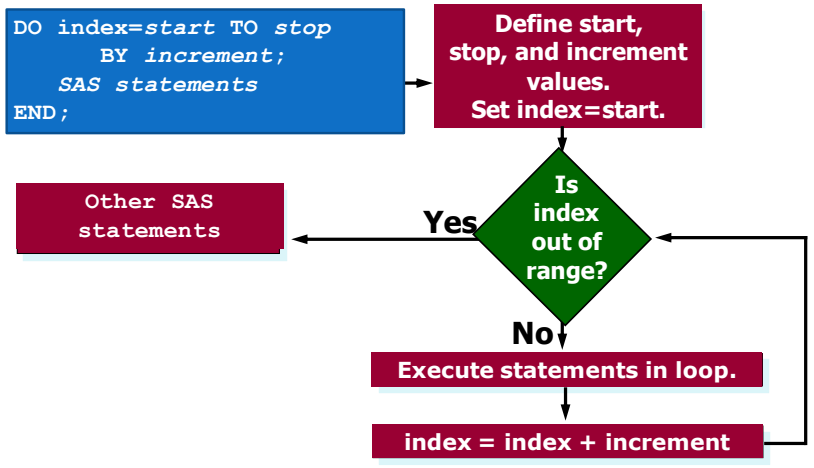
```
LABEL variable = 'label'
      variable = 'label'
      variable = 'label';
```

  - A label can be up to 256 characters.
  - Any number of variables can be associated with labels in a single LABEL statement.
  - Using a LABEL statement in a DATA step permanently associates labels with variables by storing the label in the descriptor portion of the SAS data set.

Business Scenario: Formats and Labels

```
data work.comp;
   set orion.sales;
   Bonus=500;
   Compensation=sum(Salary,Bonus);
   BonusMonth=month(Hire_Date);
   drop Gender Salary Job_Title Country
        Birth_Date;
   format Bonus Compensation dollar8.
          Hire_Date date9.;
   label Employee_ID="Employee ID"
         First_Name="First Name"
         Last_Name="Last Name"
         BonusMonth="Month of Bonus"
         Hire_Date="Hire Date";
run;
```

5. SAS Data Step – podmíněné kódy, cykly, pole



IF *expression* **THEN DO;**
executable statements
END;
ELSE DO;
executable statements
END;

Business Scenario

Customer_Type_ID indicates the type of club member. A value of **3010** indicates non-club members.

Orion wants to create two data sets, one for club members and one for other customers.

Listing of **customer**

Customer_ID	Customer_Name	Customer_Type_ID
4	James Kvarniq	1020
5	Sandrina Stephano	2020
9	Cornelia Krahl	2020
10	Karen Ballinger	1040
11	Elke Wallstab	1040
12	David Black	1030
13	Markus Sepke	2010
16	Ulrich Heyde	3010
17	Jimmie Evans	1030

Listing of **clubmembers**

Customer_ID	Customer_Name	Customer_Type_ID
4	James Kvarniq	1020
5	Sandrina Stephano	2020
9	Cornelia Krahl	2020
10	Karen Ballinger	1040
11	Elke Wallstab	1040

Listing of **nonclub**

Customer_ID	Customer_Name	Customer_Type_ID
16	Ulrich Heyde	3010
23	Tulio Devereaux	3010
24	Robyn Klem	3010
27	Cynthia Mccluney	3010
29	Candy Kinsey	3010

The DATA Statement

The DATA statement lists the data sets to be created.

General form of the DATA statement:

DATA <*SAS-data-set(s)*> <*SAS-data-set(s)*>;

By default, the same rows are written to every listed data set.

```
data work.clubmembers work.nonclub;
   set orion.customer;
run;
```

The OUTPUT Statement

- The OUTPUT statement controls when the values are written to the output SAS data set.

General form of the OUTPUT statement:

OUTPUT <*SAS-data-set(s)*>**;**

- If no data set is specified, then output goes to all of the data sets listed in the DATA statement.

Conditional Execution

General form of IF-THEN and ELSE statements:

IF *expression* **THEN** *statement*;
ELSE *statement*;

An *expression* contains operands and operators that form a set of instructions that produce a value.

Operands are
- variable names
- constants.

Operators are
- symbols that request
  - a comparison
  - a logical operation
  - an arithmetic calculation
- SAS functions.

Only one executable statement is allowed in an IF-THEN or ELSE statement.

Program with Conditional Output

When **Customer_Type_ID** is equal to **3010**, this indicates that the customer is not a club member. Rows where this expression is true are output to the **work.nonclub** SAS data set.

```
data work.clubmembers work.nonclub;
   set orion.customer;
   if Customer_Type_ID = 3010
      then output work.nonclub;
   else output work.clubmembers;
run;
```

ep03d05.sas

Program with Conditional Output

Otherwise, rows are written to the **work.clubmembers** data set.

```
data work.clubmembers work.nonclub;
   set orion.customer;
   if Customer_Type_ID = 3010
      then output work.nonclub;
   else output work.clubmembers;
run;
```

Business Scenario

Orion management wants to add an additional column to the **clubmembers** data set to indicate the membership type. **Customer_Type_ID** numbers between 1000 and 2000 are Members, and between 2000 and 3000 are Gold Members.

Partial Listing of **clubmembers**

	Customer_LastName	Birth_Date	Customer_Address	Street_ID	Street_Number	Customer_Type_ID	Type
1	Kvarniq	27JUN1974	4382 Gralyn Rd	9260106519	4382	1020	Club Member
2	Stephano	09JUL1979	6468 Cog Hill Ct	9260114570	6468	2020	Gold Club Member
3	Krahl	27FEB1974	Kallstadterstr. 9	3940106659	9	2020	Gold Club Member
4	Ballinger	18OCT1984	425 Bryant Estates Dr	9260129395	425	1040	Club Member
5	Wallstab	16AUG1974	Carl-Zeiss-Str. 15	3940108592	15	1040	Club Member
6	Black	12APR1969	1068 Haithcock Rd	9260103713	1068	1030	Club Member
7	Sepke	21JUL1988	Iese 1	3940105189	1	2010	Gold Club Member
8	Evans	17AUG1954	391 Greywood Dr	9260123306	391	1030	Club Member
9	Asmussen	02FEB1954	117 Langtree Ln	9260112361	117	1020	Club Member
10	Füßling	23FEB1964	Hechtsheimerstr. 18	3940106547	18	2030	Gold Club Member

Executing Multiple Statements Conditionally

If **Customer_Type_ID** does not equal **3010**, then these three statements should be executed:

```
if Customer_Type_ID < 2000
   then Type="Club Member";
else Type="Gold Club Member";
output clubmembers;
```

However, only one executable statement is allowed after THEN or ELSE.

```
data work.clubmembers work.nonclub;
   set orion.customer;
   if Customer_Type_ID = 3010
      then output work.nonclub;
   else output work.clubmembers;
run;
```

Executing Multiple Statements Conditionally

You can use the **DO** and **END** statements to execute a group of statements based on a condition.

General form of the DO and END statements:

```
IF expression THEN DO;
   executable statements
END;
ELSE DO;
   executable statements
END;
```

Program with Conditional Output

```
data work.clubmembers work.nonclub;
   set orion.customer;
   if Customer_Type_ID = 3010
      then output nonclub;
   else do;
      if Customer_Type_ID < 2000
         then Type="Club Member";
      else Type="Gold Club Member";
      output clubmembers;
   end;
run;
```

ep03d06.sas

Business Scenario: Results

Partial Listing of **clubmembers**

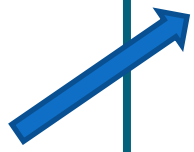
	Customer_LastName	Birth_Date	Customer_Address	Street_ID	Street_Number	Customer_Type_ID	Type
1	Kvarniq	27JUN1974	4382 Gralyn Rd	9260106519	4382	1020	Club Member
2	Stephano	09JUL1979	6468 Cog Hill Ct	9260114570	6468	2020	Gold Club M
3	Krahl	27FEB1974	Kallstadterstr. 9	3940106659	9	2020	Gold Club M
4	Ballinger	18OCT1984	425 Bryant Estates Dr	9260129395	425	1040	Club Member
5	Wallstab	16AUG1974	Carl-Zeiss-Str. 15	3940108592	15	1040	Club Member
6	Black	12APR1969	1068 Haithcock Rd	9260103713	1068	1030	Club Member
7	Sepke	21JUL1988	Iese 1	3940105189	1	2010	Gold Club M
8	Evans	17AUG1954	391 Greywood Dr	9260123306	391	1030	Club Member
9	Asmussen	02FEB1954	117 Langtree Ln	9260112361	117	1020	Club Member
10	Fußling	23FEB1964	Hechtsheimerstr. 18	3940106547	18	2030	Gold Club M

Why are some values of **Type** truncated?

Program with Conditional Output

When a program is compiled, each variable must be defined with a name, type, and length. The length of **Type** is set to 11 based on the first occurrence in the program.

```
data work.clubmembers work.nonclub;
   set orion.customer;
   if Customer_Type_ID = 3010
      then output nonclub;
   else do;
      if Customer_Type_ID < 2000
         then Type="Club Member";
      else Type="Gold Club Member";
       output clubmembers;
   end;
run;
```



Délka je nastavena na **11** znaků.

Conditional Statements

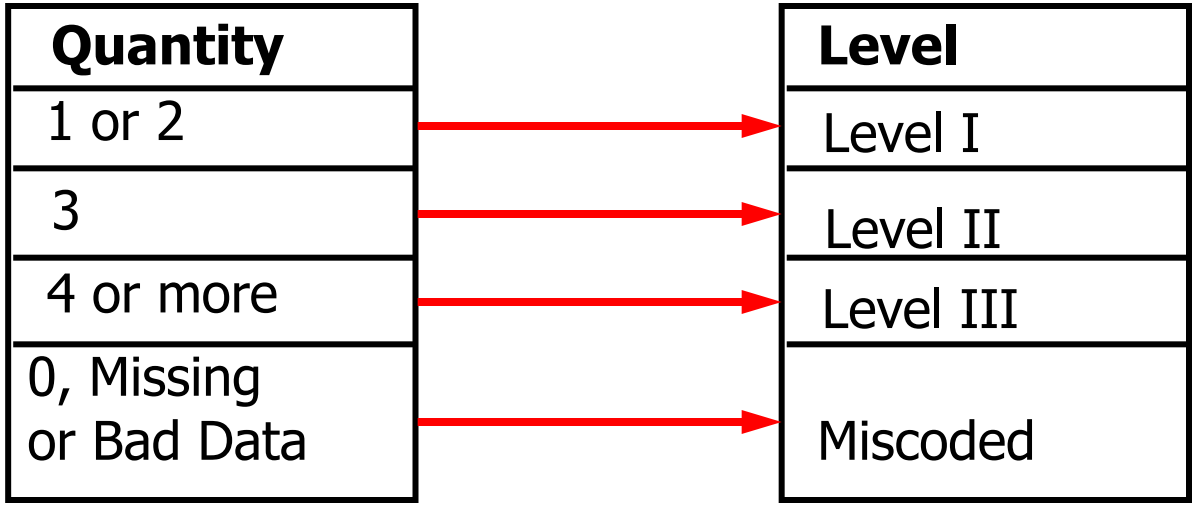
- Conditional statements can create values for a new variable based on whether a condition is true or false.
- General form of the IF-THEN and ELSE statements:

IF *expression* **THEN** *statement*;
ELSE IF *expression* **THEN** *statement*;
. . .

Conditionally Executing Statements

```
data univ.totalorders;
   set univ.customerorders;
   TotalSale=UnitPrice*Quantity;
   if Quantity in (1,2) then Level='Level I';
   else if Quantity=3 then Level='Level II';
   else if Quantity ge 4 then Level='Level III';
   else Level='Miscoded';
run;
```

c03s3d1.sas



Listing Output

- Partial Output

Quantity	Level	Total Sale
1	Level I	$20
3	Level I	$234
1	Level I	$50
5	Level I	$560
1	Level I	$12
2	Level I	$84
2	Level I	$46

Where are the Level II and Level III values?

The LENGTH Statement

- You can use the LENGTH statement to define the length of a variable explicitly.
- General form of the LENGTH statement:

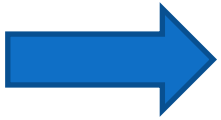
LENGTH *variable(s)* $ *length*;

Example:

```
length Level $ 9;
```

The LENGTH Statement

```
data univ.totalorders;
   set univ.customerorders;
   length Level $ 9;
   TotalSale=UnitPrice*Quantity;
   if Quantity in (1,2)then Level='Level I';
   else if Quantity=3 then Level='Level II';
   else if Quantity ge 4 then Level='Level III';
   else Level='Miscoded';
run;
```



Quantity	Level	Total Sale
1	Level II	$20
3	Level I	$234
1	Level III	$50
5	Level I	$560
1	Level I	$12
2	Level I	$84
2	Level I	$46

First. and Last. Variables

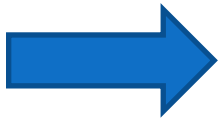
- If you use a by statement along with a set statement in a data step then SAS creates two automatic variables, FIRST.variable and LAST.variable, where variable is the name of the by variable. FIRST.variable has a value 1 for the first observation in the by group and 0 for all other observations in the by group. LAST.variable has a value 1 for the last observation in the by group and 0 for all other observations in the by group.

Zdroj: http://www.pauldickman.com/teaching/sas/set_by.php

First. and Last. Variables

```
data temp;
input group x;
datalines;
1 23
1 34
1 .
1 45
2 78
2 92
2 45
2 89
2 34
2 76
3 31
4 23
4 12
;
run;
```

```
/*************************************************
The automatic variables first.group and last.group are not saved with the data set.
Here we write them to data set variables to show their contents.
*************************************************/
data new;
set temp;
by group;
first=first.group;
last=last.group;
run;

proc print;
title 'Raw data along with first.group and last.group';
run;
```



Obs	group	x	first	last
1	1	23	1	0
2	1	34	0	0
3	1	.	0	0
4	1	45	0	1
5	2	78	1	0
6	2	92	0	0
7	2	45	0	0
8	2	89	0	0
9	2	34	0	0
10	2	76	0	1
11	3	31	1	1
12	4	23	1	0
13	4	12	0	1

Zdroj: http://www.pauldickman.com/teaching/sas/set_by.php

First. and Last. Variables

```
/*************************************************
A common task in data cleaning is to identify
observations with a duplicate ID number. If we set
the data set by ID, then the observations which
are not duplicated will be both the first and the
last with that ID number. We can therefore write
any observations which are not both first.id and
last.id to a separate data set and examine them.
*************************************************/
data single dup;
set temp;
by group;
if first.group and last.group then output single;
else output dup;
run;

proc print data=dup;
run;

proc print data=single;
run;
```

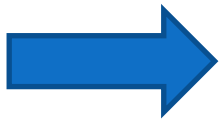
Obs	group	x
1	1	23
2	1	34
3	1	.
4	1	45
5	2	78
6	2	92
7	2	45
8	2	89
9	2	34
10	2	76
11	4	23
12	4	12

Obs	group	x
1	3	31

Zdroj: http://www.pauldickman.com/teaching/sas/set_by.php

First. and Last. Variables

```
/***********************************************
We may also want to do data set processing within
each by group. In this example we construct the
cumulative sum of the variable X within each group.
***********************************************/
data cusum(keep=group sum);
set temp;
by group;
if first.group then sum=0;
sum+x;
if last.group then output;
run;

proc print data=cusum noobs;
title 'Sum of X within each group';
run;
```



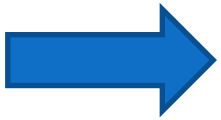
Sum of X within each group

group	sum
1	102
2	414
3	31
4	35

Zdroj: http://www.pauldickman.com/teaching/sas/set_by.php

First. and Last. Variables

```
/*************************************************
As an aside, if you simply want the sum of X within
each group, one of the many way of obtaining this
is with PROC PRINT.
*************************************************/
proc print data=temp;
title 'All data with X summed within each group';
by group;
sum x;
sumby group;
run;
```



```
All data with X summed within each group
---------------------------------- group=1 ---------------
                                    Obs     x
                                      1    23
                                      2    34
                                      3     .
                                      4    45
                                    -----   ---
                                    group   102

---------------------------------- group=2 ----------------
                                    Obs     x

                                      5    78
                                      6    92
                                      7    45
                                      8    89
                                      9    34
                                     10    76
                                    -----   ---
                                    group   414

---------------------------------- group=3 ----------------
                                    Obs     x
                                     11    31

---------------------------------- group=4 ----------------
                                    Obs     x
                                     12    23
                                     13    12
                                    -----   ---
                                    group    35
                                           ===
                                           582
```

Zdroj: http://www.pauldickman.com/teaching/sas/set_by.php

First. and Last. Variables and Sum Statement

- The First. and Last. variables along with the sum statement can be used to create the values for the **summary** data set.

```
data orders(keep=Customer_Name Quantity
                 Product_ID Total_Retail_Price)
     noorders(keep=Customer_Name Birth_Date)
     summary(keep=Customer_Name NumOrders);
   merge orion.customer
         work.order_fact(in=order);
   by Customer_ID;
   if order=1 then do;
      output orders;
      if first.Customer_ID then NumOrders=0;
      NumOrders+1;
      if last.Customer_ID then output summary;
   end;
   else output noorders;
run;
```

DO loops - introduction

- DO loops can be used to do the following:
  - perform repetitive calculations
  - generate data
  - eliminate redundant code
  - execute SAS code conditionally
  - read data
- Rozlišujeme 2 základní formy DO cyklů:
  - Iterative ...pevně daná délka cyklu
  - Conditional iterative ... cyklus běží pokud/dokud je splněna zadaná podmínka

Business Need

- Orion Star wants to encourage employees to save for retirement.
- If 6% of an employee's salary is invested in a 401(k) plan each year (not to exceed $11,000), how much money could be saved after 30, 40, and 50 years?
- (Assume that the average return of the plan is 11%.)

Repetitive Processing

Partial DATA Step

```
data retire(keep = EmployeeID Salary Investment
                   Value_after_30_years
                   Value_after_40_years
                   Value_after_50_years);
   set univ.usemps;
   Retirement = 0;
   Investment = 0.06 * Salary;
   if Investment gt 11000 then Investment = 11000;
   *Year 1;
   Retirement = Retirement + Investment;
   Retirement = Retirement * 1.11;
   *Year 2;
   Retirement = Retirement + Investment;
   Retirement = Retirement * 1.11;
   *Year 3;
         .
         .
         .
   *Year 30;
   Retirement = Retirement + Investment;
   Retirement = Retirement * 1.11;
   Value_after_30_years = Retirement;
```

DO Loop Syntax

- General form of a simple iterative DO loop:

```
DO index-variable=start TO stop <BY increment>;
    SAS statements
END;
```

- The values of *start*, *stop*, and *increment*
  - must be numbers or expressions that yield numbers
  - are established before executing the loop
  - if omitted, *increment* defaults to 1.

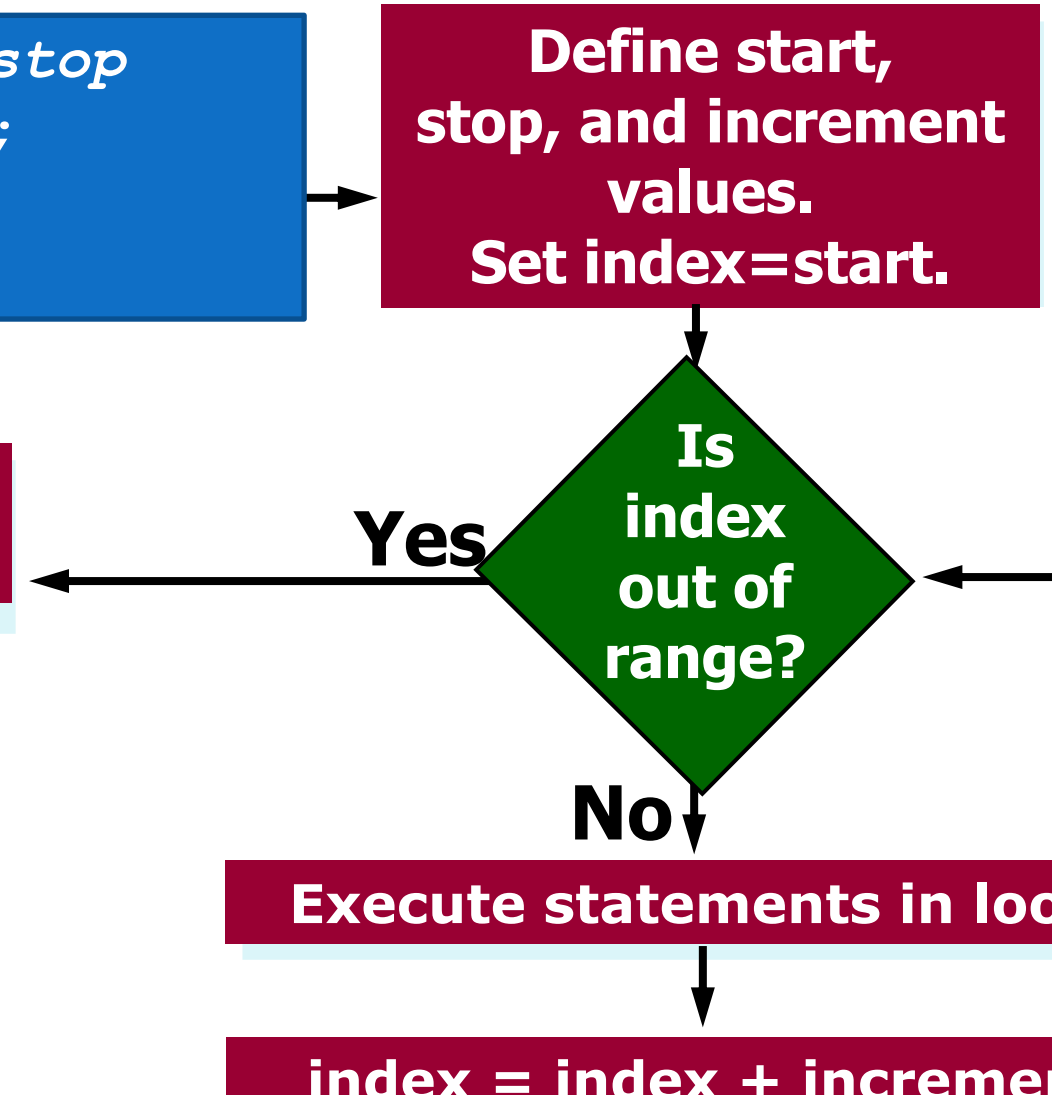
The Iterative DO Statement

- *Index-variable* details:
  - The *index-variable* is written to the output data set by default.
  - At the termination of the loop, the value of *index-variable* is one *increment* beyond the *stop* value.

Modifying the value of *index-variable* affects the number of iterations, and might cause infinite looping or early loop termination.

DO Loop Processing

```
DO index=start TO stop
      BY increment;
   SAS statements
END;
```



DO Loop Syntax

You can define the values used to increment the loop with ***start*** **TO** ***stop*** **<BY** ***increment*****>**.

```
do i=2 to 10 by 2;
```

2 4 6 8 10 12

```
do i=10 to 2 by -2;
```

10 8 6 4 2 0

```
do k=3.6 to 3.8 by .05;
```

3.6 3.65 3.70 3.75 3.80 3.85

DO Loop Syntax

- Expressions:

```
do z=k to n/10;
```

- Dates:

```
do date='01JAN2003'd to '31JAN2003'd;
```

DO Loop Syntax

- General form of a DO loop with a value list:

```
DO index-variable=value1, value2, value3...;
    SAS statements
END;
```

- The values in the list can be numeric or character.
- Discrete numeric values separated by commas:

```
do n=1,5,15,30,60;
```

- Character values enclosed in quotes and separated by commas:

```
do month='JAN','FEB','MAR';
```

DO Loop Code

```
data retire(keep = EmployeeID Salary Investment
                   Value_after_30_years
                   Value_after_40_years
                   Value_after_50_years);
   set univ.usemps;
   Retirement = 0;
   Investment = 0.06 * Salary;
   if Investment gt 11000 then Investment = 11000;
   do year = 1 to 50;
      /* Add the Investment amount each year */
      Retirement = Retirement + Investment;
      Retirement= Retirement * 1.11;
       /* Retirement value after 30, 40 and 50 years */
       if Year = 30 then
          Value_after_30_years = Retirement;
       else if Year = 40 then
          Value_after_40_years = Retirement;
       else if Year = 50 then
          Value_after_50_years = Retirement;
   end;
run;
```

c03s4d1.sas

Conditional Iterative Processing

- You can use DO WHILE and DO UNTIL statements to stop the loop when a condition is met rather than when the loop executed a specific number of times.

To avoid infinite loops, be sure that the specified condition will be met.

The DO WHILE Statement

- The DO WHILE statement executes statements in a DO loop repetitively while a condition is true.
- General form of the DO WHILE loop:

```
DO WHILE (expression);
      <additional SAS statements>
END;
```

- The value of *expression* is evaluated at the **top** of the loop.
- The statements in the loop never execute if *expression* is initially false.

The DO UNTIL Statement

- The DO UNTIL statement executes statements in a DO loop repetitively until a condition is true.

- General form of the DO UNTIL loop:

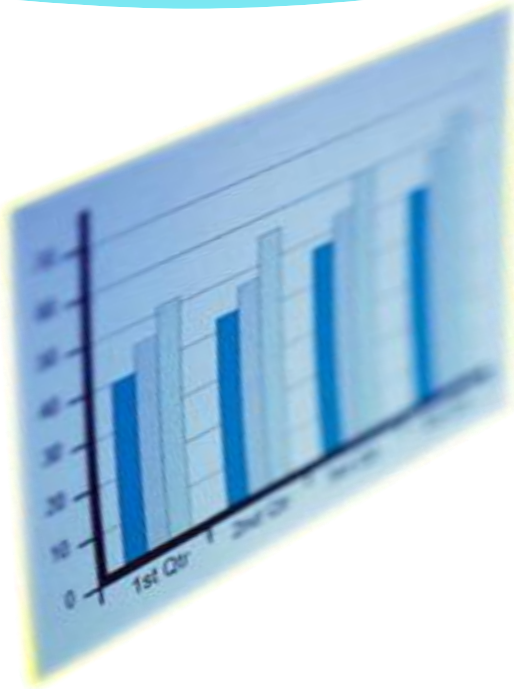
```
DO UNTIL (expression);
      <additional SAS statements>
END;
```

- The value of *expression* is evaluated at the **bottom** of the loop.
- The statements in the loop are executed at least once.

Although the condition is placed at the top of the loop, it is evaluated at the bottom of the loop.

Business Scenario

- Determine the number of years that it would take for an account to exceed $1,000,000 if $5,000 is invested annually at 4.5 percent.



Using the DO UNTIL Statement

```
data invest;
   do until(Capital>1000000);
      Year+1;
      Capital+5000;
      Capital+(Capital*.045);
   end;
run;

proc print data=invest noobs;
   format Capital dollar14.2;
run;
```

- PROC PRINT Output

Capital	Year
$1,029,193.17	52

Iterative DO Loop with a Conditional Clause

- You can combine DO WHILE and DO UNTIL statements with the iterative DO statement.

- General form of the iterative DO loop with a conditional clause:

```
DO index-variable=start TO stop <BY increment>
    WHILE | UNTIL (expression);
    <additional SAS statements>
END;
```

This is one method of avoiding an infinite loop in a DO WHILE or DO UNTIL statements.

Using DO UNTIL with an Iterative DO Loop

- Determine the value of the account again. Stop the loop if 30 years is reached or more than $250,000 is accumulated.

```
data invest;
   do Year=1 to 30 until(Capital>250000);
      Capital+5000;
      Capital+(Capital*.045);
   end;
run;
proc print data=invest noobs;
   format capital dollar14.2;
run;
```

In a DO UNTIL loop, the condition is checked **before** the index variable is incremented.

PROC PRINT Output

Year	Capital
27	$264,966.67

Using DO WHILE with an Iterative DO Loop

- Determine the value of the account again, but this time use a DO WHILE statement.

```
data invest;
   do Year=1 to 30 while(Capital<=250000);
      Capital+5000;
      Capital+(Capital*.045);
   end;
run;
proc print data=invest noobs;
   format capital dollar14.2;
run;
```

In a DO WHILE loop, the condition is checked **after** the index variable is incremented.

PROC PRINT Output

Year	Capital
28	$264,966.67

Nested DO Loops

Nested DO loops are loops within loops.

- Be sure to use different index variables for each loop.
- Each DO statement must have a corresponding END statement.
- The inner loop executes completely for each iteration of the outer loop.

```
DO index-variable-1=start TO stop <BY increment>;
    DO index-variable-2=start TO stop  <BY increment>;
        <additional SAS statements>
    END;
END;
```

Leave statement

- Stops processing the current loop and resumes with the next statement in the sequence.

```
data week;
        input name $ idno start_yr status $ dept $;
        bonus=0;
        do year= start_yr to 1991;
            if bonus ge 500 then leave;
            bonus+50;
        end;
        datalines;
Jones 9011 1990 PT PUB
Thomas 876 1976 PT HR
Barnes 7899 1991 FT TECH
Harrell 1250 1975 FT HR
Richards 1002 1990 FT DEV
Kelly 85 1981 PT PUB
Stone 091 1990 PT MAIT ;
```

Více viz
http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000202782.htm

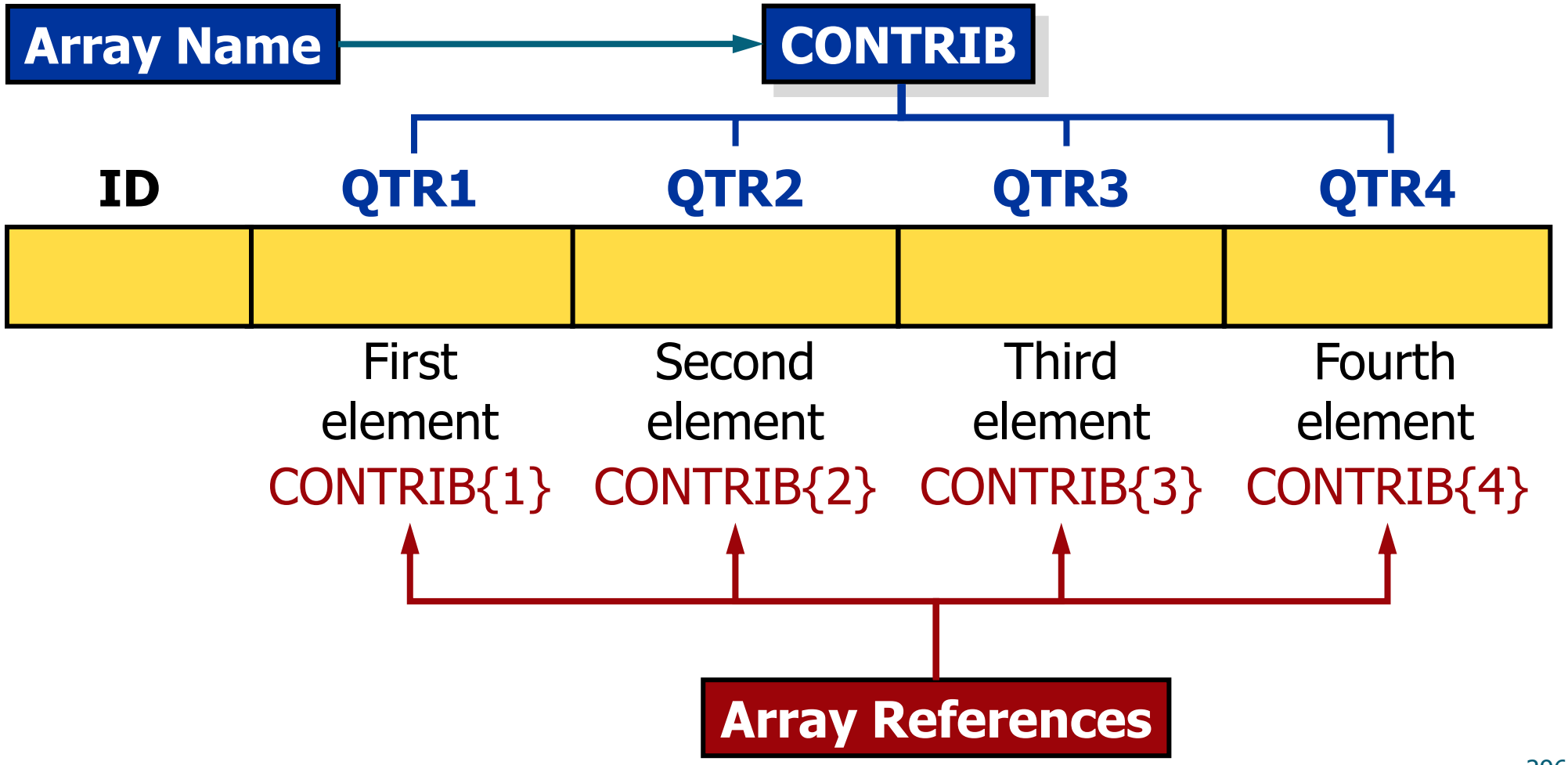
Array Processing

- You can use arrays to simplify programs that
  - perform repetitive calculations
  - create many variables with the same attributes
  - read data
  - rotate SAS data sets by making variables into observations or observations into variables
  - compare variables
  - perform a table lookup

What Is a SAS Array?

- A *SAS array*
  - is a temporary grouping of SAS variables that are arranged in a particular order
  - is identified by an *array name*
  - exists only for the duration of the current DATA step
  - is **not** a variable
- Each value in an array is
  - called an *element*
  - identified by a *subscript* that represents the position of the element in the array.
- When you use an *array reference*, the corresponding value is substituted for the reference.

What Is a SAS Array?



The ARRAY Statement

- The ARRAY statement defines the elements in an array. These elements are processed as a group. You refer to elements of the array by the array name and subscript.

ARRAY *array-name* {*subscript*} <$> <*length*> <*array-elements*> <(*initial-value-list*)>;

{subscript}	the number of elements
$	indicates character elements
length	the length of elements
array-elements	the names of elements

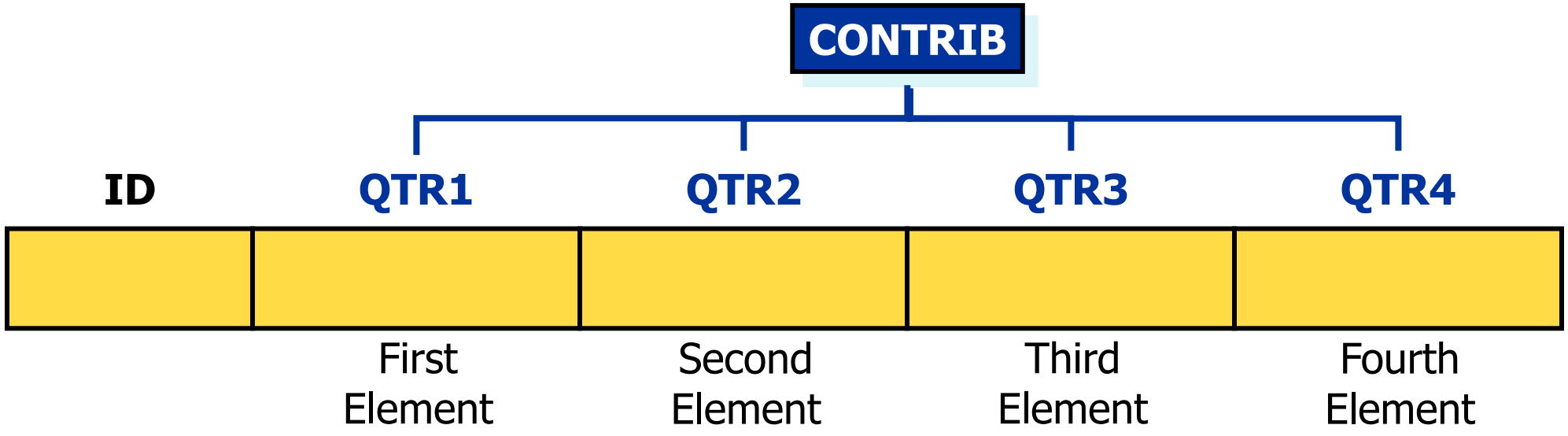
The ARRAY Statement

- The ARRAY statement
  - must contain all numeric or all character elements
  - must be used to define an array before the array name can be referenced
  - creates variables if they do not already exist

Defining an Array

- Write an ARRAY statement that defines the four quarterly contribution variables as elements of an array.

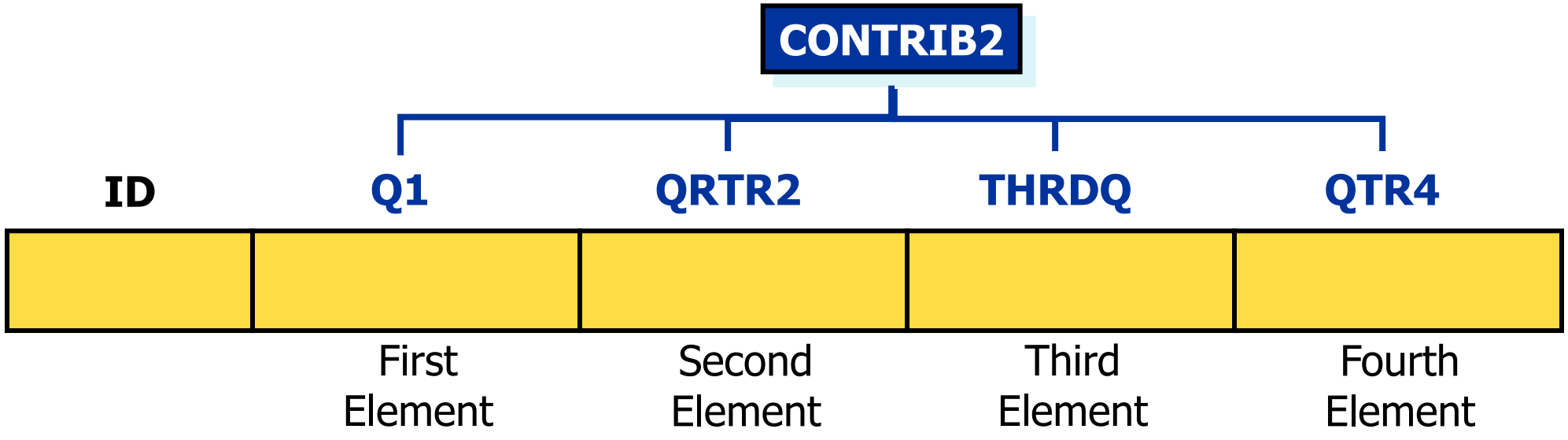
```
array Contrib{4} Qtr1 Qtr2 Qtr3 Qtr4;
```



Defining an Array

- Variables that are elements of an array do not need to have similar, related, or numbered names.

```
array Contrib2{4} Q1 Qrtr2 ThrdQ Qtr4;
```



Defining an Array

```
data charity(keep=employee_id qtr1-qtr4);
   set orion.employee_donations;
   array Contrib1{3} qtr1-qtr4;
   array Contrib2{5} qtr:;
   /* additional SAS statements */
run;
```

The subscript and element-list must agree.

```
177     array Contrib1{3} qtr1-qtr4;
ERROR: Too many variables defined for the dimension(s) specified for
the array Contrib1.
178     array Contrib2{5} qtr:;
ERROR: Too few variables defined for the dimension(s) specified for
the array Contrib2.
```

Processing an Array

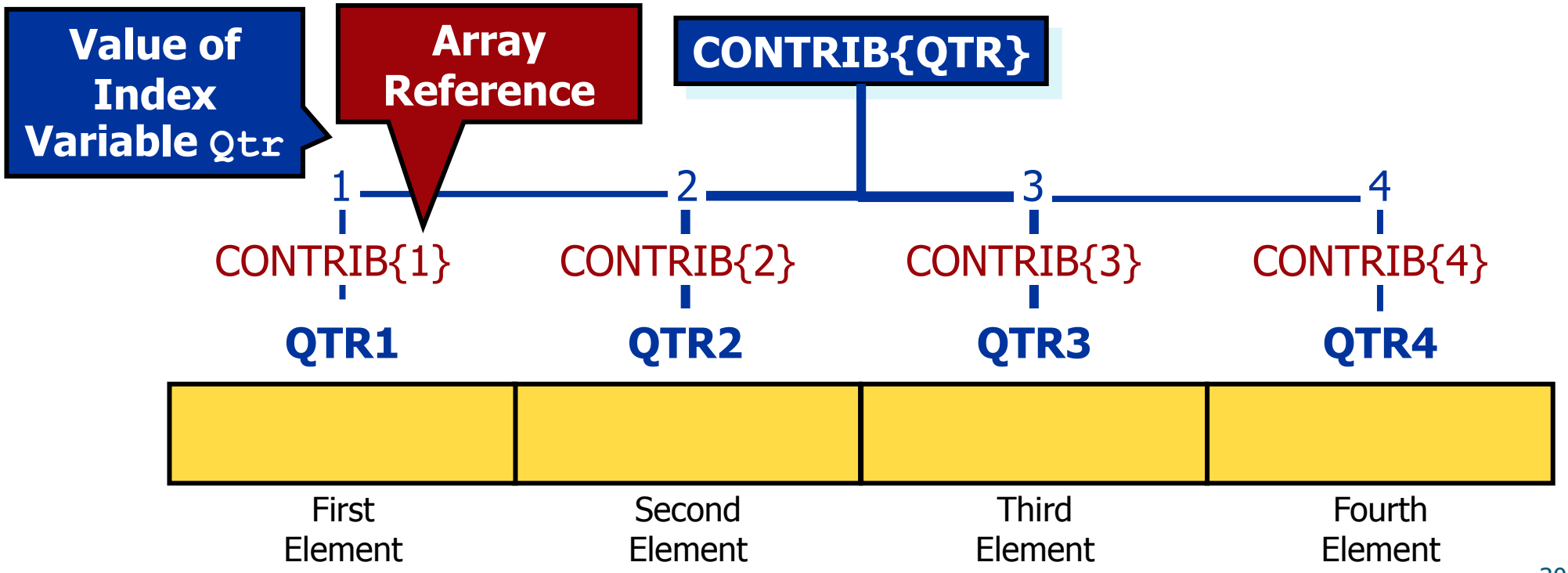
- Array processing often occurs within DO loops. An iterative DO loop that processes an array has the following form:

```
DO index-variable=1 TO number-of-elements-in-array;
      additional SAS statements
         using array-name{index-variable}...
END;
```

- To execute the loop as many times as there are elements in the array, specify that the values of *index-variable* range from 1 to *number-of-elements-in-array*.

Processing an Array

```
array Contrib{4} Qtr1 Qtr2 Qtr3 Qtr4;
do Qtr = 1 to 4;
   Contrib{Qtr} = Contrib{Qtr}*1.25;
end;
```



Performing Repetitive Calculations

```
data charity(drop = Qtr);
   set univ.donate;
   array Contrib{4} Qtr1 Qtr2 Qtr3 Qtr4;
   do Qtr = 1 to 4;
      Contrib{Qtr} = Contrib{Qtr}*1.25;
   end;
run;
```

c03s5d1.sas

Performing Repetitive Calculations

```
proc print data = charity noobs;
run;
```

- Partial PROC PRINT Output

ID	Qtr1	Qtr2	Qtr3	Qtr4
E00224	15.00	41.25	27.50	.
E00367	43.75	60.00	50.00	37.50
E00441	.	78.75	111.25	112.50
E00587	20.00	23.75	37.50	36.25
E00598	5.00	10.00	7.50	1.25

Using an Array as a Function Argument

- The program below passes an array to the SUM function.

```
data test;
   set orion.employee_donations;
   array val{4} qtr1-qtr4;
   Tot1=sum(of qtr1-qtr4);
   Tot2=sum(of val{*});
run;
proc print data=test;
   var employee_id tot1 tot2;
run;
```

The array is passed as if it were a variable list.

- Partial PROC PRINT Output

Obs	Employee_ID	Tot1	Tot2
1	120265	25	25
2	120267	60	60
3	120269	80	80

The DIM Function

- The DIM function returns the number of elements in an array. This value is often used as the stop value in a DO loop.

- General form of the DIM function: DIM(*array_name*)

```
array Contrib{*} qtr:;
num_elements=dim(Contrib);

do i=1 to num_elements;
   Contrib{i}=Contrib{i}*1.25;
end;
run;
```

```
data charity;
   set orion.employee_donations;
   keep employee_id qtr1-qtr4;
   array Contrib{*} qtr:;
   do i=1 to dim(Contrib);
      Contrib{i}=Contrib{i}*1.25;
   end;
run;
```

Creating Variables with Arrays

```
data change;
   set orion.employee_donations;
   drop i;
   array Contrib{4} Qtr1-Qtr4;
   array Diff{3};
   do i=1 to 3;
      Diff{i}=Contrib{i+1}-Contrib{i};
   end;
run;
```

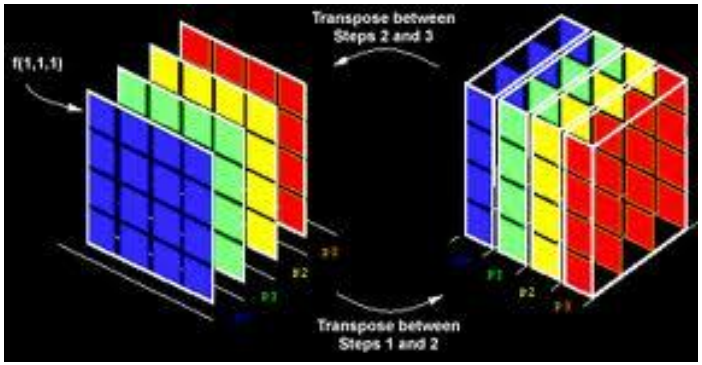
The **Contrib** array refers to existing variables. The **Diff** array creates three variables: **Diff1**, **Diff2**, and **Diff3**.

Array s „nedefinovanou“ dimenzí

```
%let list= Qtr1 Qtr2 Qtr3 Qtr4;

data change;
   set orion.employee_donations;
   array Contrib{*} &list;
      … more SAS statements …
run;
```

```
data tab2;
set tab1;
  array AllNums{*} _numeric_;
  do i = 1 to dim(AllNums);
     … more SAS statements …
  end;
run;
```

6. SAS Data Step – spojování a rotace tabulek



Concatenation

- A *concatenation*
  - combines two or more data sets, one after the other, into a single data set
  - uses the SET statement.
- The new data set
  - contains all observations from the original data sets in sequential order
  - by default, contains all variables from the original data sets.

Concatenation

Data Set 1

Month	Sales
JAN	25354
FEB	28999
MAR	26489

+

Data Set 2

Month	Sales
APR	23541
MAY	24877
JUN	24653

=

Combined

Month	Sales
JAN	25354
FEB	28999
MAR	26489
APR	23541
MAY	24877
JUN	24653

Concatenation

Month	Sales	Goal
JAN	25354	26500
FEB	28999	27000
MAR	26489	27500

+

Month	Sales
APR	23541
MAY	24877
JUN	24653

=

Month	Sales	Goal
JAN	25354	26500
FEB	28999	27000
MAR	26489	27500
APR	23541	.
MAY	24877	.
JUN	24653	.

Coding for Concatenation

- Use the SET statement in a DATA step to concatenate SAS data sets.
- General form of a DATA step concatenation:

```
DATA SAS-data-set ;
     SET SAS-data-set-1 SAS-data-set-2 ;
RUN;
```

- Příklad:

```
data univ.mastercustomers;
   set univ.uscustomers
       univ.usnewcustomers;
run;
```

c04s1d1.sas

Interleaving

- *Interleaving* uses a SET statement and a BY statement to combine two or more SAS data sets.
- The data set created through interleaving
  - contains all observations from the original data sets
  - is arranged by the values of the BY variable(s)
  - by default, contains all variables from the original data sets.

Interleaving

Month	Sales
JAN	26510
FEB	24530
MAR	20122
MAR	14258

+

Month	Sales
JAN	21654
JAN	19873
FEB	22306
FEB	24003
MAR	19855
APR	23502

=

Month	Sales
JAN	26510
JAN	21654
JAN	19873
FEB	24530
FEB	22306
FEB	24003
MAR	20122
MAR	14258
MAR	19855
APR	23502

Sorting a SAS Data Set

- Before you interleave SAS data sets, all data sets must be sorted on the variable(s) that determine(s) the order of observations in the final data set.
- You can use PROC SORT to sort data.
- General form of PROC SORT:

```
PROC SORT DATA=input-SAS-data-set
          OUT=output-SAS-data-set;
    BY <DESCENDING> by-variable(s);
RUN;
```

Sorting a SAS Data Set

- The SORT procedure
  - rearranges the observations in a SAS data set
  - can create a new SAS data set that contains the rearranged observations
  - can sort on multiple variables
  - can sort in ascending (default) or descending order
  - does not generate printed output
  - treats missing values as the smallest possible value.

PROC SORT Example

```
proc sort data=univ.uscustomers
          out=uscustomers;
   by CustomerID;
run;

proc sort data=univ.usnewcustomers
          out=usnewcustomers;
   by CustomerID;
run;
```

c04s1d2.sas

Coding to Interleave SAS Data Sets

- After the original data sets are properly sorted, the DATA step with a SET statement and a BY statement is used to interleave the sorted data sets.
- General form of the DATA step:

```
DATA SAS-data-set;
     SET SAS-data-set-1 SAS-data-set-2;
     BY variable(s);
RUN;
```

Interleaving Example

```
data univ.mastercustomers;
   set uscustomers usnewcustomers;
   by CustomerID;
run;
```

c04s1d2.sas

Match-Merging

- *Match-merging*
  - combines observations from two or more SAS data sets into a single observation in a new data set according to the values of a common variable
  - can be used to combine observations having a one-to-one, one-to-many, or many-to-many relationship.

Business Scenario

- The data set **univ.customerorders** contains sales order information.

Customer ID	Order Date	OrderID	ProductID	Quantity	Price
029858	15128	1239347234	230100600005	4	130
029858	15171	1239686972	240800100020	1	122
029858	15171	1239686972	240800100036	1	468

- The **univ.mastercustomers** contains mailing address and other information about customers.

Customer ID	Customer FirstName	Customer LastName	CustomerAddress	CustomerGroup
000492	David	Dulin	147 Bowling Farm Ct	Orion Club
000551	Blu	Peachey	85 Lake Boone Trl	Internet/...
000738	Jerry	Krejci	700 Fernwood Dr	Orion Club Gold

Match-Merging

Create a data set named **shipping** that contains the name and addresses of customers who have placed orders by merging the **univ.customerorders** data set and the **univ.mastercustomers** data set.

Partial SAS Output

Customer ID	Order Date	...	Customer First Name	Customer LastName	CustomerAddress
029858	15128	...	Alice	Maxam	81 Flagstone Pl
029858	15171	...	Alice	Maxam	81 Flagstone Pl
029858	15171	...	Alice	Maxam	81 Flagstone Pl

Coding the Match-Merge

- Data sets combined with a match-merge must be sorted by the common variable. Use PROC SORT to prepare data if necessary.
- The DATA step with a MERGE statement and a BY statement is used to match-merge two or more SAS data sets.
- General form of the DATA step:

```
DATA SAS-data-set ;
      MERGE SAS-data-set-1 SAS-data-set-2;
      BY variable(s);
RUN;
```

Match-Merging with Nonmatches

```
proc sort data=univ.customerorders
             out = customerorders;
   by CustomerID;

proc sort data=univ.mastercustomers
             out=mastercustomers;
   by CustomerID;

data shipping;
   merge customerorders
         mastercustomers;
   by CustomerID;
run;
```

c04s2d1.sas

Match-Merging with Nonmatches

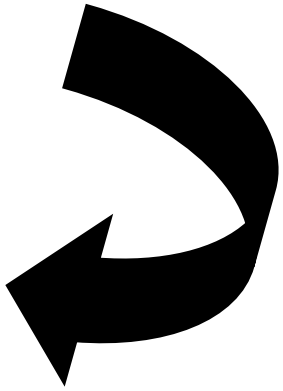
- By default, SAS writes all observations, matches and nonmatches, to the output data set.

Partial **customerorders** data set

CustomerID	OrderDate
029858	15265
029858	15278
030643	15129

Partial **mastercustomers** data set

CustomerID	CustomerAddress
029858	81 Flagstone Pl
030596	582 Guffy Drive
030643	13 Highfalls Court



Partial **work.shipping** data set

CustomerID	OrderDate	CustomerAddress
029858	15265	81 Flagstone Pl
029858	15278	81 Flagstone Pl
030596	.	582 Guffy Drive
030643	15129	13 Highfalls Court

Controlling Nonmatches

- The IN= data set option identifies whether a SAS data set contributed data to the current observation.
- An IF statement conditionally enables a following statement to execute.
- Combined use of the two techniques writes out only matches to the final data set.

General form of the IN= data set option:

SAS-data-set (IN = *variable*)

variable is a temporary numeric variable that has two possible values:

0	indicates that the data set did **not** contribute to the current observation.
1	indicates that the data set **did** contribute to the current observation.

The IN= Data Set Option

- Využití při spojování tabulek:

```
DATA SAS-data-set ;
     MERGE       SAS-data-set-1 (IN=IN1)
                 SAS-data-set-2 (IN=IN2);
     BY variable(s);
RUN;
```

Eliminating Nonmatches

Resulting Data Set

```
proc sort data=univ.customerorders
            out=customerorders;
   by CustomerID;
run;

proc sort data=univ.mastercustomers
            out=mastercustomers;
   by CustomerID;
run;

data shipping;
   merge customerorders (in=inorders)
         mastercustomers (in=inmaster);
   by CustomerID;
   if inorders=1 and inmaster=1;
run;
```

c04s2d2.sas

Eliminating Nonmatches

- The observations that do **not** appear in **both** data sets are **not** written to the new data set.

Partial **`customerorders`** data set

CustomerID	OrderDate
029858	15265
029858	15278
030643	15129

Partial **`mastercustomers`** data set

CustomerID	CustomerAddress
029858	81 Flagstone Pl
~~030596~~	~~582 Guffy Drive~~
030643	13 Highfalls Court

Partial **`work.shipping`** data set

CustomerID	OrderDate	CustomerAddress
029858	15265	81 Flagstone Pl
029858	15278	81 Flagstone Pl
030643	15129	13 Highfalls Court

The RENAME= Data Set Option

- The RENAME= data set option can be used to change the name of a variable from an input data set.
- General form of the RENAME= data set option:

SAS-data-set(**RENAME** = (*old-name* = *new-name*))

Using the RENAME= Option

- If the key variable in the last example were named differently in each data set, then the RENAME= option would need to be used.

Partial Orders Information

CustNum	OrderDate
029858	15265
029858	15278
030643	15129

Partial Customer Information

CustomerID	CustomerAddress
029858	81 Flagstone Pl
030596	582 Guffy Drive
030643	13 Highfalls Court

Using the RENAME= Option

```
proc sort data=univ.customerorders
            out=customerorders;
   by CustomerID;
run;

proc sort data=univ.mastercustomers
            out=mastercustomers;
   by CustomerID;
run;

data shipping;
   merge customerorders ((in=inorders)
                  rename=(CustNum=CustomerID))
         mastercustomers (in=inmaster);
   by CustomerID;
   if inorders=1 and inmaster=1;
run;
```

Rotating a SAS Data Set

- Restructure the input data set, and create a separate observation for each nonmissing quarterly contribution. The output data set, **rotate**, should contain only **Employee_ID**, **Period**, and **Amount**.

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	Paid_By
120265	.	.	.	25	Cash or Check
120267	15	15	15	15	Payroll Deduction
120269	20	20	20	20	Payroll Deduction

Employee_ID	Period	Amount
120265	Qtr4	25
120267	Qtr1	15
120267	Qtr2	15
120267	Qtr3	15
120267	Qtr4	15
120269	Qtr1	20
120269	Qtr2	20
120269	Qtr3	20
120269	Qtr4	20

Rotating a SAS Data Set

- The DATA step below rotates the input data set. An output observation will be written if a contribution was made in a given quarter.

```
data rotate (keep=Employee_Id Period Amount);
   set orion.employee_donations
             (drop=recipients paid_by);
   array contrib{4} qtr1-qtr4;
   do i=1 to 4;
      if contrib{i} ne . then do;
         Period=cats("Qtr",i);
         Amount=contrib{i};
         output;
       end;
   end;
run;
```

Only include nonmissing values

The TRANSPOSE Procedure

```
PROC TRANSPOSE DATA=input-data-set
                    <OUT=output-data-set>
                    <NAME = variable-name>;
    <BY <DESCENDING> variable-1
      <...<DESCENDING> variable-n> <NOTSORTED>;>
    <VAR variable(s);>
    <ID variable;>
RUN;
```

NAME=	specifies a new name for the _NAME_ column. The values in this column identify the variable that supplied the values in the row.
BY	specifies the variable(s) to use to form BY groups.
VAR	specifies the variable(s) to transpose.
ID	specifies the variable whose values will become the new variables.

The TRANSPOSE Procedure

- The TRANSPOSE procedure
  - transposes selected variables into observations
  - transposes numeric variables by default
  - transposes character variables only if explicitly listed in a VAR statement.

Using the Transpose Procedure

Start with a simple PROC TRANSPOSE step:

```
proc transpose
     data=orion.employee_donations
     out=rotate2;
run;
```

Partial Listing of **rotate2**

NAME	_LABEL_	COL1	COL2	COL3	...	COL124
Employee_ID	Employee ID	120265	120267	120269		121147
Qtr1		.	15	20		10
Qtr2		.	15	20		10
Qtr3		.	15	20		10
Qtr4		25	15	20		10

The output is very different from the desired results. A row was created for each variable. A column was created for each of the 124 observations.

Results of a Simple Transposition

- Compare PROC TRANSPOSE output to the original data:

Partial Listing of **orion.employee_donations**

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	Paid_By
120265	.	.	.	25	Cash or Check
120267	15	15	15	15	Payroll Deduction
120269	20	20	20	20	Payroll Deduction

Partial Listing of **rotate2**

NAME	_LABEL_	COL1	COL2	COL3	...	COL124
Employee_ID	Employee ID	120265	120267	120269		121147
Qtr1		.	15	20		10
Qtr2		.	15	20		10
Qtr3		.	15	20		10
Qtr4		25	15	20	.	10

- All the numeric variables were transposed by default. **Paid_By**, a character variable, was not transposed.

Results of a Simple Transposition

Partial Listing of **orion.employee_donations**

Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4	Paid_By
120265	.	.	.	25	Cash or Check
120267	15	15	15	15	Payroll Deduction
120269	20	20	20	20	Payroll Deduction
120270	20	10	5	.	Cash or Check
120271	20	20	20	20	Payroll Deduction

Partial Listing of **rotate2**

NAME	_LABEL_	COL1	COL2	COL3	...	COL124
Employee_ID	Employee ID	120265	120267	120269		121147
Qtr1		.	15	20		10
Qtr2		.	15	20		10
Qtr3		.	15	20		10
Qtr4		25	15	20	.	10

- **Each observation (row) in the input data set becomes a variable (column) in the output data set.**

PROC TRANSPOSE Results

- The data should be grouped by **`Employee_ID`** with a separate observation for each transposed variable.

Partial Listing of **`rotate2`**

NAME	_LABEL_	COL1	COL2	COL3	...	COL124
Employee_ID	Employee ID	120265	120267	120269		120271
Qtr1		.	15	20		20
Qtr2		.	15	20		20
Qtr3		.	15	20		20
Qtr4		25	15	20		20

Employee_ID	_NAME_	COL1
120265	Qtr1	.
120265	Qtr2	.
120265	Qtr3	.
120265	Qtr4	25
120267	Qtr1	15
120267	Qtr2	15
...		

The BY Statement

Use a BY statement to group the output by **Employee_ID**.

```
proc transpose
     data=orion.employee_donations
     out=rotate2;
   by Employee_ID;
run;
proc print data=rotate2 noobs;
run;
```

All numeric variables other than the BY variable are transposed.

Improved PROC TRANSPOSE Results

Use of the BY statement results in one observation for each transposed variable per **Employee_ID**, and includes missing values.

Partial PROC PRINT Output

Employee_ID	_NAME_	COL1
120265	Qtr1	.
120265	Qtr2	.
120265	Qtr3	.
120265	Qtr4	25
120267	Qtr1	15
120267	Qtr2	15
120267	Qtr3	15
120267	Qtr4	15

If there were additional numeric variables, an observation would be created for each.

The VAR Statement

- The VAR statement is used to specify which variables to transpose. It can include character and numeric variables.

```
proc transpose
     data=orion.employee_donations
     out=rotate2;
   by Employee_ID;
   var Qtr1-Qtr4;
run;
proc print data=rotate2 noobs;
run;
```

The VAR statement has no effect in this example because **Qtr1-Qtr4** will be transposed by default.

Renaming Variables in PROC TRANSPOSE

```
proc transpose
        data=orion.employee_donations
        out=rotate2(rename=(col1=Amount))
        name=Period;
   by employee_id;
run;
proc print data=rotate2 noobs;
run;
```

The RENAME= data set option is used to change the name of COL1.

The PROC TRANSPOSE option, NAME=, is used to rename _NAME_.

Partial Listing of `rotate2`

Employee_ID	Period	Amount
120265	Qtr1	.
120265	Qtr2	.
120265	Qtr3	.
120265	Qtr4	25
120267	Qtr1	15
120267	Qtr2	15

The WHERE= Data Set Option

- There is no option or statement in PROC TRANSPOSE to eliminate observations with missing values for the transposed variable. However, this can be achieved using a WHERE= data set option in the output data set.

```
proc transpose
        data=orion.employee_donations
        out=rotate2(rename=(col1=Amount)
                    where=(Amount ne .))
        name=Period;
   by employee_id;
run;
proc print data=rotate2 noobs;
run;
proc freq data=rotate2;
   tables Period/nocum nopct;
   label Period=" ";
run;
```

No Missing Values

Partial PROC PRINT Output:

Employee_ID	Period	Amount
120265	Qtr4	25
120267	Qtr1	15
120267	Qtr2	15
120267	Qtr3	15
120267	Qtr4	15
120269	Qtr1	20
120269	Qtr2	20
120269	Qtr3	20
120269	Qtr4	20
120270	Qtr1	20
120270	Qtr2	10
120270	Qtr3	5

The resulting data set has no missing values.

Business Scenario

- The manager of Sales asked for a report showing monthly sales and a total for each customer.

Sketch of the Desired Report

Monthly Sales by Customer

Customer_ID	Month1	Month2	…	Month12	Total
1	1000	.		500	2000
2	.	.		200	750
3	1200	.		.	2200
4	500	150		350	1000
5	.	1000		.	2500

Business Scenario Considerations

- The data set **orion.order_summary** contains an observation for each month in which a customer placed an order (101 total observations). The data set is sorted by **Customer_ID** and has no missing values.

Partial Listing of **orion.order_summary**

Customer_ID	Month	Order_ Sale_Amt
5	5	478.00
5	6	126.80
5	9	52.50
5	12	33.80
10	3	32.60
10	4	250.80
10	5	79.80
10	6	12.20
10	7	163.29

The number of observations per customer varies.

Business Scenario Considerations

- The report requires rotating the columns into rows. Use PROC TRANSPOSE again to restructure the data set, and this time from narrow to wide.

Customer_ID	Order_Month	Sale_Amt
5	5	478.00
5	6	126.80
5	9	52.50
5	12	33.80
10	3	32.60

Desired Output

Customer_ID	Month1	...	Month5	Month6	...	Month9	...	Month12
5	.		478.00	126.80		52.50		33.80

Using PROC TRANSPOSE

- Start with a simple PROC TRANSPOSE.

Partial Listing of **orion.order_summary**

Customer_ID	Month	Order_ Sale_Amt
5	5	478.00
5	6	126.80
5	9	52.50
5	12	33.80
10	3	32.60
10	4	250.80
10	5	79.80

101 observations

```
proc transpose data=orion.order_summary
               out=annual_orders;
run;
proc print data=annual_orders noobs;
run;
```

Using PROC TRANSPOSE

- The resulting data set has three observations, one for each numeric variable in the input data set: **Customer_ID**, **Order_Month**, and **Sale_Amt**.

NAME	_LABEL_	COL1	COL2	COL3	COL4	COL5	...	COL101
Customer_ID	Customer ID	5	5.0	5.0	5.0	10.0		70201.0
Order_Month		5	6.0	9.0	12.0	3.0		8.0
Sale_Amt		478	126.8	52.5	33.8	32.6		1075.5

Customer 5

The variables **COL1-COL101** represent the 101 observations in the input data set.

Group the output by **Customer_ID**.

The BY Statement

- The BY statement groups by **Customer_ID** and produces an observation for each transposed variable, **Order_Month** and **Sale_Amt**.

```
proc transpose data=orion.order_summary
               out=annual_orders;
   by Customer_ID;
run;
```

Notice the varying number of columns for each customer.

Customer_ID	_NAME_	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8	COL9
5	Order_Month	5.0	6.0	9.0	12.0	.	.	.	.	.
5	Sale_Amt	478.0	126.8	52.5	33.8	.	.	.	.	.
10	Order_Month	3.0	4.0	5.0	6.0	7.00	8.0	11.0	12.0	.
10	Sale_Amt	32.6	250.8	79.8	12.2	163.29	902.5	1894.6	143.3	.
11	Order_Month	9.0	.	.	.	.	.	.	.	.
11	Sale_Amt	78.2	.	.	.	.	.	.	.	.

Creating Columns Based on a Variable

- Instead of transposing **`Order_Month`**, use its values to create new variables. A value of 5.0 represents orders placed in May, 6.0 represents orders placed in June, and so on.

Customer_ID	_NAME_	COL1	COL2	COL3	COL4	COL5	COL6	COL7	COL8	COL9
5	Order_Month	5.0	6.0	9.0	12.0	.	.	.	.	.
5	Sale_Amt	478.0	126.8	52.5	33.8	.	.	.	.	.
10	Order_Month	3.0	4.0	5.0	6.0	7.00	8.0	11.0	12.0	.
10	Sale_Amt	32.6	250.8	79.8	12.2	163.29	902.5	1894.6	143.3	.
11	Order_Month	9.0	.	.	.	.	.	.	.	.
11	Sale_Amt	78.2	.	.	.	.	.	.	.	.

- Add an ID statement.

The ID Statement

- The ID statement identifies the variable whose values will become the names of the new columns.

```
proc transpose data=orion.order_summary
               out=annual_orders;
   by Customer_ID;
   id Order_Month;
run;
```

The values of `Order_Month` (1, 2, 3, ... 12) are used to create variable names `_1` through `_12`.

Customer_ID	_NAME_	_5	_6	_9	_12	...
5	Sale_Amt	478.0	126.80	52.5	33.80	
10	Sale_Amt	79.8	12.20	.	143.30	
11	Sale_Amt	.	.	78.2	.	
12	Sale_Amt	.	48.40	87.2	.	
18	Sale_Amt	.	.	.	.	

The remaining variable, **Sale_Amt**, is transposed.

Changing the Variable Names

- The PREFIX= option is used to set a prefix for each new variable name. The prefix replaces the underscore.

```
proc transpose data=orion.order_summary
               out=annual_orders
               prefix=Month;
      by Customer_ID;
      id Order_Month;
run;
```

Customer_ID	_NAME_	Month5	Month6	Month9	...
5	Sale_Amt	478.0	126.80	52.5	
10	Sale_Amt	79.8	12.20	.	
11	Sale_Amt	.	.	78.2	
12	Sale_Amt	.	48.40	87.2	
18	Sale_Amt	.	.	.	

Print the Transposed Data Set

- A VAR statement in the PRINT procedure specifies the desired order of the variables.

```
proc print data=annual_orders noobs;
   var Customer_ID Month1-Month12;
run;
```

Customer_ID	Month1	Month2	Month3	Month4	Month5	...
5	.	.	.	.	478.0	
10	.	.	32.6	250.8	79.8	
11	.	.	.	.	.	
12	.	117.6	.	.	.	
18	.	29.4	.	.	.	
24	195.6	.	46.9	.	.	
27	174.4	.	140.7	205.0	.	

Další detaily o Proc Transpose

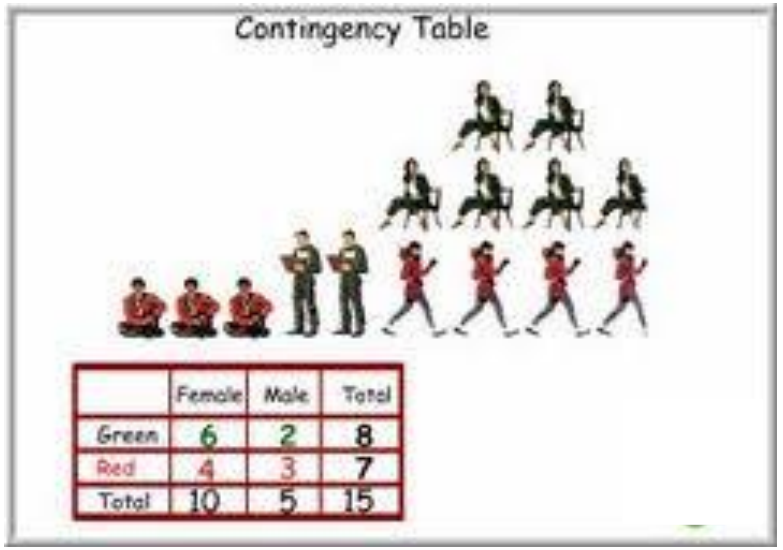
- http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#transpose-overview.htm

- http://www.google.cz/url?sa=t&source=web&cd=10&ved=0CHsQFjAJ&url=http%3A%2F%2Fwww.hasug.org%2Fnewsletters%2Fhasug200408%2Fproc_transpose.ppt&rct=j&q=sas%20proc%20transpose&ei=y1uOTe_xNcbcsgaN07WVCg&usg=AFQjCNEZzkLpkXcDMLRl8kmQdFYRkM6MIA&cad=rja

- http://support.sas.com/resources/papers/proceedings09/060-2009.pdf

- http://www2.sas.com/proceedings/sugi27/p016-27.pdf

7. Explorační analýza
- základní popis dat, tabulky



Explorační analýza – PROČ?

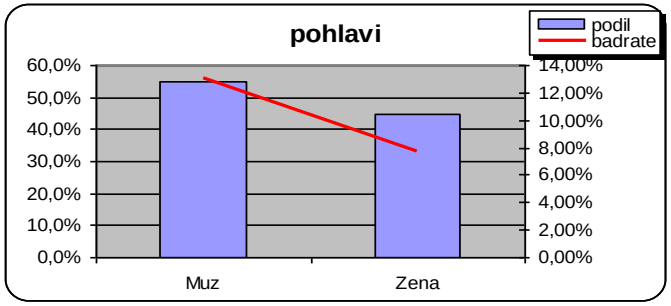
- Je třeba pochopit data:
  - najít chyby v datech
  - najít vzory v datech
  - najít porušení statistických předpokladů, testování hypotéz
  - ...a především proto, že pokud to neuděláme, budeme mít velké problémy později.

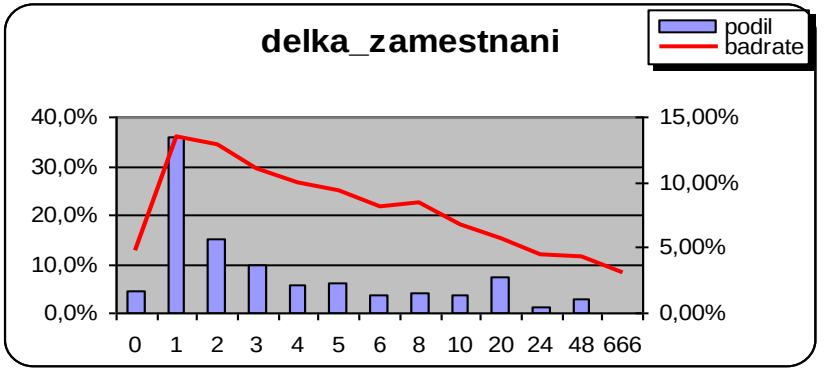
Explorace dat - jednorozměrná

- Frekvenční tabulky, histogramy:

	pocet	podil	badrate
Muz	248 768	55,0%	13,08%
Zena	203 194	45,0%	7,69%
Total	451 962	100,0%	10,66%

delka_zamestnani	pocet	podil	badrate
0	20 825	4,6%	4,69%
1	163 144	36,1%	13,43%
2	67 462	14,9%	12,80%
3	43 778	9,7%	10,97%
4	26 256	5,8%	10,01%
5	27 526	6,1%	9,32%
6	15 893	3,5%	8,16%
8	18 036	4,0%	8,39%
10	17 195	3,8%	6,72%
20	33 641	7,4%	5,60%
24	5 176	1,1%	4,48%
48	12 934	2,9%	4,28%
666	96	0,0%	3,13%
Total	451 962	100,0%	10,66%

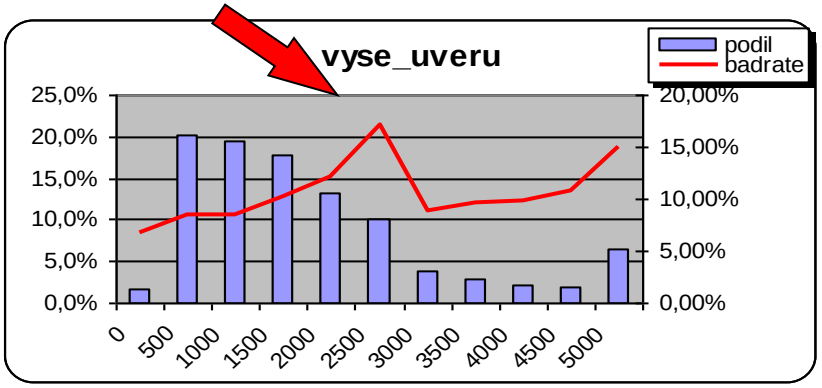




Explorace dat - jednorozměrná

- výše úvěru vs. cílová proměnná (bad rate).
  - je třeba vysvětlit veškeré „nestandardní“ závislosti
  - úplné pochopení dat vede k interpretovatelným modelům s vysokou prediktivní silou

OK? Nebo je to způsobeno jiným faktorem???

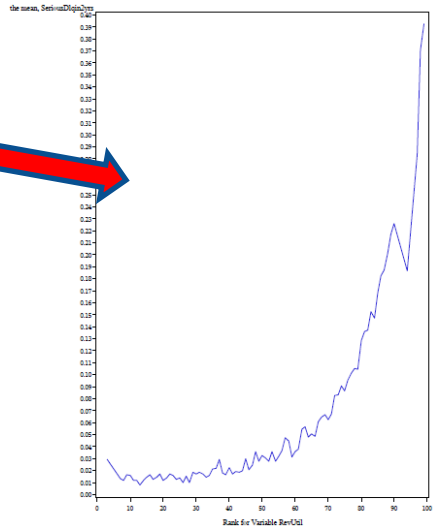


Explorace dat - jednorozměrná

- spojité proměnné:
  - průměr
  - modus
  - kvantily
  - rozptyl
  - min./maximální hodnota
  - vztah k cílové proměnné

Variable	Mean	Std Dev	Variance	Minimum	Maximum	N Miss	1st Pctl	5th Pctl
Bo_Age	36.7928463	10.0283282	100.5673670	18.0000000	99.0000000	0	21.0000000	23.0000000
Ln Orig	153467.57	68370.61	4674539752	19600.00	599000.00	0	42750.00	62000.00
Orig_LTV_Ratio_Pct	93.0798522	8.8537162	78.3882906	20.0000000	111.0000000	0	63.0000000	80.0000000
Credit score	687.6683165	62.9002322	3956.44	440.0000000	999.0000000	0	527.0000000	575.0000000
Tot mthly incm	5024.71	2952.16	8715254.06	500.0000000	65000.00	0	1473.00	2000.00
Median_state_inc	44945.07	5431.51	29501323.98	32589.00	57352.00	0	33948.00	38550.00
DTI Ratio	0.3747207	0.1758619	0.0309274	0	3.4280770	0	0	0
orig_apprd_val_amt	170661.44	81775.07	6687162030	0	870000.00	0	47000.00	68000.00
pur_prc_amt	164681.56	79719.84	6355252570	20000.00	870000.00	0	45000.00	65000.00
Tot_mthly_debt_exp	1745.46	1089.20	1186348.36	0	17225.00	0	0	0

Variable	10th Pctl	Lower Quartile	Median	Upper Quartile	90th Pctl	95th Pctl	99th Pctl
Bo_Age	25.0000000	30.0000000	37.0000000	41.0000000	50.0000000	56.0000000	69.0000000
Ln Orig	75000.00	103500.00	141500.00	190950.00	255000.00	285600.00	322700.00
Orig_LTV_Ratio_Pct	80.0000000	90.0000000	95.0000000	100.0000000	100.0000000	100.0000000	102.0000000
Credit score	606.0000000	647.0000000	688.0000000	737.0000000	769.0000000	783.0000000	801.0000000
Tot mthly incm	2402.00	3245.00	4632.00	6000.00	7975.00	9611.00	14830.00
Median_state_inc	39000.00	40171.00	43988.00	49894.00	53275.00	56763.00	56772.00
DTI Ratio	0.1738380	0.2778700	0.3761930	0.4710300	0.5746010	0.6383500	0.8506940
orig_apprd_val_amt	82000.00	113000.00	154000.00	214000.00	285000.00	327000.00	415000.00
pur_prc_amt	78000.00	108900.00	148650.00	205000.00	275512.00	319900.00	405000.00
Tot_mthly_debt_exp	655.0000000	1073.00	1578.00	2253.00	3008.00	3604.00	5290.00

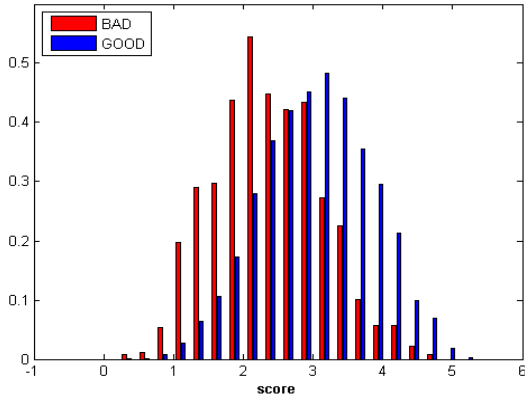


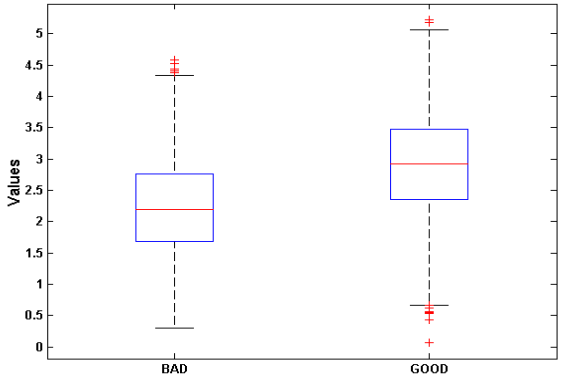
- U dichotomické cílové prom. (0/1) jde o relativní zastoupení vybrané kategorie (např. bad rate) pro vhodné intervaly zkoumané proměnné. Intervaly můžou být:
  - pevně dané, např. 0-10,10-20,...
  - decily/percentily
  - klouzavé okno

- často je vhodná kategorizace (následně frekvenční tabulky, vztah k cílové proměnné)

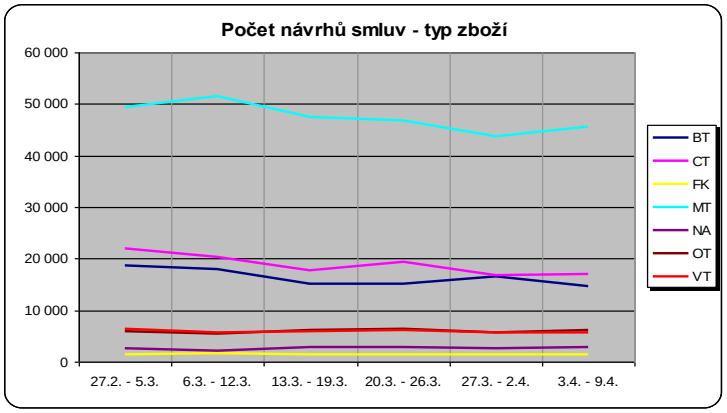
Explorace dat - jednorozměrná

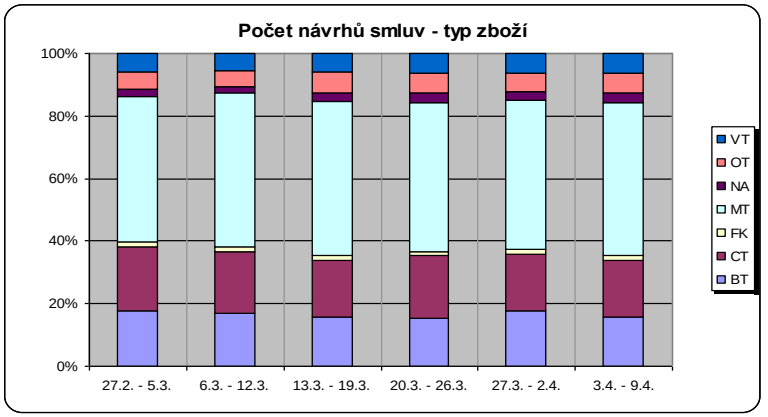
- Histogramy, box ploty





- Stabilita v čase





Explorace dat - vícerozměrná

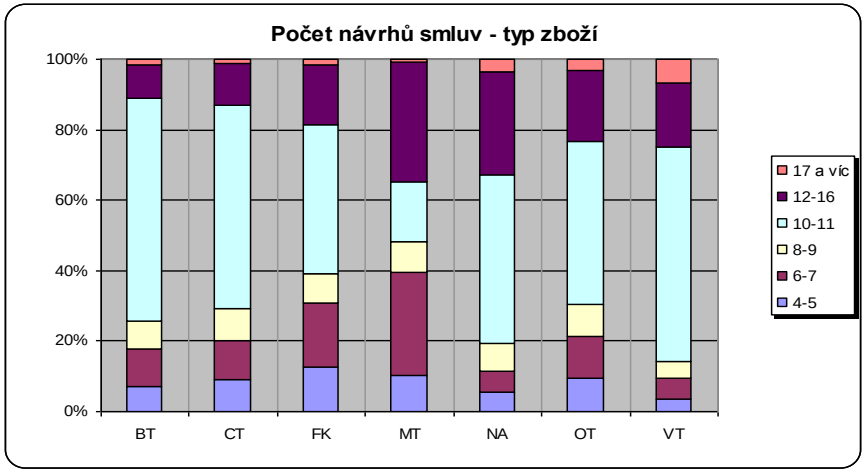
- Kontingenční tabulky
  - absolutní četnosti slouží ke kontrole jestli některá kombinace hodnot není příliš málo četná
  - relativní četnosti (řádkově + sloupcově podmíněné) slouží k odhalení vztahů mezi proměnnými

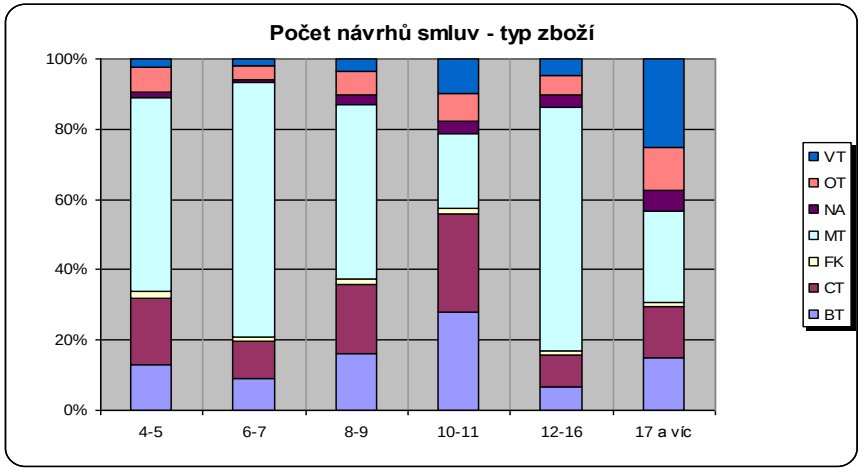
	do 5 000	5 000 - 10 000	10 000 - 15 000	víc než 15 000
BT	4 291	8 581	9 176	9 044
CT	7 587	12 493	6 500	7 236
FK	258	1 017	851	557
MT	27 191	39 551	16 524	5 992
NA	426	1 088	1 114	2 737
OT	2 478	3 689	2 103	3 475
VT	384	1 001	963	9 086

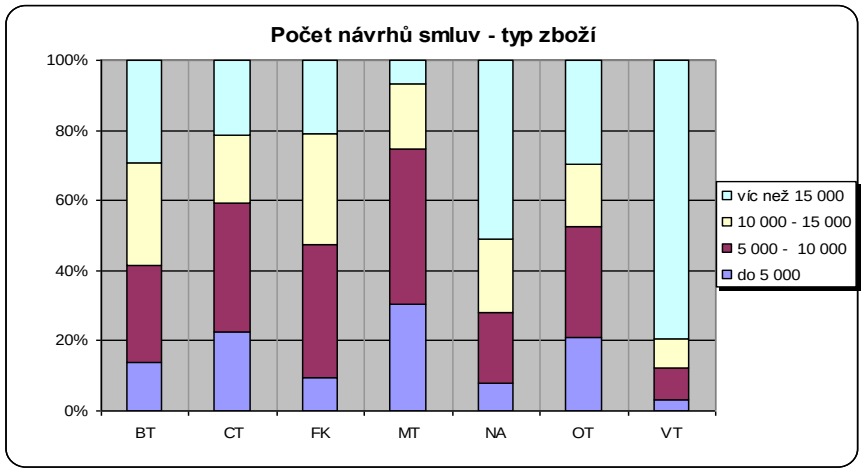
row%	do 5 000	5 000 - 10 000	10 000 - 15 000	víc než 15 000
BT	13,8%	27,6%	29,5%	29,1%
CT	22,4%	36,9%	19,2%	21,4%
FK	9,6%	37,9%	31,7%	20,8%
MT	30,5%	44,3%	18,5%	6,7%
NA	7,9%	20,3%	20,8%	51,0%
OT	21,1%	31,4%	17,9%	29,6%
VT	3,4%	8,8%	8,4%	79,5%

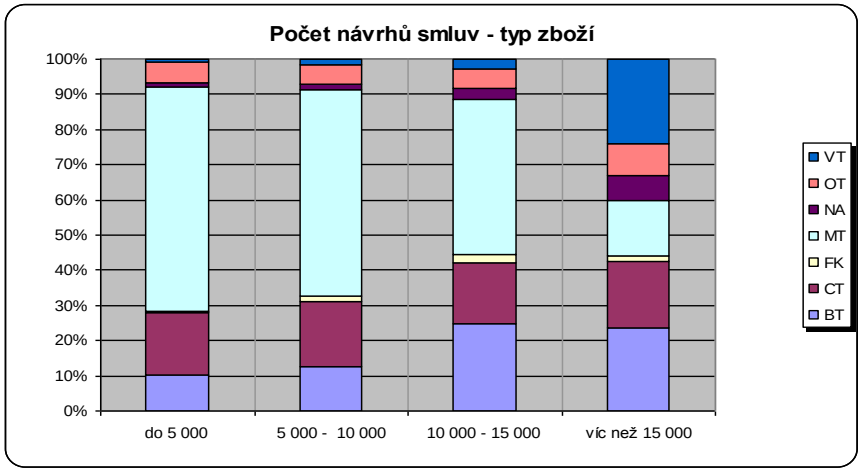
col%	do 5 000	5 000 - 10 000	10 000 - 15 000	víc než 15 000
BT	10,1%	12,7%	24,6%	23,7%
CT	17,8%	18,5%	17,5%	19,0%
FK	0,6%	1,5%	2,3%	1,5%
MT	63,8%	58,7%	44,4%	15,7%
NA	1,0%	1,6%	3,0%	7,2%
OT	5,8%	5,5%	5,6%	9,1%
VT	0,9%	1,5%	2,6%	23,8%

Explorace dat - vícerozměrná



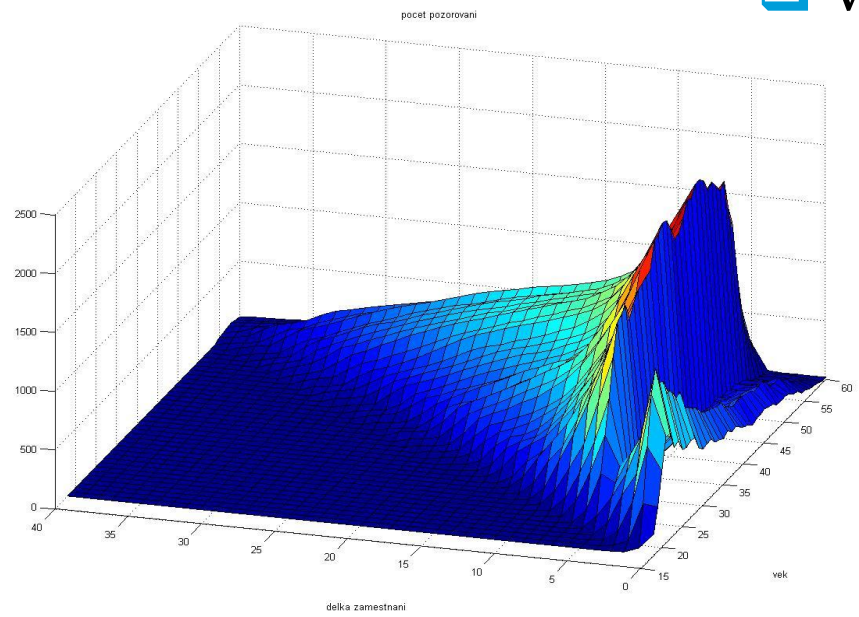






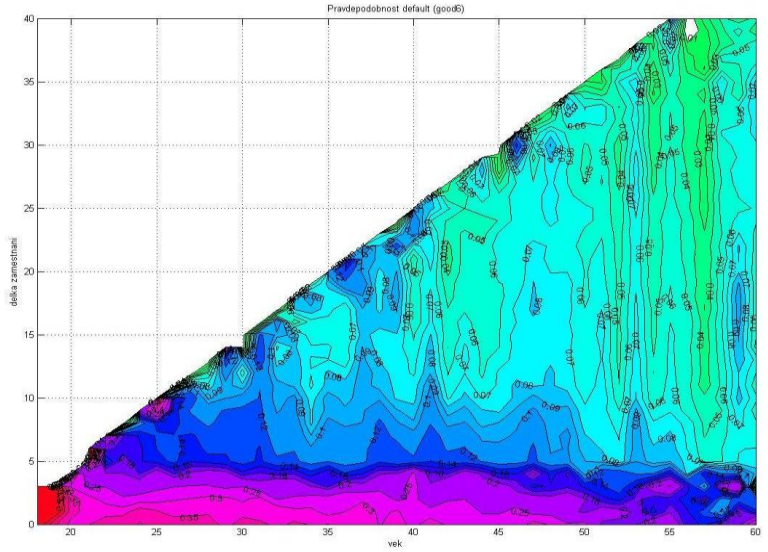
Explorace dat - vícerozměrná

- Věk vs. délka zaměstnání



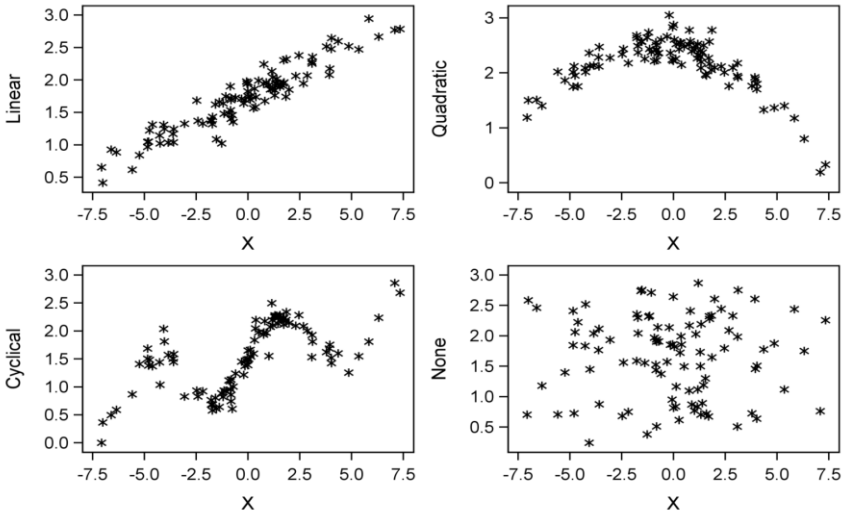
5 let …defaultní hodnota???

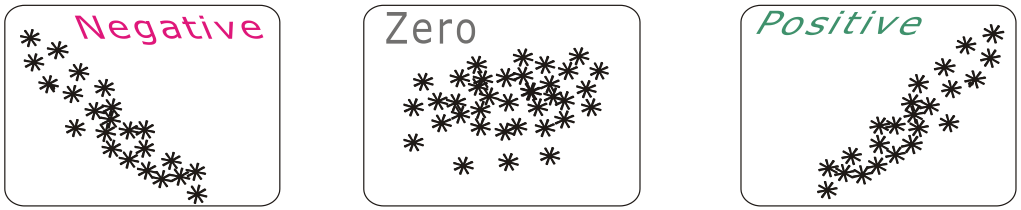
- Věk vs. délka zaměstnání vs. default

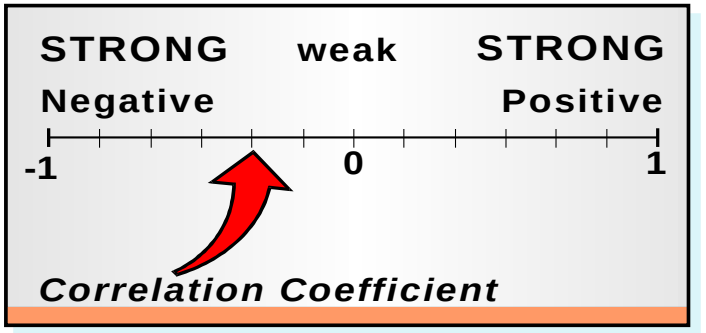


Pomocí tohoto typu grafu lze pohodlně zobrazit vztahy mezi 3 proměnnými.

Vztah mezi spojitými proměnnými – bodové grafy, korelace

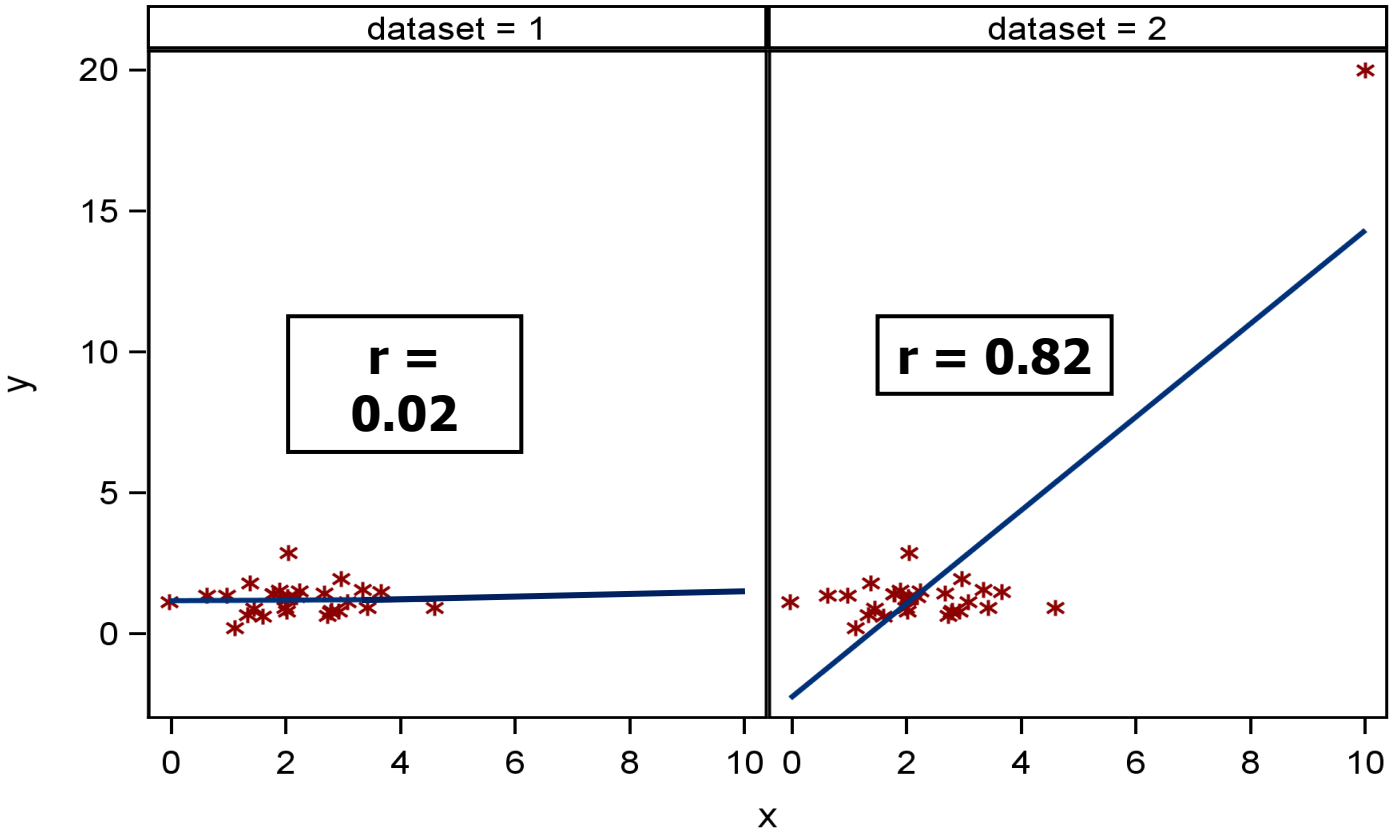






Extreme Data Values

Odlehlé (extrémní) hodnoty mohou zcela zkreslit výsledky analýzy.



Diskriminační síla proměnných pro prediktivní modely

Weight of evidence, information value

r ... number of levels (categories) of the categorical variable
g_i ... number of ″goods″ the in i-th category
b_i ... number of ″bads″ the in i-th category
$G := \Sigma g_i$ ... total number of ″goods″
$B := \Sigma b_i$ ... total number of ″bads″

Weight of evidence for the i-th category: $\boldsymbol{woe_i = ln (g_i / G) - ln (b_i / B)}$

Information value for the i-th category: $\boldsymbol{Inf\_val_i = [(g_i / G) - (b_i / B)] \cdot woe_i}$

Total information value for the corresponding variable: $\boldsymbol{Inf\_val = \Sigma\, inf\_val_i}$

Diskriminační síla proměnných

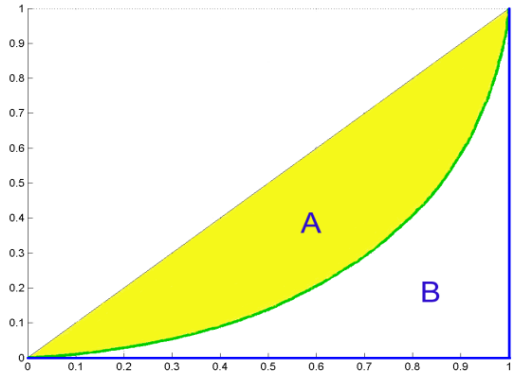
Incorporation Date											
Raw	RegVar	Percant	B	G	TOT	G/B Odds	%Good	%Bad	Bad Rate	WoE	IV
0 & NOI	inc_1	12%	139	952	1091	7	11%	19%	12,7%	-0,557	0,046116
1	inc_2	13%	133	1073	1206	8	12%	19%	11,0%	-0,394	0,023731
2-7	miss	42%	299	3601	3900	12	42%	42%	7,7%	0,007	2,04E-05
8-15	inc_3	22%	108	1942	2050	18	23%	15%	5,3%	0,408	0,030887
16+	inc_4	11%	39	1019	1058	26	12%	5%	3,7%	0,781	0,050288
Total			718	8587	9305	12			**7,7%**		**0,151**

- **<0.02 unpredictive**
- **0.02 – 0.1 weak**
- **0.1 – 0.3 medium**
- **0.3 – 0.5 strong**
- **> 0.5 too high …je třeba prověřit, pravděpodbně je něco špatně**

Diskriminační síla proměnných

- Lorenzova křivka, Giniho index

$$x = F_{m.BAD}(a)$$
$$y = F_{n.GOOD}(a),\ a \in [L, H].$$



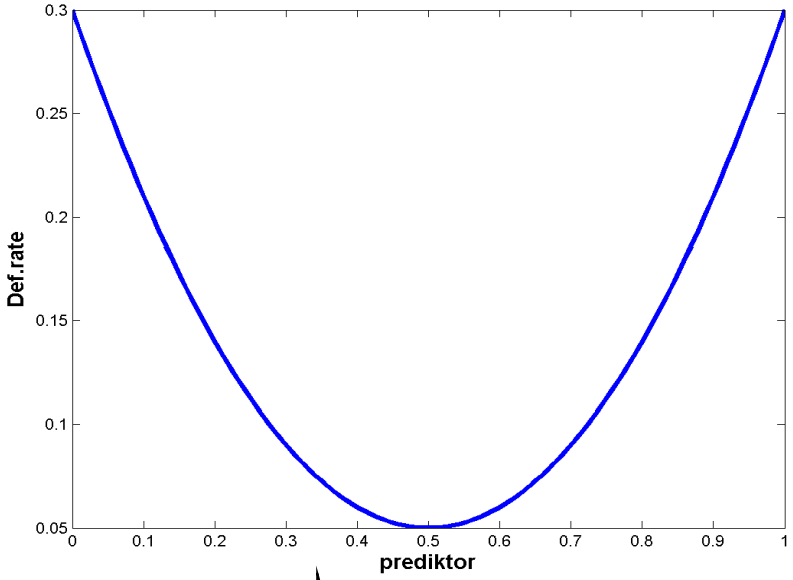
- <0.05 unpredictive
- 0.05 – 0.1 weak
- 0.1 – 0.2 medium
- 0.2 – 0.5 strong
- > 0.5 too high

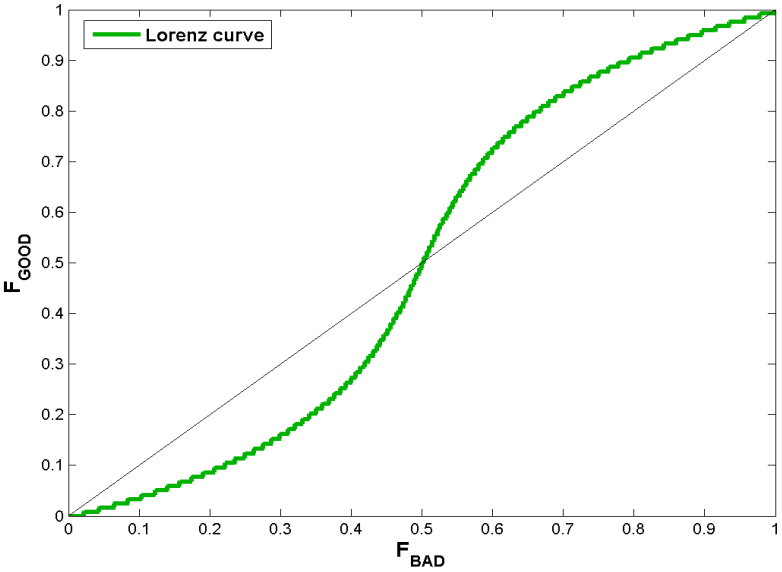
$$Gini = \frac{A}{A+B} = 2A$$

$$Gini = 1 - \sum_{k=2}^{n+m} (F_{m.BAD\,k} - F_{m.BAD\,k-1}) \cdot (F_{n.GOOD_{\,k}} + F_{n.GOOD\,k-1})$$

Diskriminační síla proměnných

- Lorenzova křivka …kontrola monotónnosti vysvětlované proměnné (def. rate) na dané vysvětlující proměnné





Kategorizace (WOE)

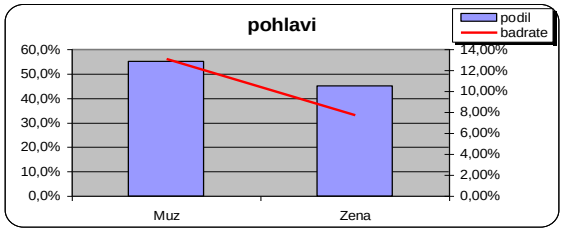
Diskriminační síla proměnných

- Je vhodné vytvořit souhrnný přehled (frekv. tabulky, histogramy, Gini, Info. Value,...) pro všechny uvažované proměnné.

pohlavi		Gini:	0,1401

Info.Value:	0,0828

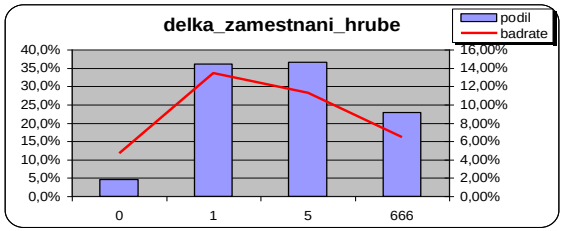
	pocet	podil	badrate
Muz	248 768	55,0%	13,08%
Zena	203 194	45,0%	7,69%
Total	451 962	100,0%	10,66%



delka_zamestnani_hrube		Gini:	0,1611

Info.Value:	0,1100

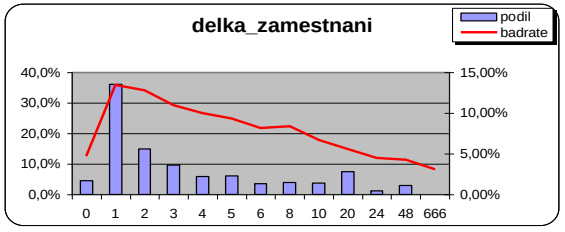
	pocet	podil	badrate
0	20 825	4,6%	4,69%
1	163 144	36,1%	13,43%
5	165 022	36,5%	11,29%
666	102 971	22,8%	6,45%
Total	451 962	100,0%	10,66%



delka_zamestnani_jemne		Gini:	0,1762

Info.Value:	0,1285

delka_zamestnani	pocet	podil	badrate
0	20 825	4,6%	4,69%
1	163 144	36,1%	13,43%
2	67 462	14,9%	12,80%
3	43 778	9,7%	10,97%
4	26 256	5,8%	10,01%
5	27 526	6,1%	9,32%
6	15 893	3,5%	8,16%
8	18 036	4,0%	8,39%
10	17 195	3,8%	6,72%
20	33 641	7,4%	5,60%
24	5 176	1,1%	4,48%
48	12 934	2,9%	4,28%
666	96	0,0%	3,13%
Total	451 962	100,0%	10,66%



The FREQ Procedure

- The FREQ procedure can do the following:
  - produce one-way to *n*-way frequency and crosstabulation (contingency) tables
  - compute chi-square tests for one-way to *n*-way tables and measures of association and agreement for contingency tables
  - automatically display the output in a report and save the output in a SAS data set
- General form of the FREQ procedure:

```
PROC FREQ DATA=SAS-data-set <option(s)>;
      TABLES variable(s) </ option(s)>;
RUN;
```

- A FREQ procedure with **no TABLES statement** generates one-way frequency tables for **all data set variables**.

The TABLES Statement

A one-way frequency table produces frequencies, cumulative frequencies, percentages, and cumulative percentages.

```
proc freq data=orion.sales;
   tables Gender Country;
run;
```

one-way frequency tables

The FREQ Procedure

Gender	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	68	41.21	68	41.21
M	97	58.79	165	100.00

Country	Frequency	Percent	Cumulative Frequency	Cumulative Percent
AU	63	38.18	63	38.18
US	102	61.82	165	100.00

The TABLES Statement

An *n*-way frequency table produces cell frequencies, cell percentages, cell percentages of row frequencies, and cell percentages of column frequencies, plus total frequency and percent.

```
proc freq data=orion.sales;
   tables Gender*Country;
run;
```

rows

columns

two-way frequency table

The FREQ Procedure

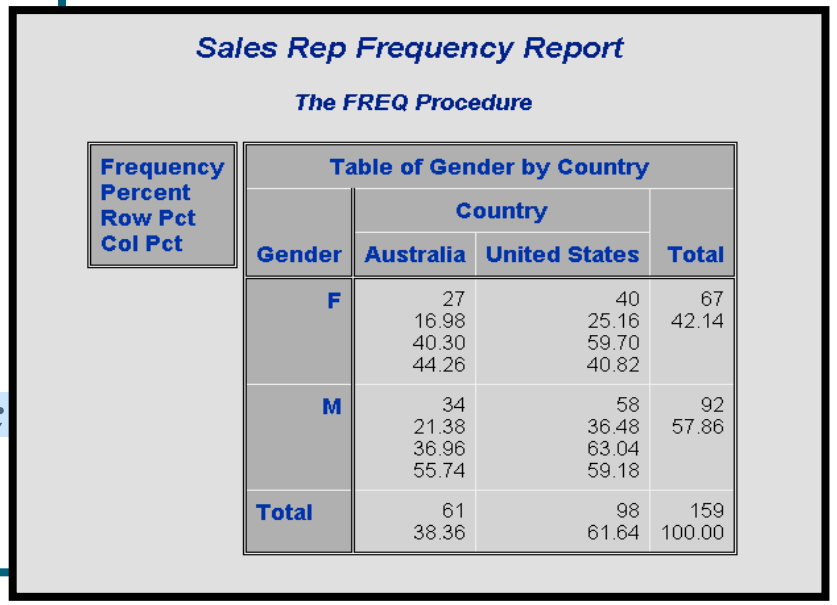
Table of Gender by Country

Gender / Country Frequency Percent Row Pct Col Pct	AU	US	Total
F	27 16.36 39.71 42.86	41 24.85 60.29 40.20	68 41.21
M	36 21.82 37.11 57.14	61 36.97 62.89 59.80	97 58.79
Total	63 38.18	102 61.82	165 100.00

Additional SAS Statements

- Additional statements can be added to enhance the report.

```
proc format;
   value $ctryfmt 'AU'='Australia'
                  'US'='United States';
run;

options nodate pageno=1;

ods html file='p112d01.html';
proc freq data=orion.sales;
   tables Gender*Country;
   where Job_Title contains 'Rep';
   format Country $ctryfmt.;
   title 'Sales Rep Frequency Report';
run;
ods html close;
```



Sales Rep Frequency Report

The FREQ Procedure

Frequency Percent Row Pct Col Pct	Table of Gender by Country		
	Country		
Gender	Australia	United States	Total
F	27 16.98 40.30 44.26	40 25.16 59.70 40.82	67 42.14
M	34 21.38 36.96 55.74	58 36.48 63.04 59.18	92 57.86
Total	61 38.36	98 61.64	159 100.00

Options to Suppress Display of Statistics

- Options can be placed in the TABLES statement after a forward slash to suppress the display of the default statistics.

Option	Description
NOCUM	suppresses the display of cumulative frequency and cumulative percentage.
NOPERCENT	suppresses the display of percentage, cumulative percentage, and total percentage.
NOFREQ	suppresses the display of the cell frequency and total frequency.
NOROW	suppresses the display of the row percentage.
NOCOL	suppresses the display of the column percentage.

Option	Description
LIST	displays *n*-way tables in list format.
CROSSLIST	displays *n*-way tables in column format.
FORMAT=	formats the frequencies in *n*-way tables.

LIST and CROSSLIST Options

Gender	Country	Frequency	Percent	Cumulative Frequency	Cumulative Percent
F	Australia	27	16.36	27	16.36
F	United States	41	24.85	68	41.21
M	Australia	36	21.82	104	63.03
M	United States	61	36.97	165	100.00

```
tables Gender*Country / list;
```

Table of Gender by Country

Gender	Country	Frequency	Percent	Row Percent	Column Percent
F	Australia	27	16.36	39.71	42.86
	United States	41	24.85	60.29	40.20
	Total	68	41.21	100.00	
M	Australia	36	21.82	37.11	57.14
	United States	61	36.97	62.89	59.80
	Total	97	58.79	100.00	
Total	Australia	63	38.18		100.00
	United States	102	61.82		100.00
	Total	165	100.00		

```
tables Gender*Country / crosslist;
```

PROC FREQ Statement Options

- Options can also be placed in the PROC FREQ statement.

Option	Description
NLEVELS	displays a table that provides the number of levels for each variable named in the TABLES statement.
PAGE	displays only one table per page.
COMPRESS	begins the display of the next one-way frequency table on the same page as the preceding one-way table ifthere is enough space to begin the table.

```
proc freq data=orion.sales nlevels;
   tables Gender Country Employee_ID;
run;
```

The FREQ Procedure

Number of Variable Levels

Variable	Levels
Gender	2
Country	2
Employee_ID	165

Output Data Sets

- PROC FREQ produces output data sets using two different methods.
  - The TABLES statement with an OUT= option is used to create a data set with **frequencies and percentages**.

TABLES *variables* **/ OUT=***SAS-data-set* <*options*>**;**

  - The OUTPUT statement with an OUT= option is used to create a data set with **specified statistics** such as the chi-square statistic.

OUTPUT OUT=*SAS-data-set* <*options*>**;**

The MEANS Procedure

- The *MEANS procedure* provides data summarization tools to compute descriptive statistics for variables across all observations and within groups of observations.
- General form of the MEANS procedure:

```
PROC MEANS DATA=SAS-data-set <statistic(s)> <option(s)>;
      VAR analysis-variable(s);
      CLASS classification-variable(s);
RUN;
```

- By default, the MEANS procedure reports the number of nonmissing observations, the mean, the standard deviation, the minimum value, and the maximum value of all numeric variables.

```
proc means data=orion.sales;
run;
```

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
Employee_ID	165	120713.90	450.0866939	120102.00	121145.00
Salary	165	31160.12	20082.67	22710.00	243190.00
Birth_Date	165	3622.58	5456.29	-5842.00	10490.00
Hire_Date	165	12054.28	4619.94	5114.00	17167.00

The VAR Statement

The *VAR statement* identifies the analysis variables and their order in the results.

```
proc means data=orion.sales;
   var Salary;
run;
```

The MEANS Procedure

Analysis Variable : Salary

N	Mean	Std Dev	Minimum	Maximum
165	31160.12	20082.67	22710.00	243190.00

The CLASS Statement

- The *CLASS statement* identifies variables whose values define subgroups for the analysis.

```
proc means data=orion.sales;
   var Salary;
   class Gender Country;
run;
```

The MEANS Procedure

Analysis Variable : Salary

Gender	Country	N Obs	N	Mean	Std Dev	Minimum	Maximum
F	AU	27	27	27702.41	1728.23	25185.00	30890.00
	US	41	41	29460.98	8847.03	25390.00	83505.00
M	AU	36	36	32001.39	16592.45	25745.00	108255.00
	US	61	61	33336.15	29592.69	22710.00	243190.00

The CLASS Statement

```
proc means data=orion.sales;
   var Salary;
   class Gender Country;
run;
```

classification variables

analysis variable

The MEANS Procedure

Analysis Variable : Salary

Gender	Country	N Obs	N	Mean	Std Dev	Minimum	Maximum
F	AU	27	27	27702.41	1728.23	25185.00	30890.00
	US	41	41	29460.98	3847.03	25200.00	83505.00
M	AU	36	36	32001.3			00
	US	61	61	33336.15	29592.69	22710.00	243190.00

statistics for analysis variable

The CLASS statement adds the N Obs column, which is the number of observations for each unique combination of the class variables.

PROC MEANS Statistics

- The statistics to compute and the order to display them can be specified in the PROC MEANS statement.

```
proc means data=orion.sales sum mean range;
   var Salary;
   class Country;
run;
```

The MEANS Procedure

Analysis Variable : Salary

Country	N Obs	Sum	Mean	Range
AU	63	1900015.00	30158.97	83070.00
US	102	3241405.00	31778.48	220480.00

- další dostupné statistiky:

Descriptive Statistic Keywords				
CLM	CSS	CV	LCLM	MAX
MEAN	MIN	MODE	N	NMISS
KURTOSIS	RANGE	SKEWNESS	STDDEV	STDERR
SUM	SUMWGT	UCLM	USS	VAR
Quantile Statistic Keywords				
MEDIAN | P50	P1	P5	P10	Q1 | P25
Q3 | P75	P90	P95	P99	QRANGE
Hypothesis Testing Keywords				
PROBT	T			

PROC MEANS Statement Options

- Options can also be placed in the PROC MEANS statement.

Option	Description
MAXDEC=	specifies the number of decimal places to use in printing the statistics.
FW=	specifies the field width to use in displaying the statistics.
NONOBS	suppresses reporting the total number of observations for each unique combination of the class variables.

```
proc means data=orion.sales maxdec=0;
```

Analysis Variable : Salary

Country	N Obs	N	Mean	Std Dev	Minimum	Maximum
AU	63	63	30159	12699	25185	108255
US	102	102	31778	23556	22710	243190

```
proc means data=orion.sales maxdec=1;
```

Analysis Variable : Salary

Country	N Obs	N	Mean	Std Dev	Minimum	Maximum
AU	63	63	30159.0	12699.1	25185.0	108255.0
US	102	102	31778.5	23555.8	22710.0	243190.0

Output Data Sets

- PROC MEANS produces output data sets using the following method:

OUTPUT OUT=*SAS-data-set* <*options*>**;**

- The output data set contains the following variables:
  - BY variables
  - class variables
  - the automatic variables **_TYPE_** and **_FREQ_**
  - the variables requested in the OUTPUT statement

OUTPUT Statement OUT= Option

```
proc means data=orion.sales sum mean range;
   var Salary;
   class Gender Country;
   output out=work.means1;
run;

proc print data=work.means1;
run;
```

The statistics in the PROC statement impact only the MEANS report, not the data set.

Obs	Gender	Country	_TYPE_	_FREQ_	_STAT_	Salary
1			0	165	N	165.00
2			0	165	MIN	22710.00
3			0	165	MAX	243190.00
4			0	165	MEAN	31160.12
5			0	165	STD	20082.67
6		AU	1	63	N	63.00
7					MIN	25185.00
8					MAX	108255.00
9		AU	1	63	MEAN	30158.97
10		AU	1	63	STD	12699.14
11		US	1	102	N	102.00
12		US	1	102	MIN	22710.00
13		US	1	102	MAX	243190.00
14		US	1	102	MEAN	31778.48
15		US	1	102	STD	23555.84
16	F		2	68	N	68.00
17	F		2	68	MIN	25185.00
18	F		2	68	MAX	83505.00
19	F		2	68	MEAN	28762.72
20	F		2	68	STD	6974.15

default statistics

OUTPUT Statement OUT= Option

- The OUTPUT statement can also do the following:
  - specify the statistics for the output data set
  - select and name variables

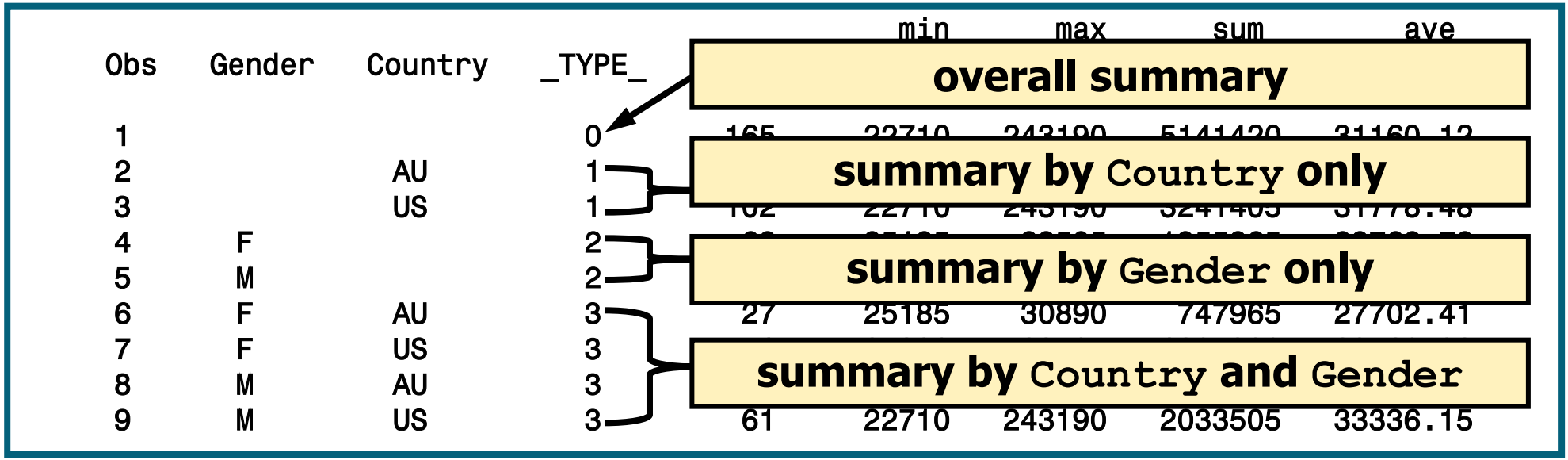
```
proc means data=orion.sales noprint;
   var Salary;
   class Gender Country;
   output out=work.means2
          min=minSalary max=maxSalary
          sum=sumSalary mean=aveSalary;
run;
proc print data=work.means2;run;
```

- The NOPRINT option suppresses the display of all output.

Obs	Gender	Country	_TYPE_	_FREQ_	min Salary	max Salary	sum Salary	ave Salary
1			0	165	22710	243190	5141420	31160.12
2		AU	1	63	25185	108255	1900015	30158.97
3		US	1	102	22710	243190	3241405	31778.48
4	F		2	68	25185	83505	1955865	28762.72
5	M		2	97	22710	243190	3185555	32840.77
6	F	AU	3	27	25185	30890	747965	27702.41
7	F	US	3	41	25390	83505	1207900	29460.98
8	M	AU	3	36	25745	108255	1152050	32001.39
9	M	US	3	61	22710	243190	2033505	33336.15

OUTPUT Statement OUT= Option

- **_TYPE_** is a numeric variable that shows which combination of class variables produced the summary statistics in that observation.



OUTPUT Statement OUT= Option

Obs	Gender	Country	_TYPE_	_FREQ_	min Salary	max Salary	sum Salary	ave Salary
1			0	165	22710	243190	5141420	31160.12
2		AU	1	63	25185	108255	1900015	30158.97
3		US	1	102	22710	243190	3241405	31778.48
4	F		2	68	25185	83505	1955865	28762.72
5	M		2	97	22710	243190	3185555	32840.77
6	F	AU	3	27	25185	30890	747965	27702.41
7	F	US	3	41	25390	83505	1207900	29460.98
8	M	AU	3	36	25745	108255	1152050	32001.39
9	M	US	3	61	22710	243190	2033505	33336.15

TYPE	Type of Summary	_FREQ_
0	overall summary	165
1	summary by **Country** only	63 AU + 102 AU = 165
2	summary by **Gender** only	68 F + 97 M = 165
3	summary by **Country** and **Gender**	27 F AU + 41 F US + 36 M AU + 61 M US = 165

OUTPUT Statement OUT= Option

- Options can be added to the PROC MEANS statement to control the output data set.

Option	Description
NWAY	specifies that the output data set contain only statistics for the observations with the highest **_TYPE_** value.
DESCENDTYPES	orders the output data set by descending **_TYPE_** value.
CHARTYPE	specifies that the **_TYPE_** variable in the output data set is a character representation of the binary value of **_TYPE_**.

without options

Obs	Gender	Country	_TYPE_	_FREQ_	min Salary	max Salary	sum Salary	ave Salary
1			0	165	22710	243190	5141420	31160.12
2		AU	1	63	25185	108255	1900015	30158.97
3		US	1	102	22710	243190	3241405	31778.48
4	F		2	68	25185	83505	1955865	28762.72
5	M		2	97	22710	243190	3185555	32840.77
6	F	AU	3	27	25185	30890	747965	27702.41
7	F	US	3	41	25390	83505	1207900	29460.98
8	M	AU	3	36	25745	108255	1152050	32001.39
9	M	US	3	61	22710	243190	2033505	33336.15

OUTPUT Statement OUT= Option

with NWAY

Obs	Gender	Country	_TYPE_	_FREQ_	min Salary	max Salary	sum Salary	ave Salary
1	F	AU	3	27	25185	30890	747965	27702.41
2	F	US	3	41	25390	83505	1207900	29460.98
3	M	AU	3	36	25745	108255	1152050	32001.39
4	M	US	3	61	22710	243190	2033505	33336.15

with DESCENDTYPES

Obs	Gender	Country	_TYPE_	_FREQ_	min Salary	max Salary	sum Salary	ave Salary
1	F	AU	3	27	25185	30890	747965	27702.41
2	F	US	3	41	25390	83505	1207900	29460.98
3	M	AU	3	36	25745	108255	1152050	32001.39
4	M	US	3	61	22710	243190	2033505	33336.15
5	F		2	68	25185	83505	1955865	28762.72
6	M		2	97	22710	243190	3185555	32840.77
7		AU	1	63	25185	108255	1900015	30158.97
8		US	1	102	22710	243190	3241405	31778.48
9			0	165	22710	243190	5141420	31160.12

with CHARTYPE

Obs	Gender	Country	_TYPE_	_FREQ_	min Salary	max Salary	sum Salary	ave Salary
1			00	165	22710	243190	5141420	31160.12
2		AU	01	63	25185	108255	1900015	30158.97
3		US	01	102	22710	243190	3241405	31778.48
4	F		10	68	25185	83505	1955865	28762.72
5	M		10	97	22710	243190	3185555	32840.77
6	F	AU	11	27	25185	30890	747965	27702.41
7	F	US	11	41	25390	83505	1207900	29460.98
8	M	AU	11	36	25745	108255	1152050	32001.39
9	M	US	11	61	22710	243190	2033505	33336.15

The SUMMARY Procedure

- The SUMMARY procedure provides data summarization tools to compute descriptive statistics for variables across all observations and within groups of observations.

General form of the SUMMARY procedure:

```
PROC SUMMARY DATA=SAS-data-set <statistic(s)>
                                <option(s)>;
     VAR analysis-variable(s);
     CLASS classification-variable(s);
RUN;
```

The SUMMARY Procedure

- The SUMMARY procedure uses the same syntax as the MEANS procedure.
- The only differences to the two procedures are the following:

PROC MEANS	PROC SUMMARY
The PRINT option is set by default, which displays output.	The NOPRINT option is set by default, which displays no output.
Omitting the VAR statement analyzes all the numeric variables.	Omitting the VAR statement produces a simple count of observations.

The TABULATE Procedure

- The TABULATE procedure displays descriptive statistics in tabular format.

General form of the TABULATE procedure:

```
PROC TABULATE DATA=SAS-data-set <options>;
      CLASS classification-variable(s);
      VAR analysis-variable(s);
      TABLE  page-expression,
             row-expression,
             column-expression </ option(s)>;
RUN;
```

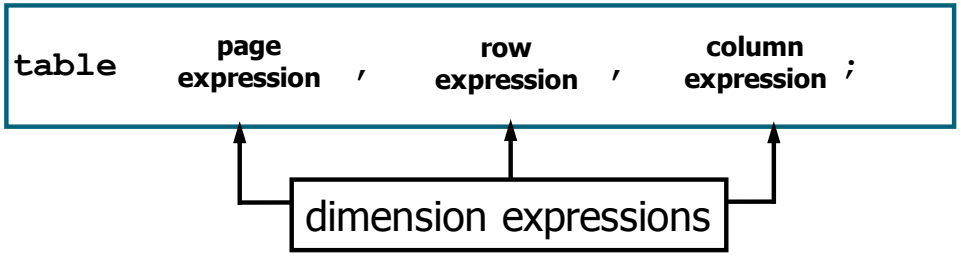
Dimensional Tables

- The TABULATE procedure produces one-, two-, or three-dimensional tables.

	page dimension	row dimension	column dimension
one-dimensional			✓
two-dimensional		✓	✓
three-dimensional	✓	✓	✓

The TABLE Statement

- The TABLE statement describes the structure of the table.



  - Commas separate the dimension expressions.
  - Every variable that is part of a dimension expression must be specified as a classification variable (CLASS statement) or an analysis variable (VAR statement).

```
table page expression , row expression , column expression ;
```

- Příklady:

```
table Country;
```

```
table Gender , Country;
```

```
table Job_Title , Gender , Country;
```

The CLASS Statement

- The CLASS statement identifies variables to be used as classification, or grouping, variables.
- General form of the CLASS statement:

CLASS *classification-variable(s)*;

- N, the number of nonmissing values, is the default statistic for classification variables.
- Examples of classification variables:

  `Job_Title`, **`Gender`**, and **`Country`**

The VAR Statement

- The VAR statement identifies the numeric variables for which statistics are calculated.
- General form of the VAR statement:

VAR *analysis-variable(s)*;

- SUM is the default statistic for analysis variables.
- Examples of analysis variables:

  `Salary` and **`Bonus`**

One/two-Dimensional Table

```
proc tabulate data=orion.sales;
   class Country;
   table Country;
run;
```

Country	
AU	US
N	N
63.00	102.00

```
proc tabulate data=orion.sales;
   class Gender Country;
   table Gender, Country;
run;
```

	Country	
	AU	US
	N	N
Gender		
F	27.00	41.00
M	36.00	61.00

Three-Dimensional Table

```
proc tabulate data=orion.sales;
   class Job_Title Gender Country;
   table Job_Title, Gender, Country;
run;
```

Job_Title Sales Rep. I

	Country	
	AU	US
Gender		
F		
M		

Job_Title Sales Rep. II

	Country	
	AU	US
	N	N
Gender		
F	10.00	14.00
M	8.00	14.00

Dimension Expression

- Elements that can be used in a dimension expression:
  - classification variables
  - analysis variables
  - the universal class variable ALL
  - keywords for statistics

- Operators that can be used in a dimension expression:
  - blank, which concatenates table information
  - asterisk *, which crosses table information
  - parentheses (), which group elements

Dimension Expression

```
proc tabulate data=orion.sales;
   class Gender Country;
   var Salary;
   table Gender all, Country*Salary;
run;
```

	Country	
	AU	US
	Salary	Salary
	Sum	Sum
Gender		
F	747965.00	1207900.00
M	1152050.00	2033505.00
All	1900015.00	3241405.00

PROC TABULATE Statistics

Descriptive Statistic Keywords				
	CSS	CV	LCLM	MAX
MEAN	MIN	MODE	N	NMISS
KURTOSIS	RANGE	SKEWNESS	STDDEV	STDERR
SUM	SUMWGT	UCLM	USS	VAR
PCTN	REPPCTN	PAGEPCTN	ROWPCTN	COLPCTN
PCTSUM	REPPCTSUM	PAGEPCTSUM	ROWPCTSUM	COLPCTSUM

Quantile Statistic Keywords				
MEDIAN | P50	P1	P5	P10	Q1 | P25
Q3 | P75	P90	P95	P99	QRANGE

Hypothesis Testing Keywords				
PROBT	T			

PROC TABULATE Statistics

```
proc tabulate data=orion.sales;
   class Gender Country;
   var Salary;
   table Gender all, Country*Salary*(min max);
run;
```

	Country			
	AU		US	
	Salary		Salary	
	Min	Max	Min	Max
Gender				
F	25185.00	30890.00	25390.00	83505.00
M	25745.00	108255.00	22710.00	243190.00
All	25185.00	108255.00	22710.00	243190.00

Additional SAS Statements

- Additional statements can be added to enhance the report.

```
proc format;
   value $ctryfmt 'AU'='Australia'
                  'US'='United States';
run;

options nodate pageno=1;

ods html file='p112d08.html';
proc tabulate data=orion.sales;
   class Gender Country;
   var Salary;
   table Gender all, Country*Salary*(min max);
   where Job_Title contains 'Rep';
   label Salary='Annual Salary';
   format Country $ctryfmt.;
   title 'Sales Rep Tabular Report';
run;
ods html close;
```

Sales Rep Tabular Report

	Country			
	Australia		United States	
	Annual Salary		Annual Salary	
	Min	Max	Min	Max
Gender				
F	25185.00	30890.00	25390.00	32985.00
M	25745.00	36605.00	22710.00	35990.00
All	25185.00	36605.00	22710.00	35990.00

Output Data Sets

- PROC TABULATE produces output data sets using the following method:

```
PROC TABULATE DATA=SAS-data-set
                  OUT=SAS-data-set <options>;
```

- The output data set contains the following variables:
  - BY variables
  - class variables
  - the automatic variables **_TYPE_**, **_PAGE_**, and **_TABLE_**
  - calculated statistics

PROC Statement OUT= Option

```
proc tabulate data=orion.sales
              out=work.tabulate;
   where Job_Title contains 'Rep';
   class Job_Title Gender Country;
   table Country;
   table Gender, Country;
   table Job_Title, Gender, Country;
run;

proc print data=work.tabulate;
run;
```

Obs	Job_Title	Gender	Country	_TYPE_	_PAGE_	_TABLE_	N
1			AU	001	1	1	61
2			US	001	1	1	98
3		F	AU	011	1	2	27
4		F	US	011	1	2	40
5		M	AU	011	1	2	34
6		M	US	011	1	2	58
7	Sales Rep. I	F	AU	111	1	3	8
8	Sales Rep. I	F	US	111	1	3	13
9	Sales Rep. I	M	AU	111	1	3	13
10	Sales Rep. I	M	US	111	1	3	29
11	Sales Rep. II	F	AU	111	2	3	10
12	Sales Rep. II	F	US	111	2	3	14
13	Sales Rep. II	M	AU	111	2	3	8
14	Sales Rep. II	M	US	111	2	3	14
15	Sales Rep. III	F	AU	111	3	3	7
16	Sales Rep. III	F	US	111	3	3	8
17	Sales Rep. III	M	AU	111	3	3	10
18	Sales Rep. III	M	US	111	3	3	9

PROC Statement OUT= Option

- **_TYPE_** is a character variable that shows which combination of class variables produced the summary statistics in that observation.

- Partial PROC PRINT Output

Obs	Job_Title	Gender	Country	_TYPE_	_PAGE_	_TABLE_	N
1			AU	001	1	1	61
2			US	001	1	1	98
3		F	AU	011	1	2	27
4		F	US	011			
5		M	AU	011			
6		M	US	011			

0 for **Job_Title**, 1 for **Gender**, and 1 for **Country**

PROC Statement OUT= Option

- **_PAGE_** is a numeric variable that shows the logical page number that contains that observation.
- Partial PROC PRINT Output

Obs	Job_Title	Gender	Country	_TYPE_	_PAGE_	_TABLE_	N
7	Sales Rep. I	F	AU	111	1		
8	Sales Rep. I	F	US	111	1		
9	Sales Rep. I	M	AU	111	1		
10	Sales Rep. I	M	US	111	1	3	29
11	Sales Rep. II	F	AU	111	2		
12	Sales Rep. II	F	US	111	2		
13	Sales Rep. II	M	AU	111	2		
14	Sales Rep. II	M	US	111	2	3	14
15	Sales Rep. III	F	AU	111	3		
16	Sales Rep. III	F	US	111	3		
17	Sales Rep. III	M	AU	111	3		
18	Sales Rep. III	M	US	111	3		

Page 1 for Sales Rep. I

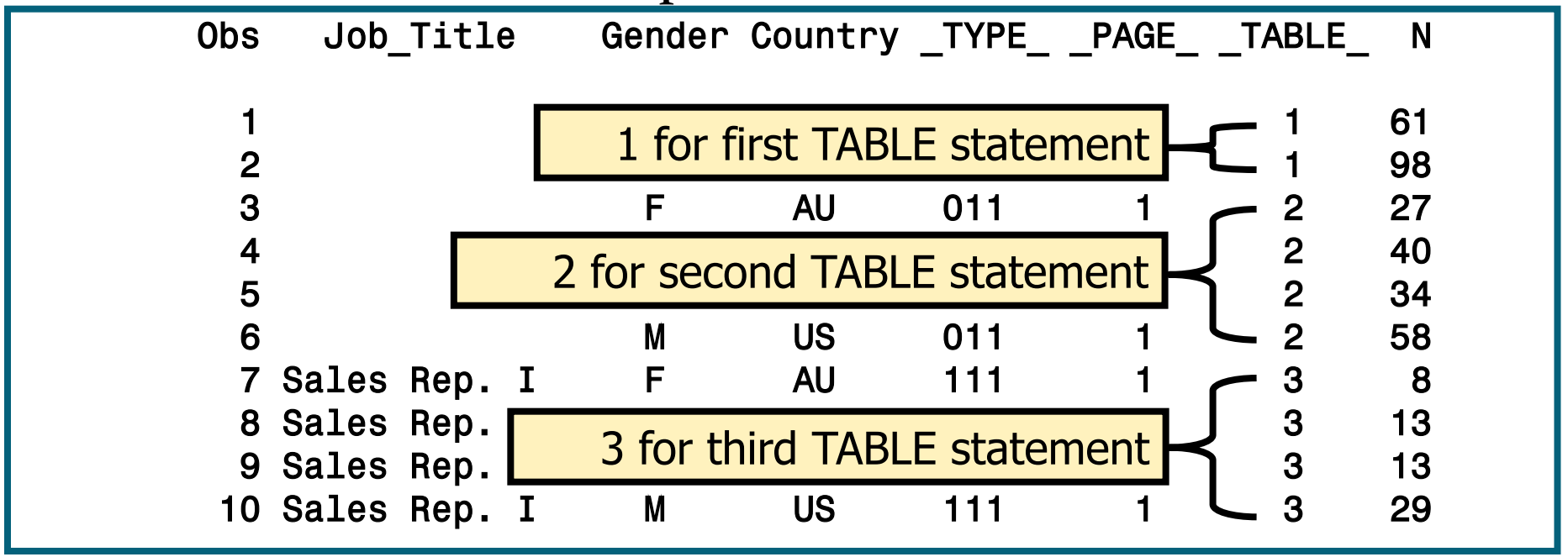
Page 2 for Sales Rep. II

Page 3 for Sales Rep. III

PROC Statement OUT= Option

- **_TABLE_** is a numeric variable that shows the number of the TABLE statement that contains that observation.

- Partial PROC PRINT Output



Více o PROC TABULATE:

- In the SUGI 28 proceedings:
  - *"The Simplicity and Power of the TABULATE Procedure"*, by Dan Bruns

  http://www2.sas.com/proceedings/sugi28/197-28.pdf
- Online (from the SUGI 27 proceedings):
  - *"Anyone Can Learn PROC TABULATE"*, by Lauren Haworth, http://www2.sas.com/proceedings/sugi27/p060-27.pdf

The UNIVARIATE Procedure

- The UNIVARIATE procedure produces summary reports that display descriptive statistics.

- General form of the UNIVARIATE procedure:

```
PROC UNIVARIATE DATA=SAS-data-set;
     VAR variable(s);
RUN;
```

- The VAR statement specifies the analysis variables and their order in the results.

The UNIVARIATE Procedure

The following PROC UNIVARIATE step shows default descriptive statistics for **Salary**.

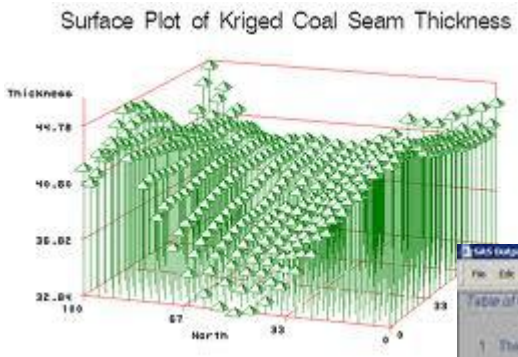
```
proc univariate data=orion.nonsales;
   var Salary;
run;
```

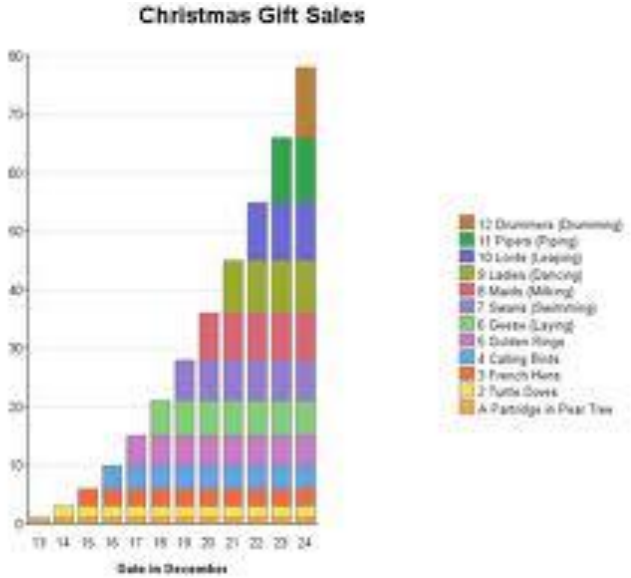
- Without the VAR statement, SAS will analyze all numeric variables.

The UNIVARIATE Procedure

- The UNIVARIATE procedure can produce the following sections of output:
  - Moments
  - Basic Statistical Measures
  - Tests for Locations
  - Quantiles
  - Extreme Observations
  - Missing Values

8. Vizualizace dat, SAS/Graph



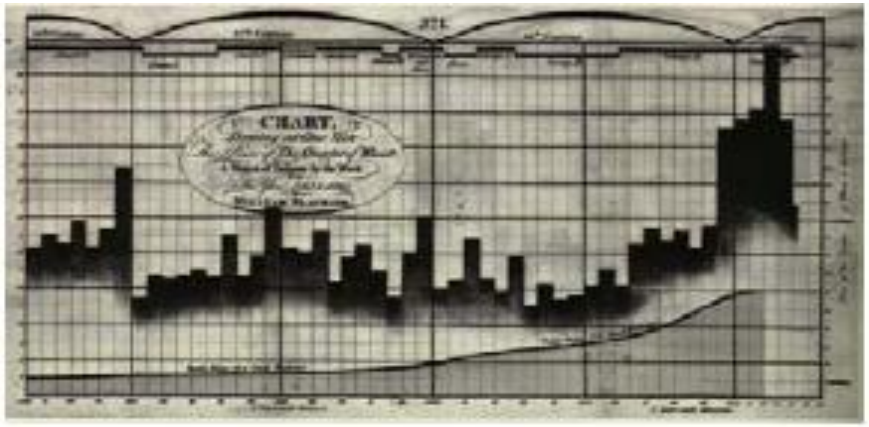


Vizualizace – zdroje

- Na prvním místě se obvykle citují knihy prof. Tufteho, např. Tufte E.R. (1983) The Visual Display of Quantitative Information, Graphic Press, Chesire, Conn.
- Weby o vizualizaci, např.
  - http://www.datavis.ca/gallery/ - galerie s poučným výkladem a příklady i nezdařených či lživých grafů
  - http://www.agocg.ac.uk/ - John Lansdown (1992) Aspects of Design in Computer Graphics: Some Notes – http://www.agocg.ac.uk/train/hitch/hitch.htm
- Jiné weby, např. stránky různých vizualizačních programů a organizací
  - http://www.cybergeography.org/atlas/atlas.html nebo http://miner3d.com/products/gallery.html

Vizualizace – historie

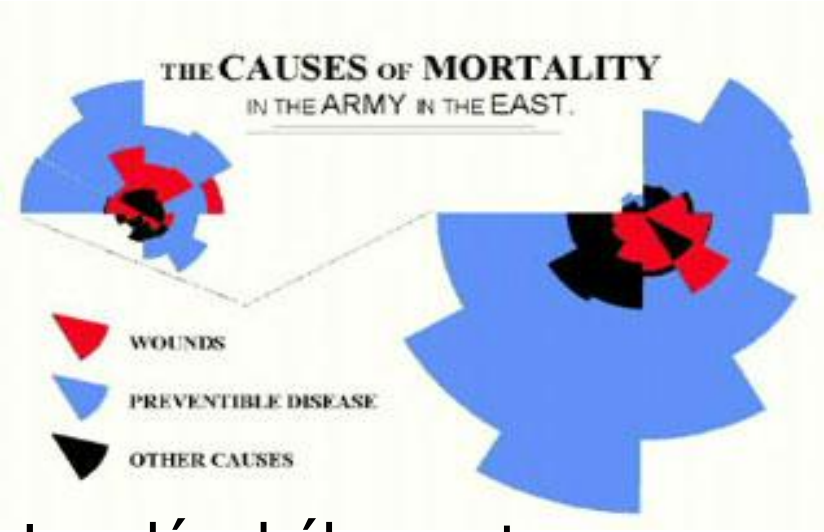
- William Playfair, 1786: první publikovaná prezentační grafika



- Dr. John Snow, 1845: epidemie cholery v Londýně

Vizualizace – historie

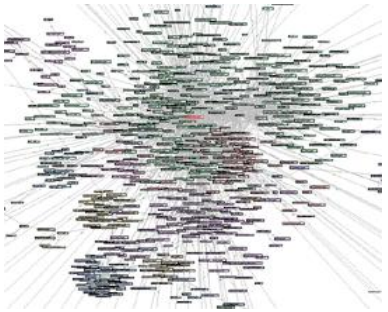
- Florence Nightingale, 1858: důvody úmrtí v průběhu Krymské války (1853-1856)



- Harry Beck, 1931: schéma Londýnského metra

Vizualizace – investigativní analýza

- http://www.i2inc.com/



Law Enforcement

» Counterterrorism
» Narcotics investigations
» Organized crime
» Intelligence analysis
» Fraud
» Missing persons
» Major investigations
» Counterfeiting
» Immigration control
» Major event security
» Money laundering
» Gang investigations

Government

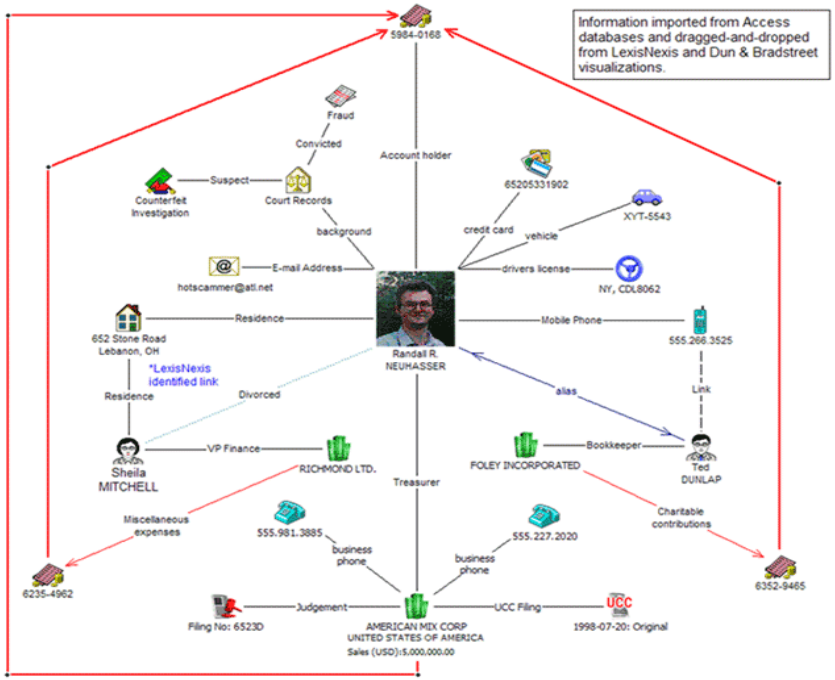
» Criminal prosecutions
» National security
» Military intelligence
» Embassy security
» Postal inspection and fraud
» Prison investigations
» Park and wildlife services
» Antitrust investigations
» Tax fraud investigations
» Customs investigations

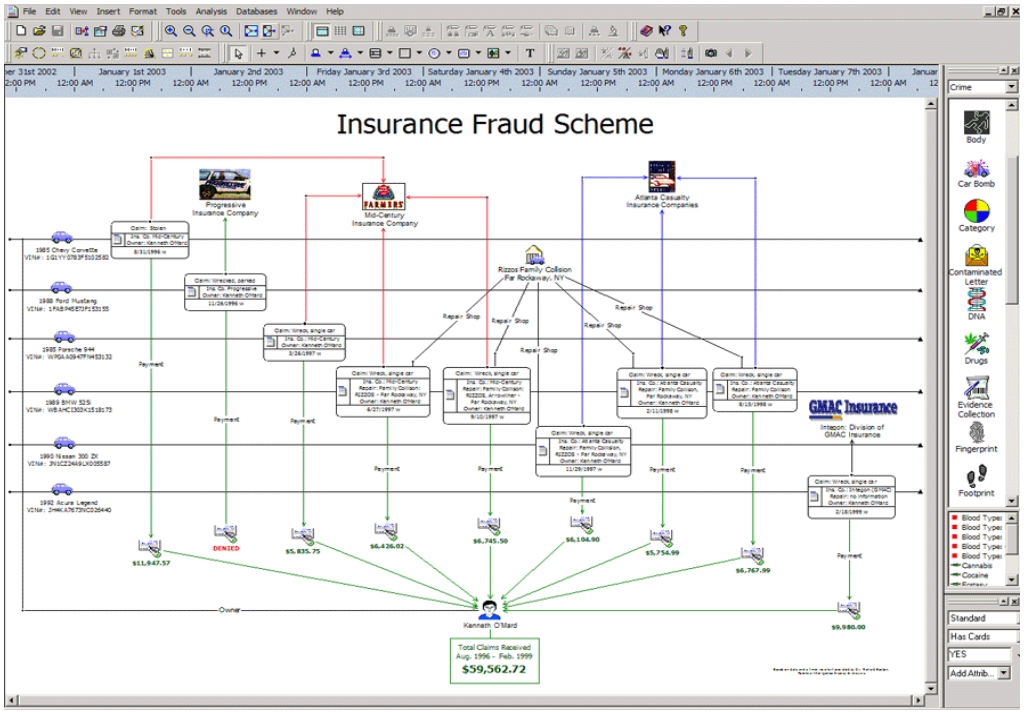
Commercial

» Forensic accounting
» Money laundering
» Insider trading violations
» Corporate security
» Anti-pirating investigations
» Entertainment copyright violations
» Competitive intelligence
» Civil lawsuits
» Fraud:
  » Credit card
  » Insurance
  » Retail
  » Health care
  » Commercial
  » Telephone

Vizualizace – investigativní analýza

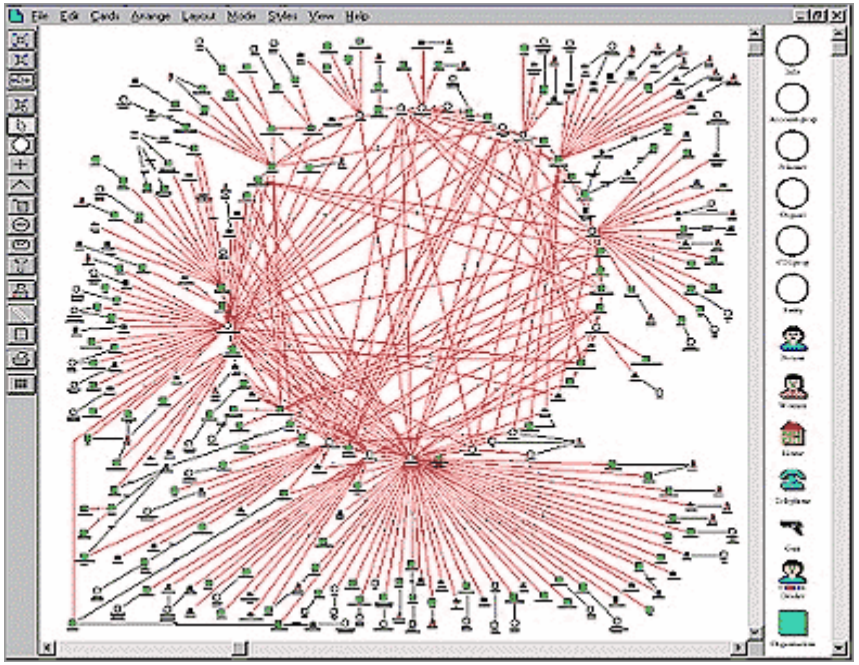
- osobní kontakty, pojistné podvody

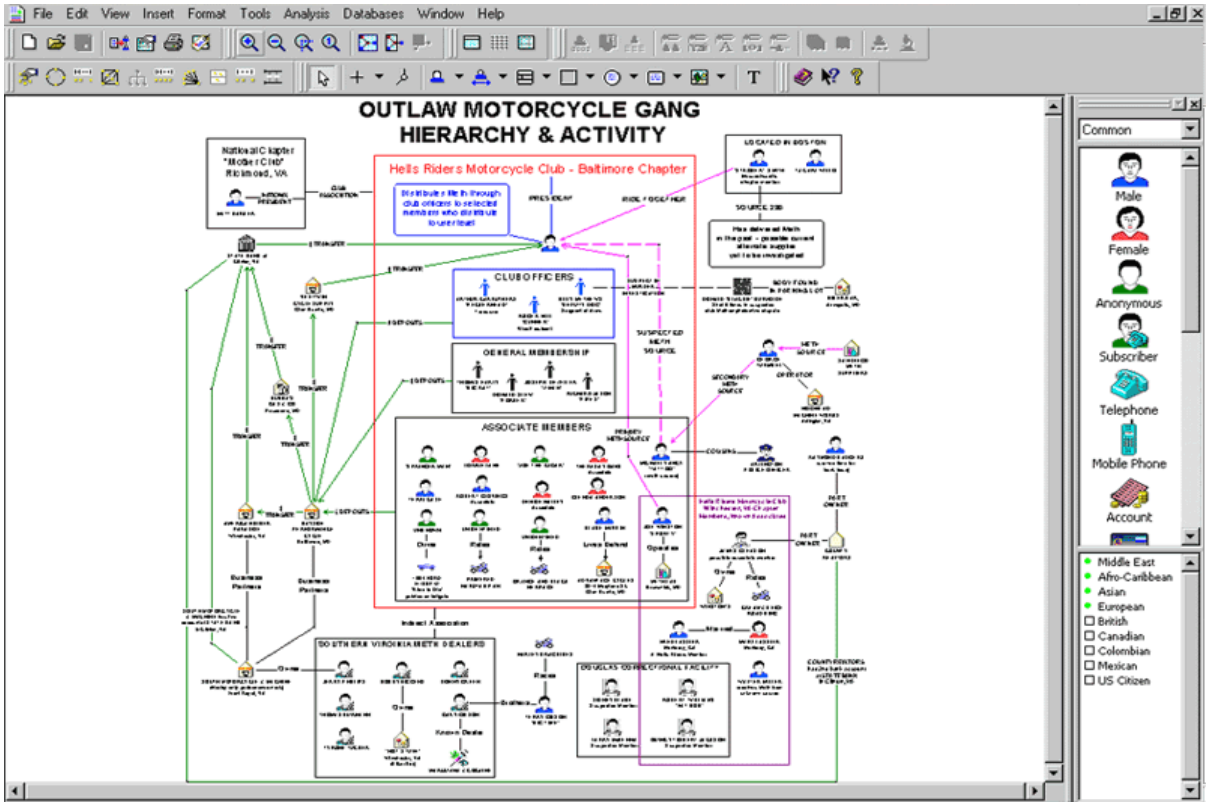




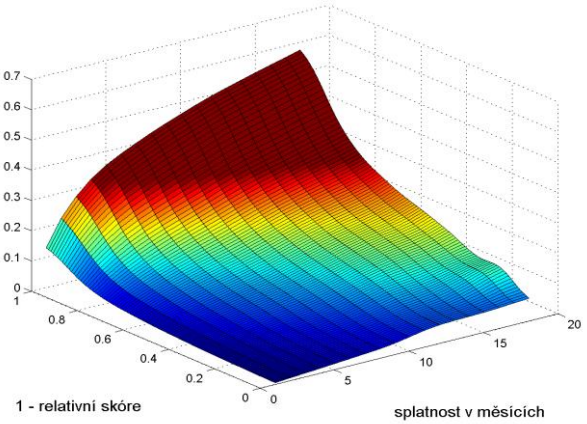
Vizualizace – investigativní analýza

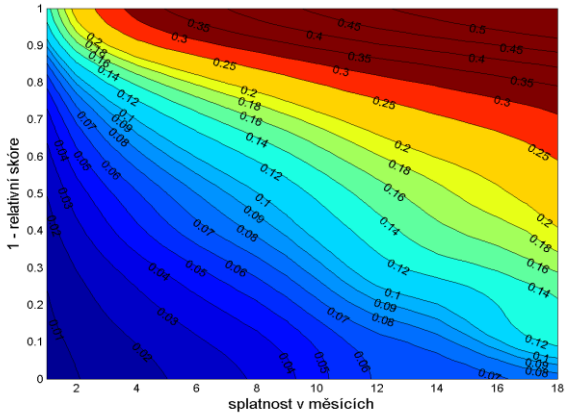
- Praní špinavých peněz, kriminální gangy

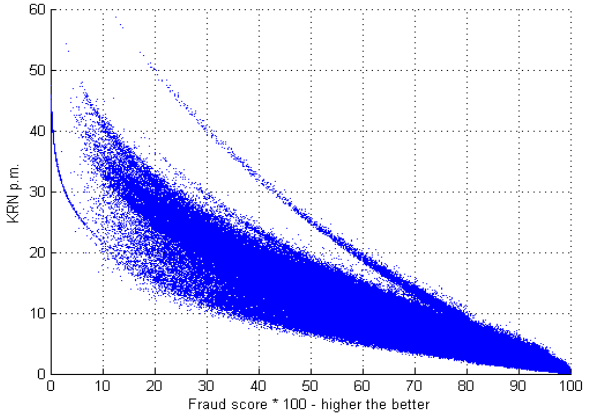


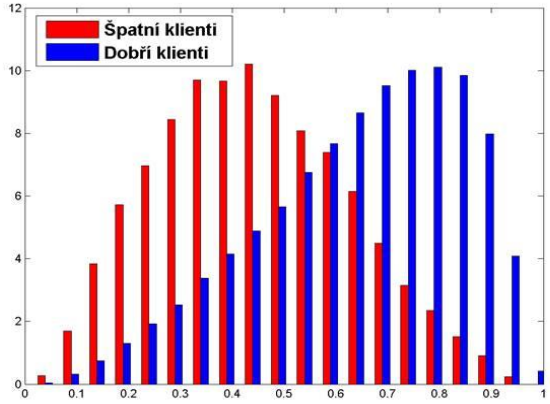


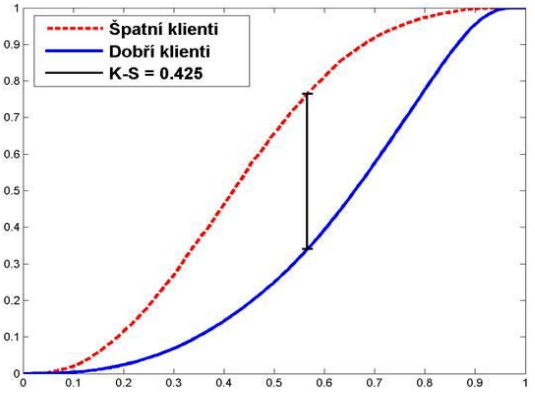
Vizualizace – risk management

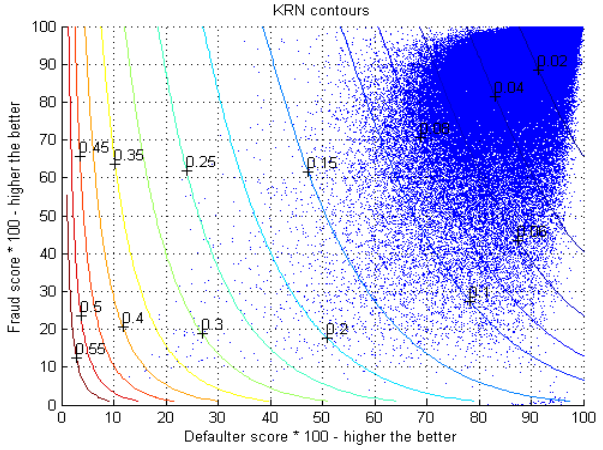




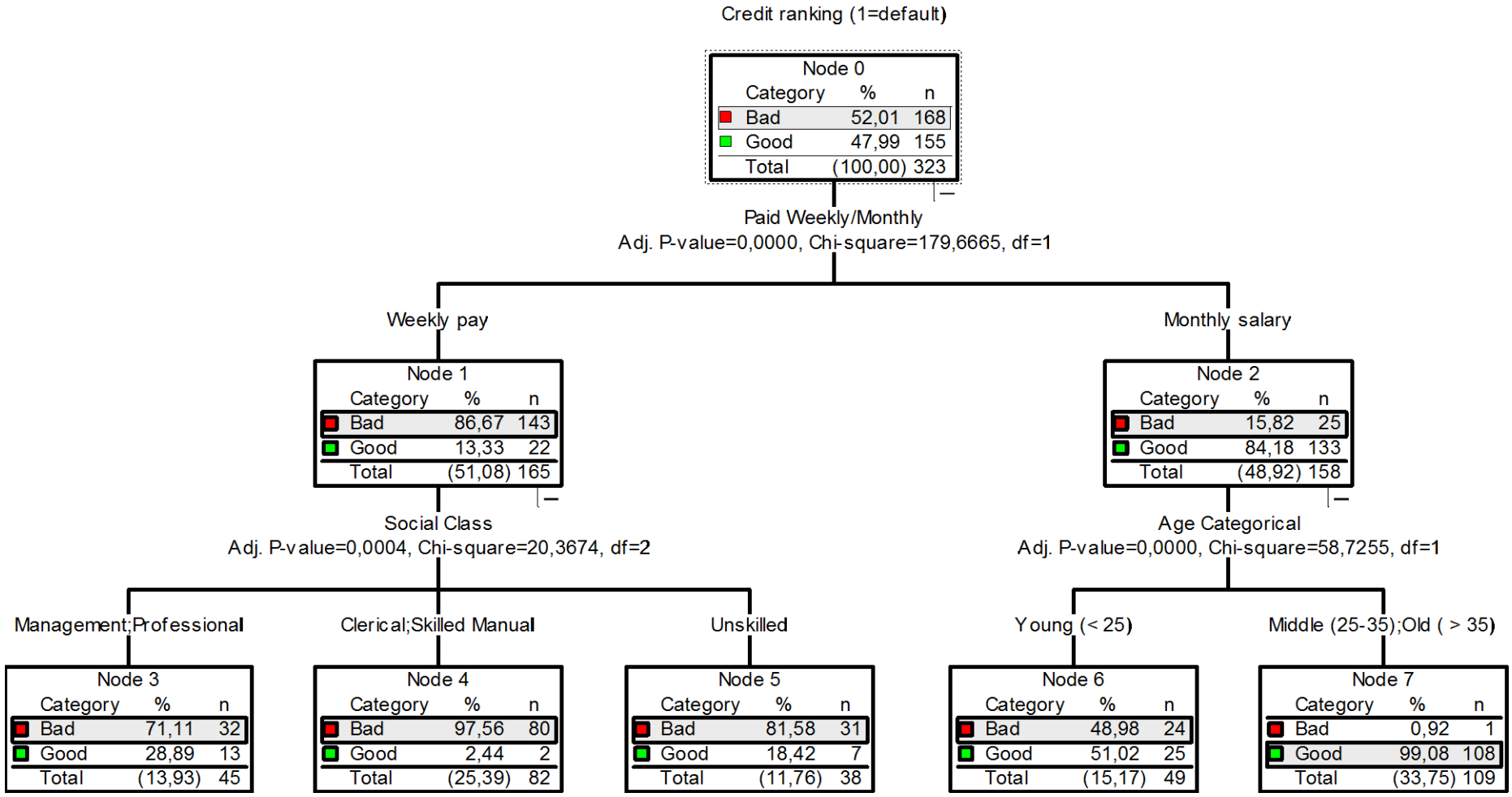




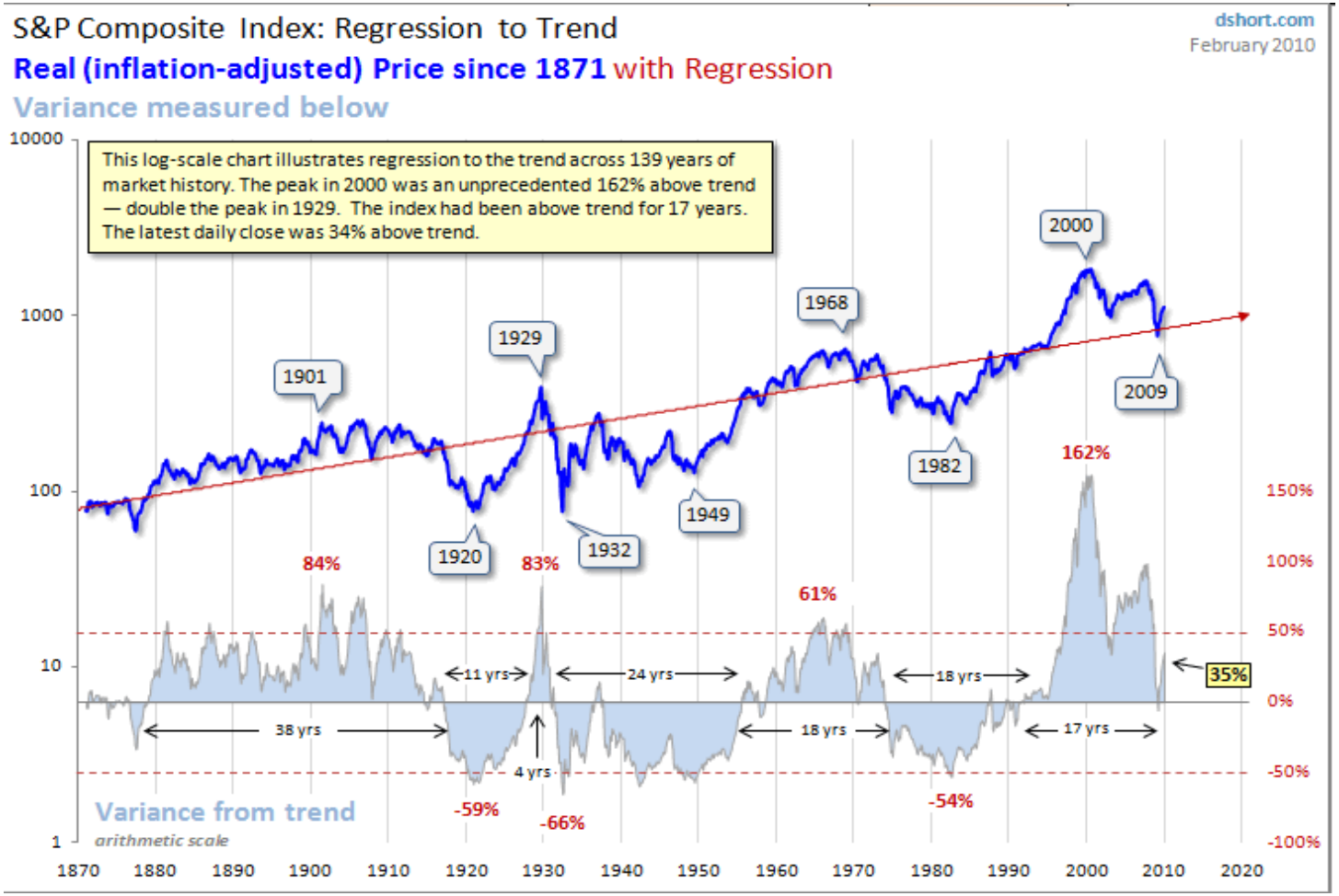




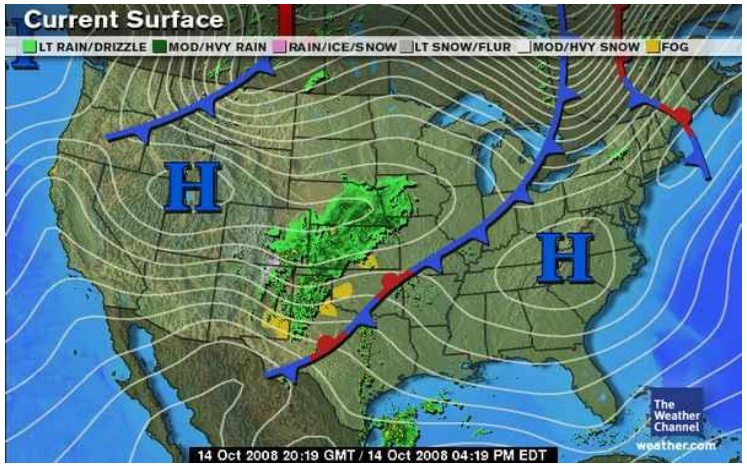
Vizualizace - dendrogram

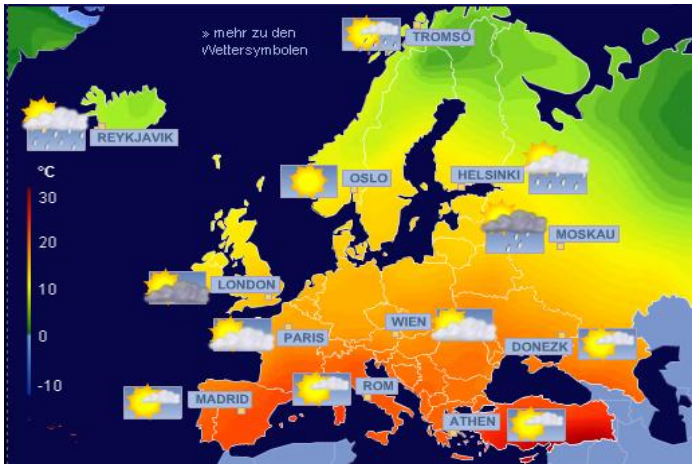


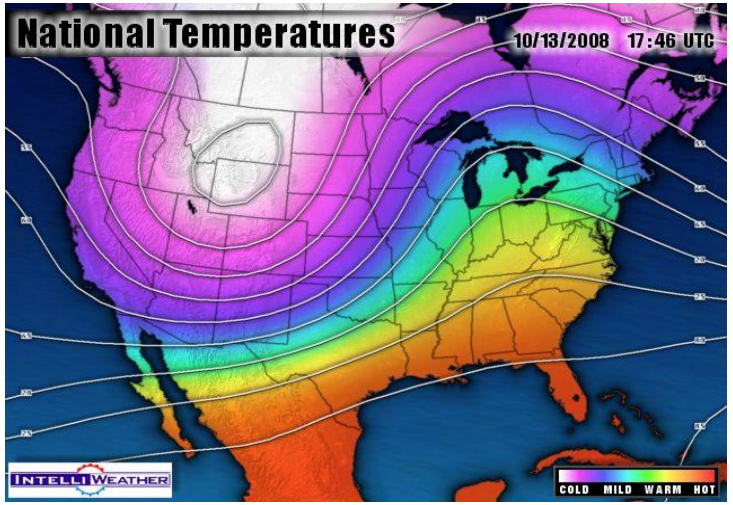
Vizualizace – ekonomie

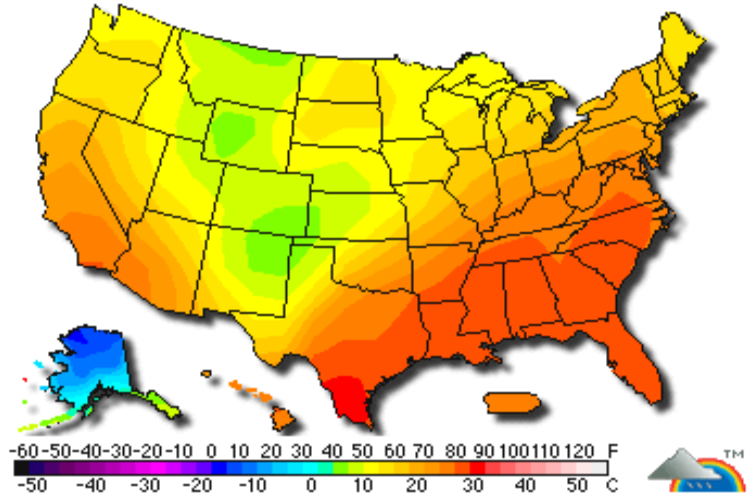


Meteo-vizualizace



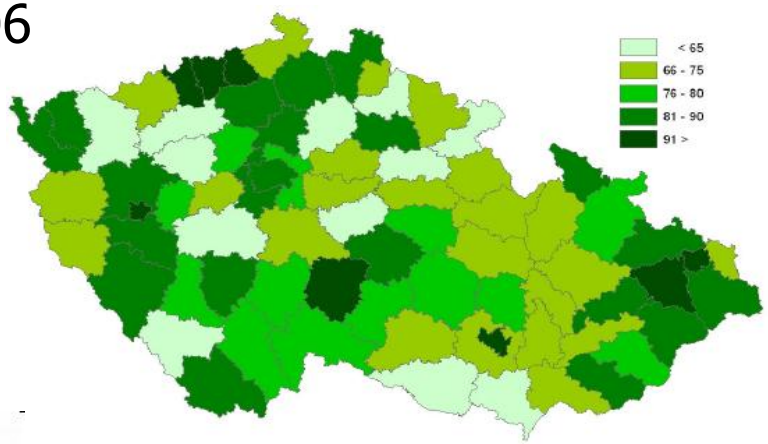


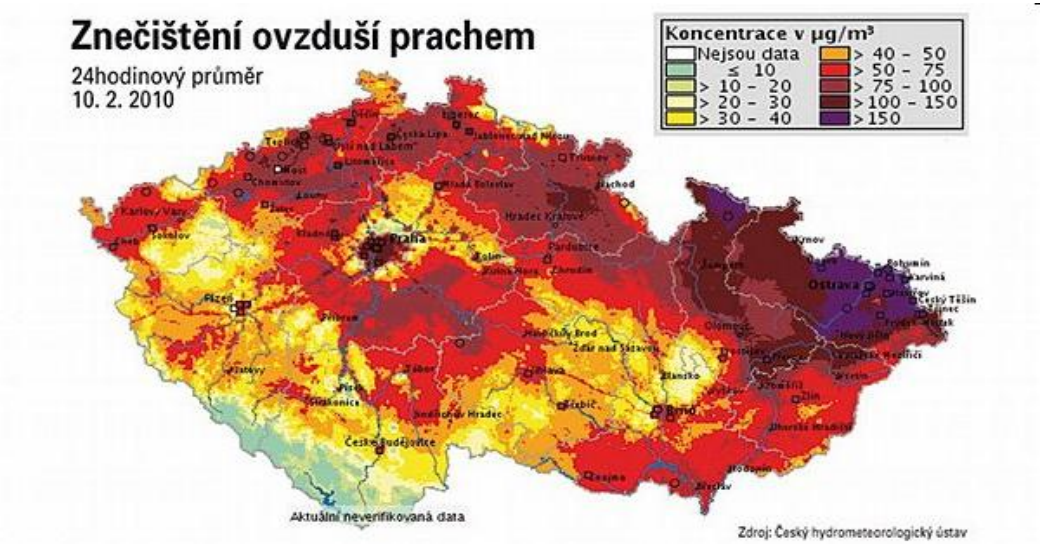




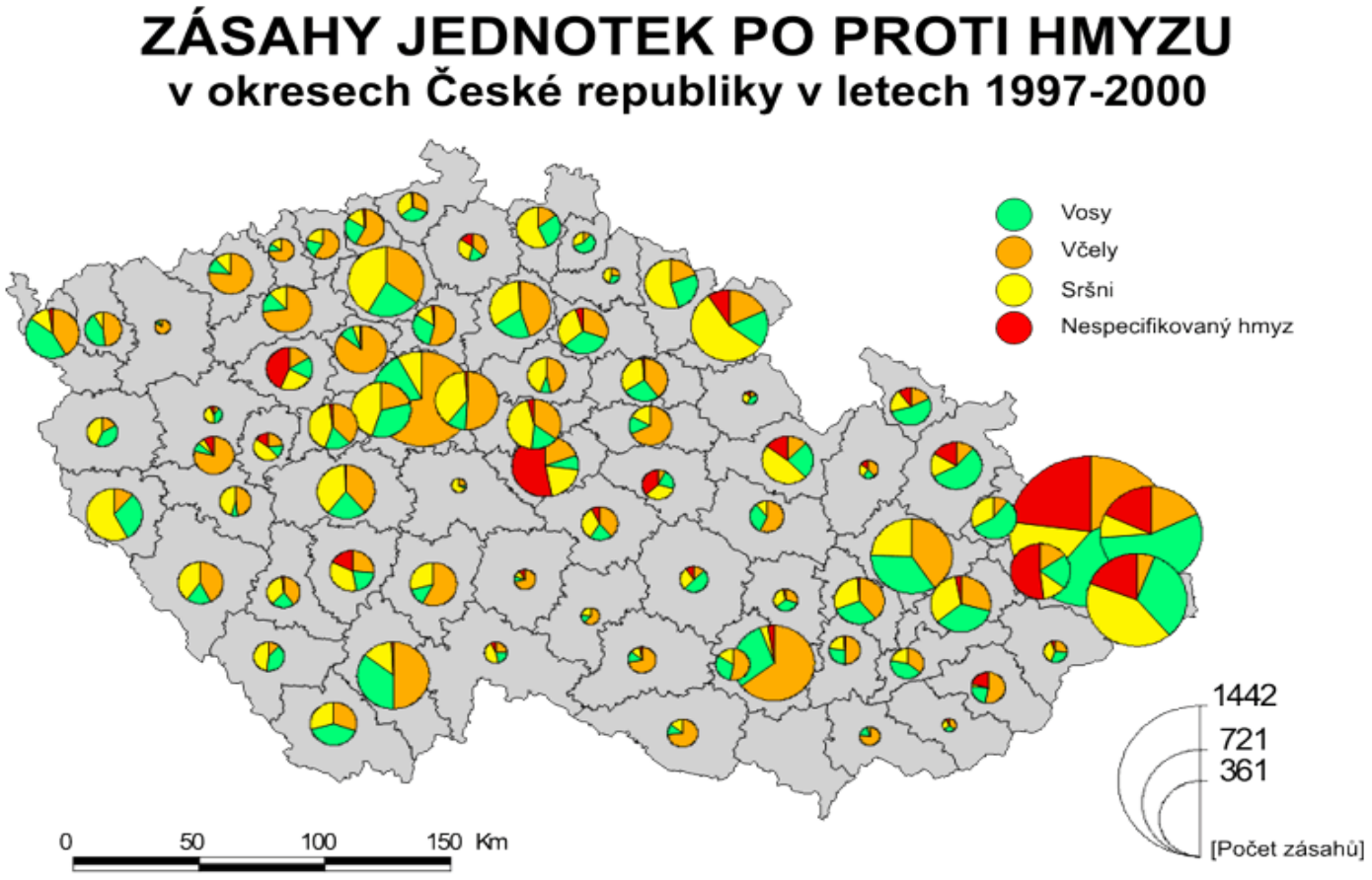
Kartogram

- Obce s počtem 500 a více obyvatel s vysokorychlostním připojením k internetu, podle okresů (%), k 31.12.2006

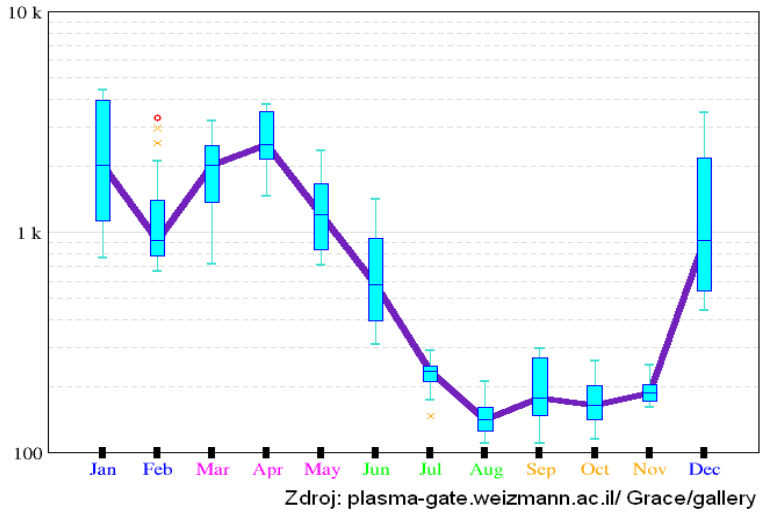




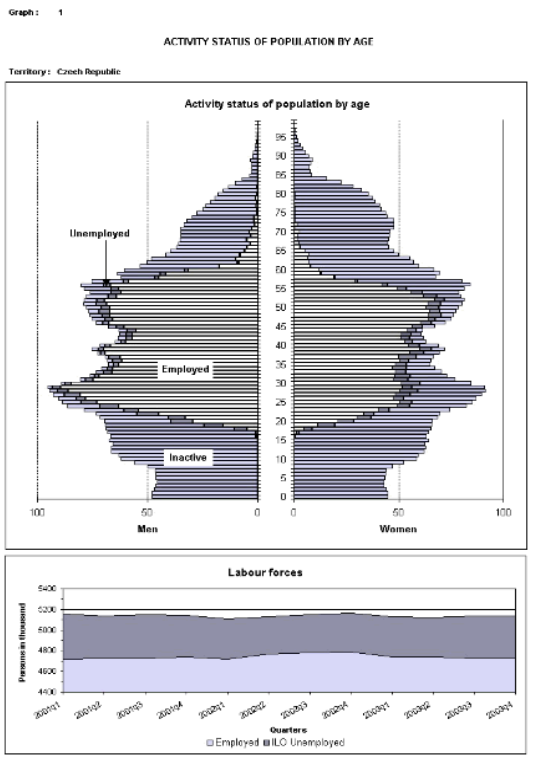
Kartodiagram

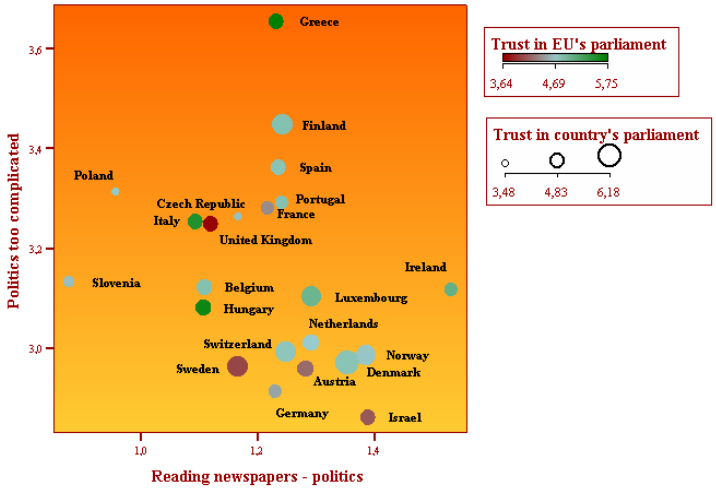


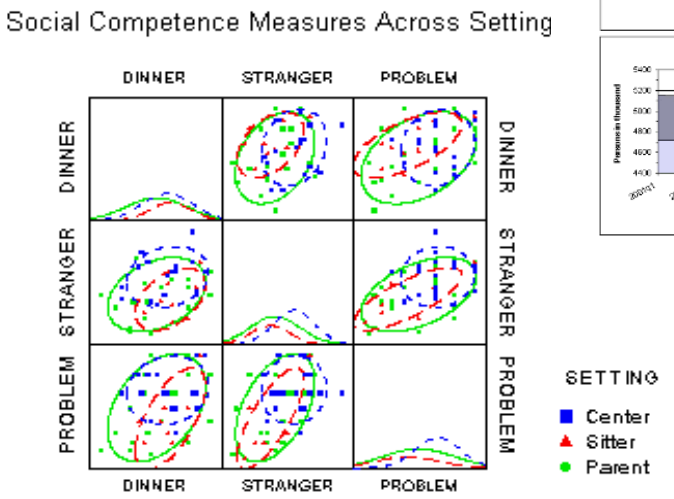
Grafy –další typy



Zdroj: plasma-gate.weizmann.ac.il/ Grace/gallery

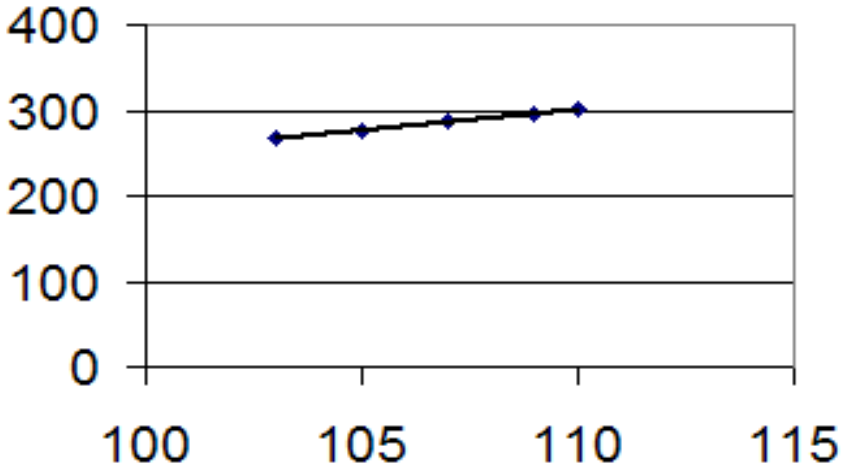


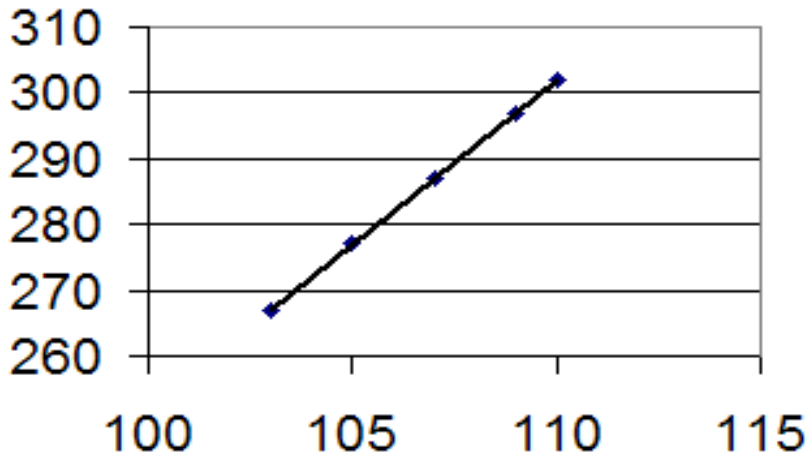




Měřítko grafu

- Která přímka roste strměji?





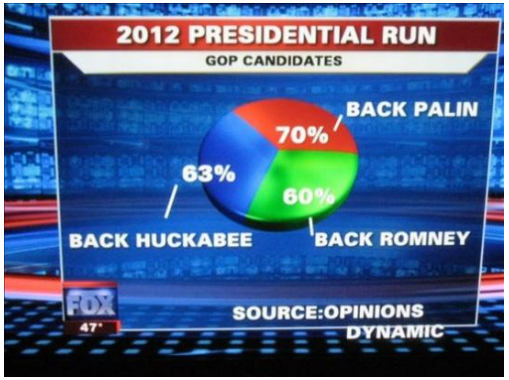
x	y
103	567
105	577
107	587
109	597
110	602

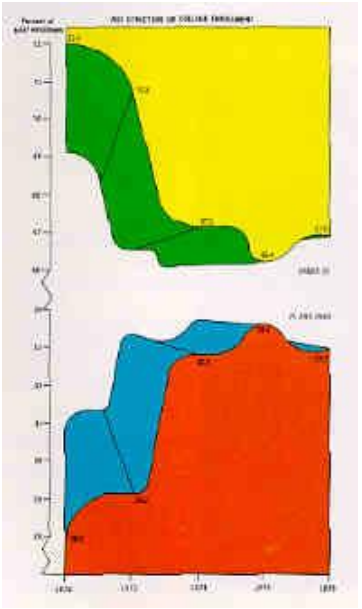
Měřítko grafu

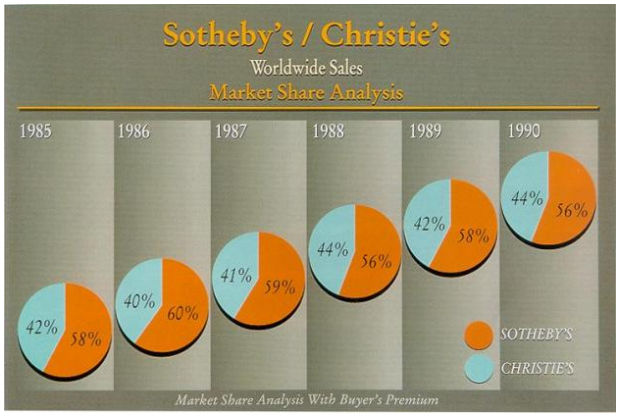
- Pohled tvůrce grafu:
  - Zvýraznění trendu – pozitivní výsledky.
  - Potlačení trendu – negativní výsledky.

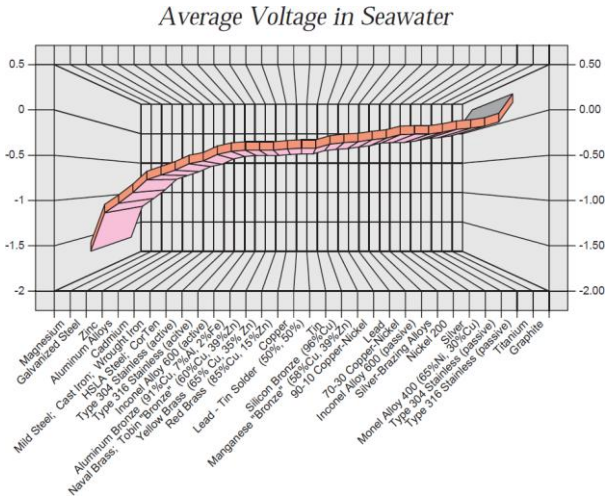
- Pohled uživatele grafu:
  - Grafy bez uvedeného měřítka jsou silně podezřelé.
  - Nepodléhat podsouvané informaci o růstu/poklesu.

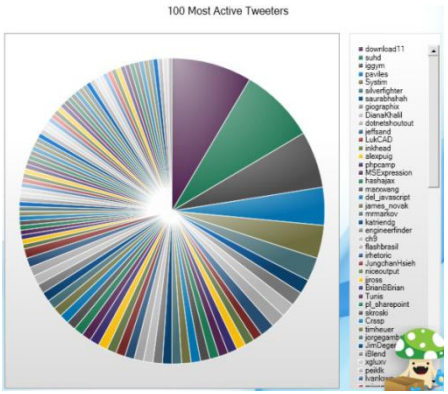
Odstrašující příklady vizualizace:

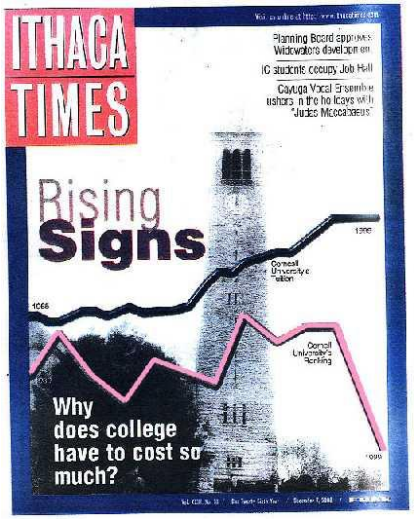












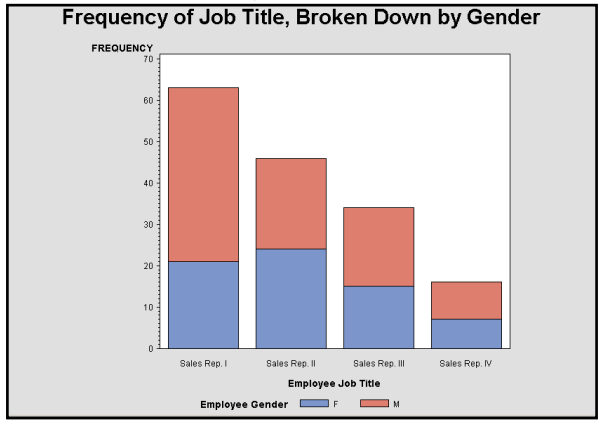
Zdroj: http://www.datavis.ca/gallery/

What Is SAS/GRAPH Software?

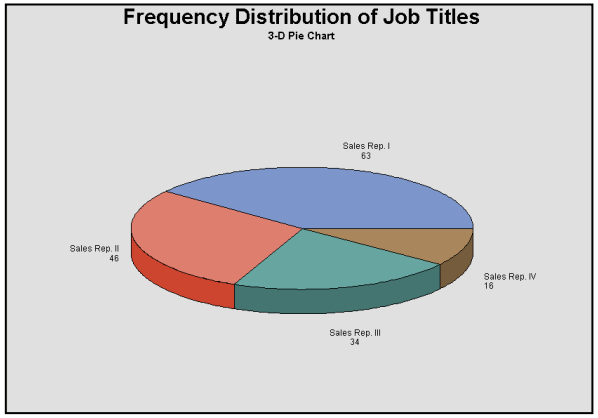
- *SAS/GRAPH software* is a component of SAS software that enables you to create the following types of graphs:
  - bar, block, and pie charts
  - two-dimensional scatter plots and line plots
  - three-dimensional scatter and surface plots
  - contour plots
  - maps
  - text slides
  - custom graphs

Základní typy grafů

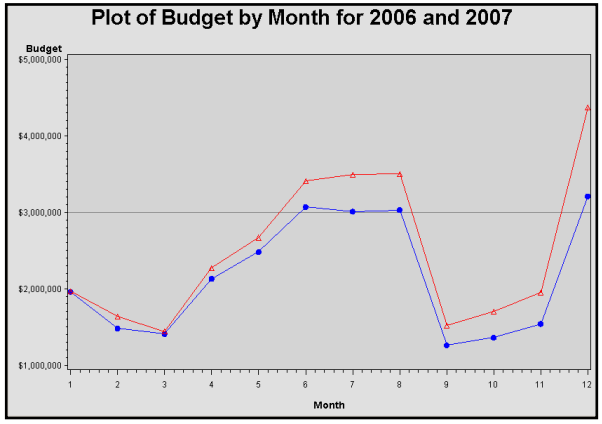
- Bar Charts (GCHART Procedure)



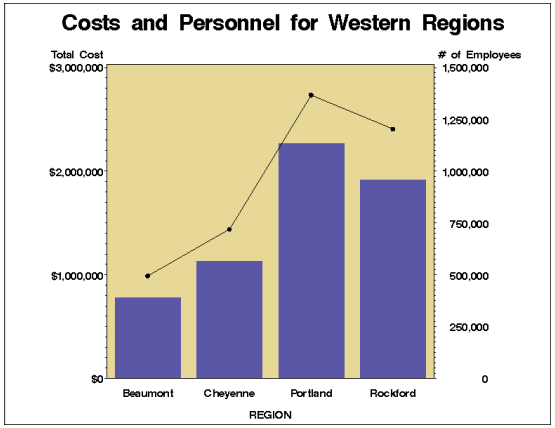
- Pie Charts (GCHART Procedure)



- Scatter and Line Plots (GPLOT Procedure)

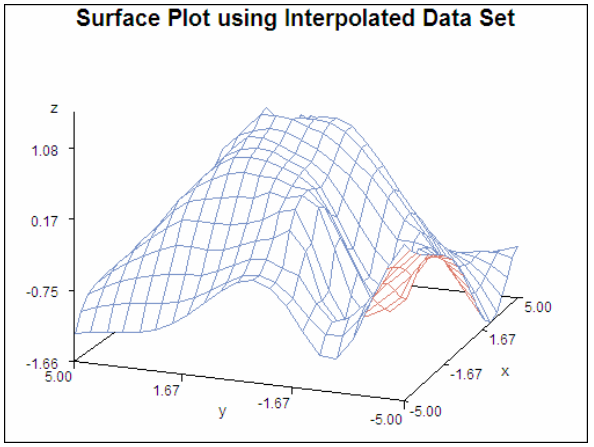


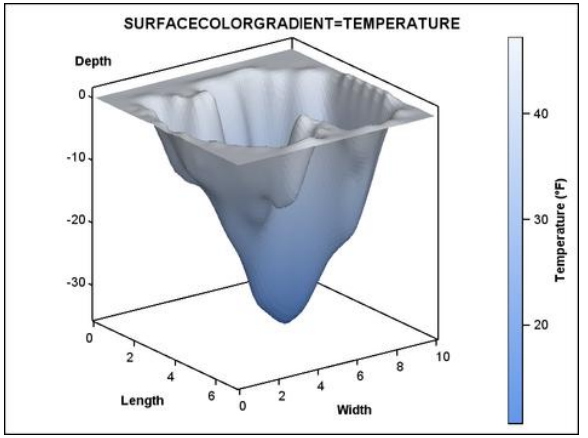
- Bar Charts with Line Plot Overlay (GBARLINE Procedure)



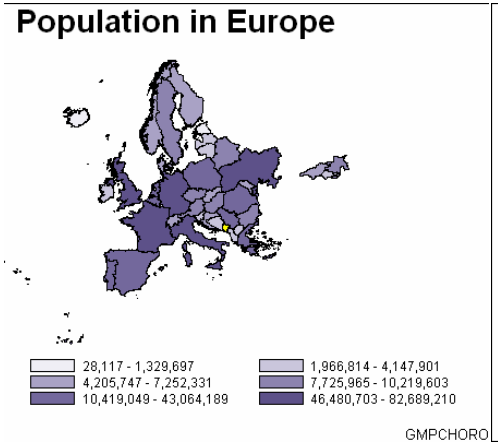
Three-Dimensional Surface and Scatter Plots, Maps

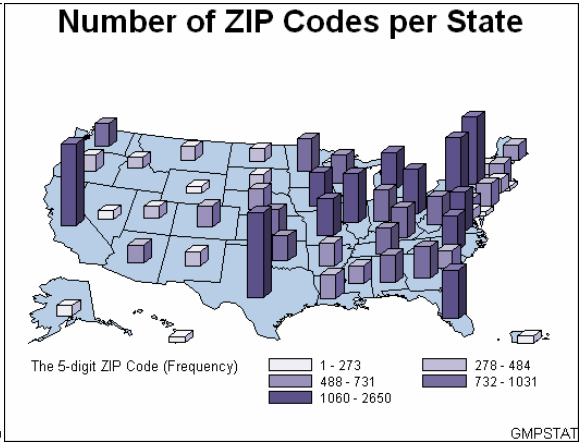
- Procedury G3D, G3GRID, SGRENDER …více na support.sas.com



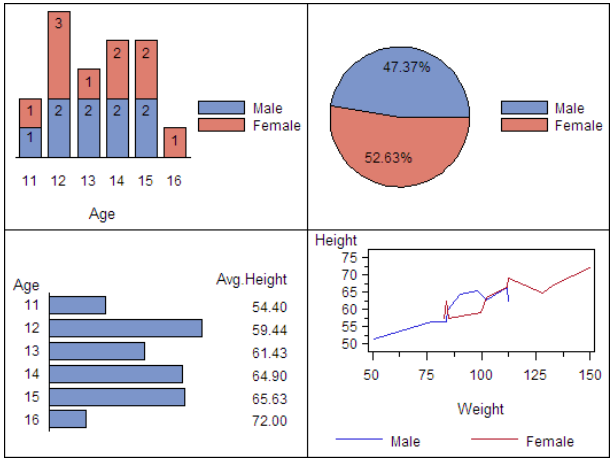


- Maps (GMAP Procedure)





- Multiple graphs on a page (GREPLAY Procedure)



Producing Bar and Pie Charts with the GCHART Procedure

- General form of the PROC GCHART statement:

PROC GCHART DATA=*SAS-data-set*;

- Use one of these statements to specify the chart type:

HBAR *chart-variable . . . </ options>*;
HBAR3D *chart-variable . . . </ options>*;

VBAR *chart-variable . . . </ options>*;
VBAR3D *chart-variable . . . </ options>*;

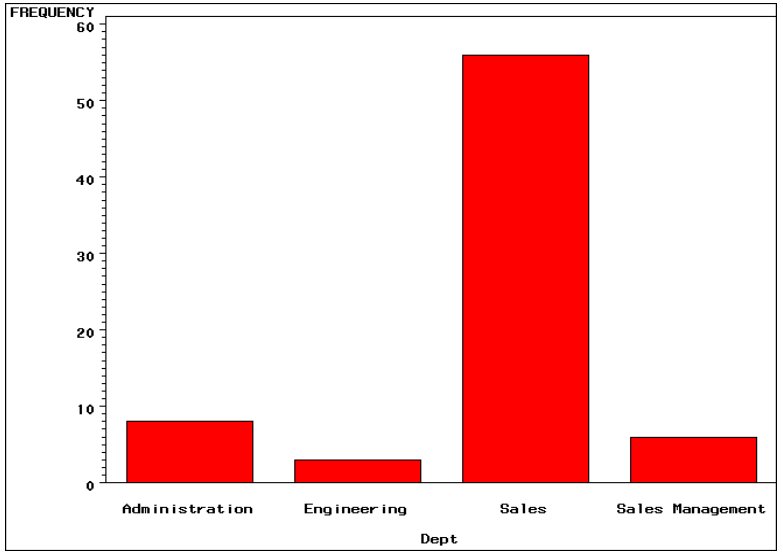
PIE *chart-variable . . . </ options>*;
PIE3D *chart-variable . . . </ options>*;

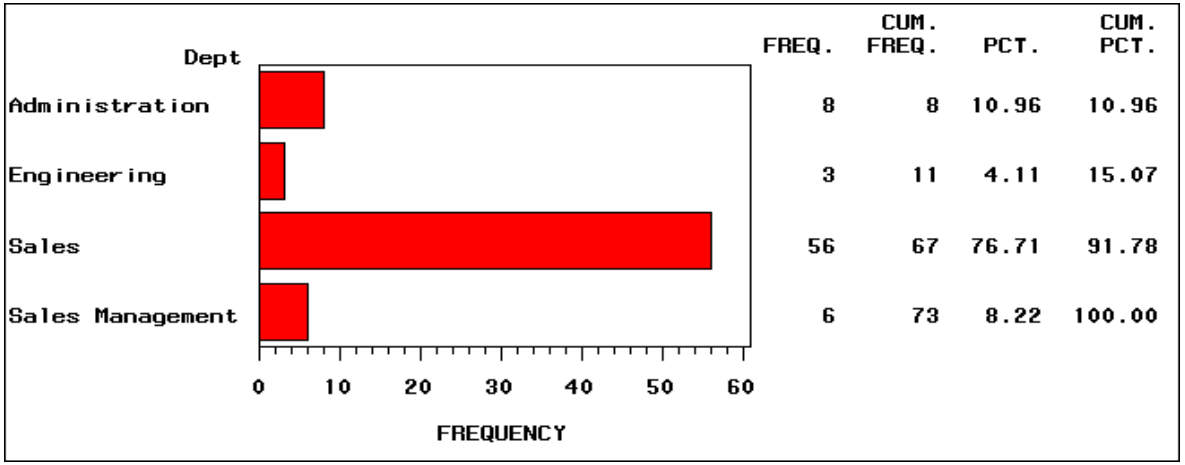
Vertical/horizontal Bar Chart

- Produce a vertical/horizontal bar chart that displays the number of employees in each department.

```
proc gchart
 data=univ.employees;
   vbar dept;
run;
```

```
proc gchart
 data=univ.employees;
   hbar dept;
run;
```





Dept	FREQ.	CUM. FREQ.	PCT.	CUM. PCT.
Administration	8	8	10.96	10.96
Engineering	3	11	4.11	15.07
Sales	56	67	76.71	91.78
Sales Management	6	73	8.22	100.00

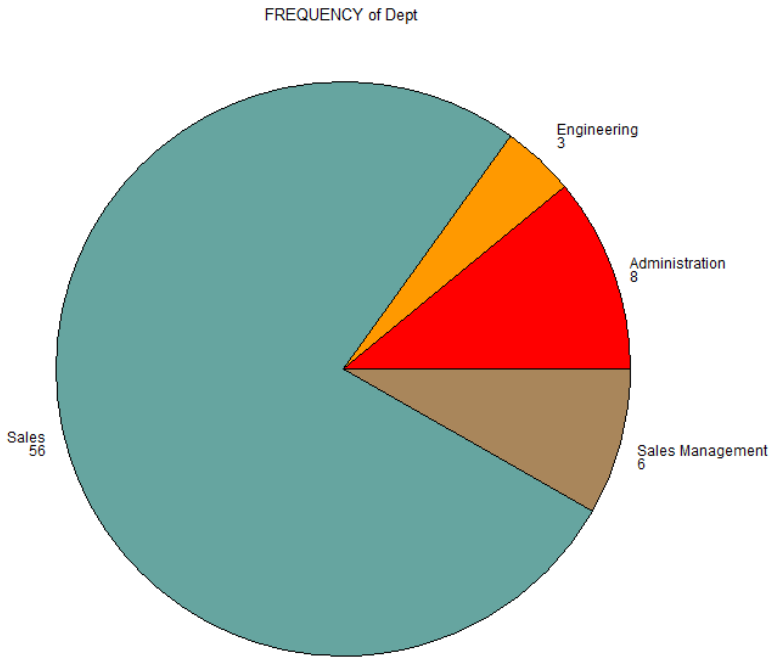
dept is the chart variable

Pie Chart

- Produce a pie chart that displays the number of employees in each department.

```
proc gchart data=univ.employees;
   pie dept;
run;
```

dept is the chart variable



Character/Numeric Chart Variable

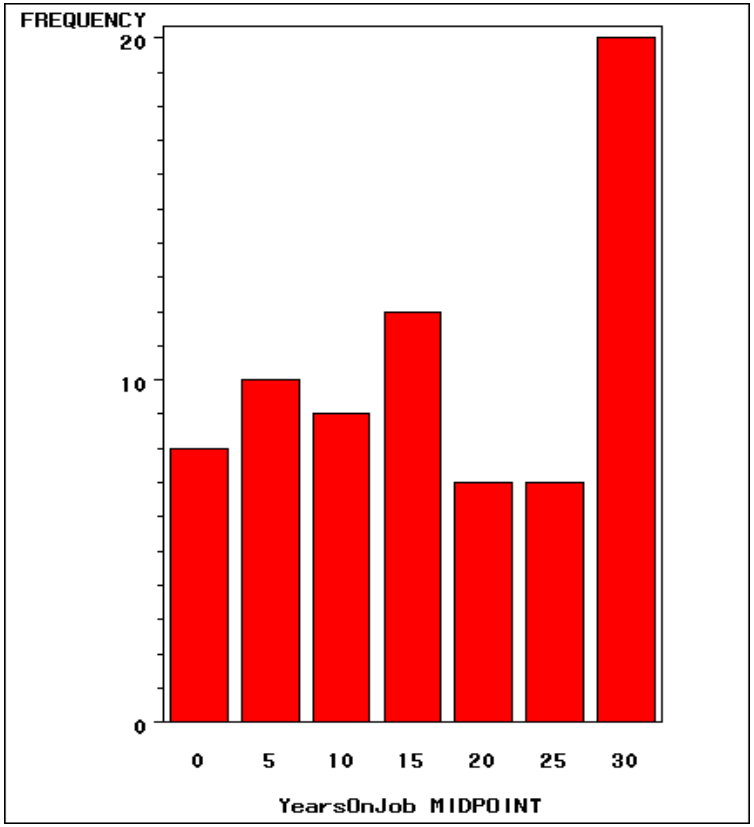
- If the chart variable is **character**, then a bar or slice is created for each unique variable value.
- For **numeric** chart variables, the variables are assumed to be continuous unless otherwise specified.
- The GCHART procedure creates the equivalent of a histogram from the data.
  - Intervals are automatically calculated and identified by midpoints.
  - One bar or slice is constructed for each midpoint.

Numeric Chart Variable

- Produce a vertical bar chart on the numeric variable **YearsOnJob**.

```
proc gchart data=univ.employees;
   vbar YearsOnJob;
run;
```

YearsOnJob is the chart variable

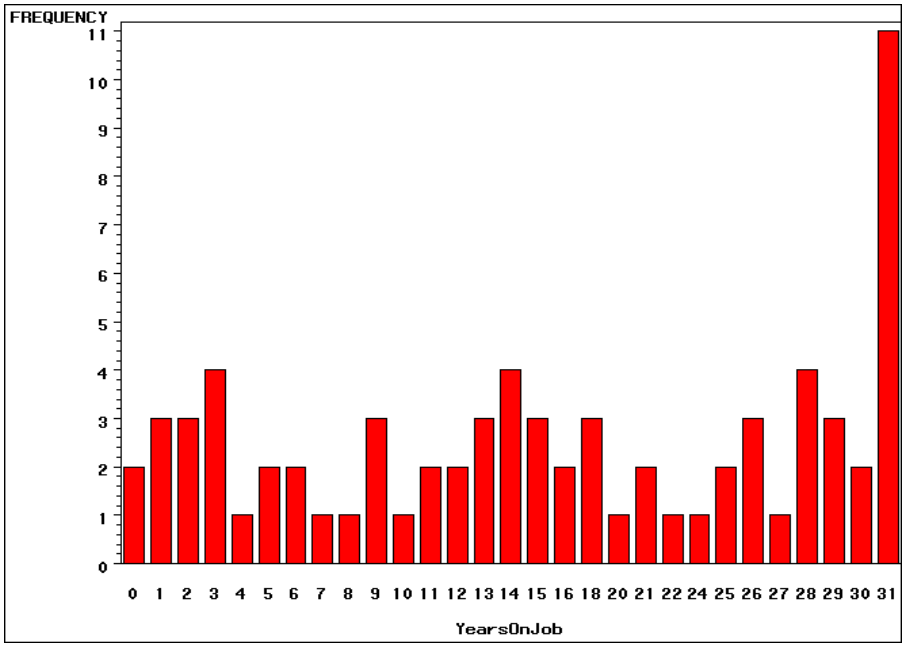


The DISCRETE Option

- To override the default behavior for numeric chart variables, use the DISCRETE option in the HBAR, VBAR, or PIE statement.
- The DISCRETE option produces a bar or slice for each unique numeric variable value; the values are no longer treated as intervals.

```
proc gchart data=univ.employees;
   vbar YearsOnJob / discrete;
run;
```

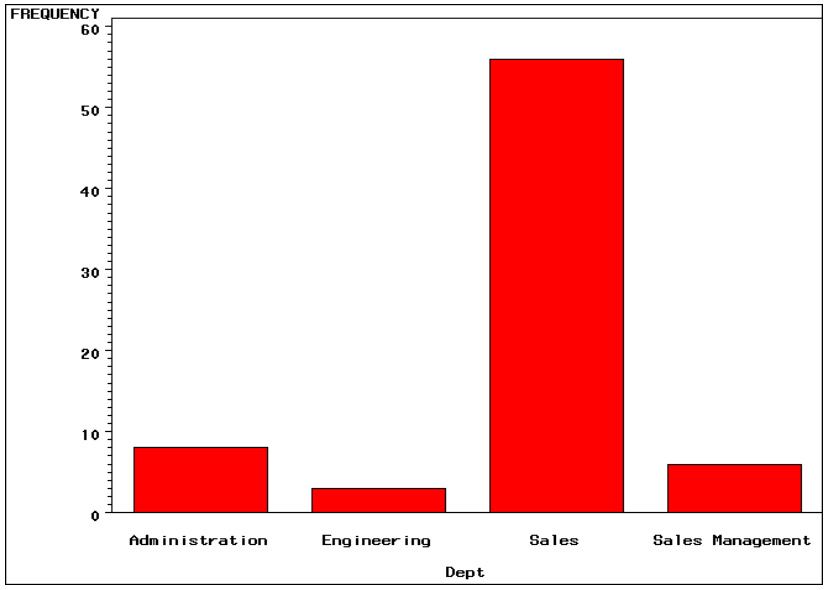
YearsOnJob is the chart variable, but the DISCRETE option modifies how SAS displays the values.

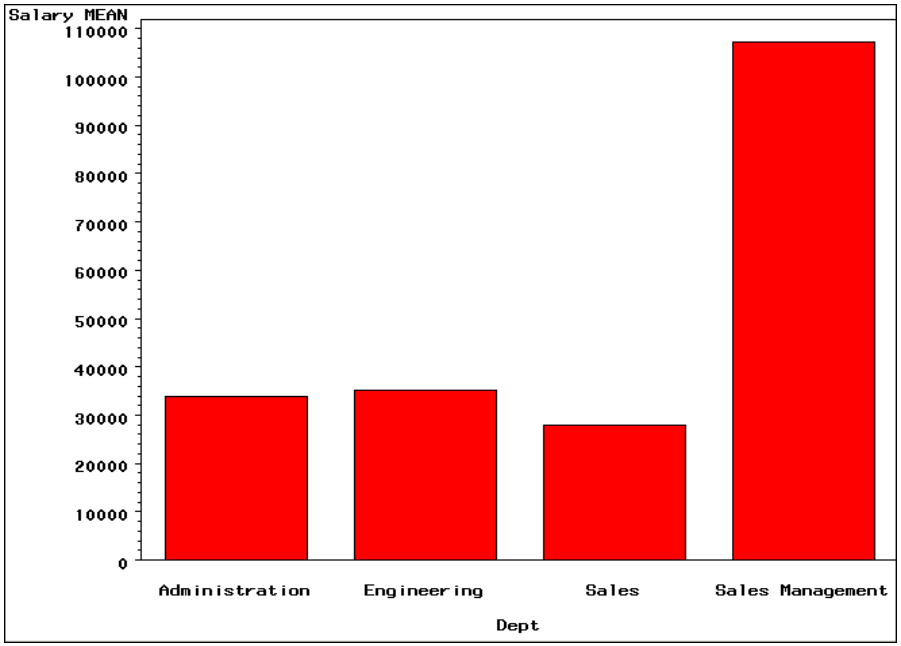


Summary Statistic

- By default, the statistic that determines the length or height of each bar or size of pie slice is a frequency count (N).

```
proc gchart data=univ.employees;
   vbar dept / sumvar=salary type=mean;
run;
```





Analysis Variable

- To override the default frequency count, you can use the following HBAR, VBAR, or PIE statement options:

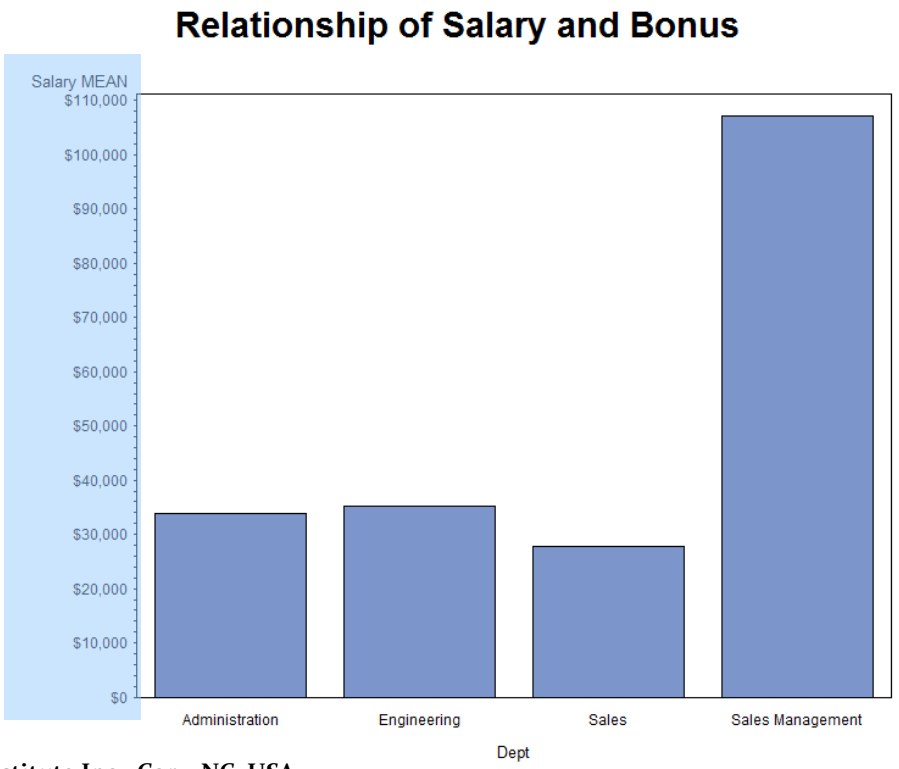
SUMVAR=	identifies the analysis variable to use for the sum or mean calculation.
TYPE=	specifies that the height or length of the bar or size of the slice represents a mean or sum of the *analysis-variable* values.

- If an analysis variable is
  - specified, the default value of TYPE is SUM
  - not specified, the default value of TYPE is FREQ.

Bar Chart Using Formats

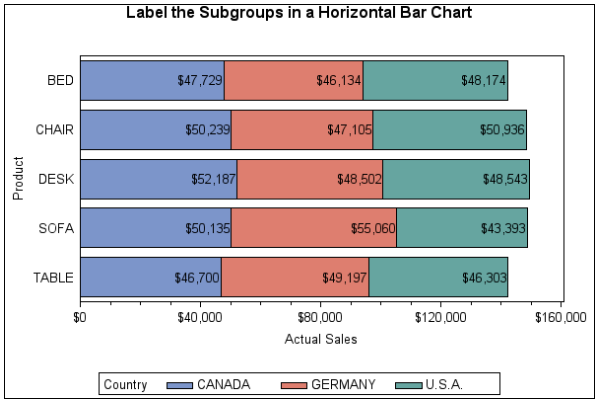
- Produce a bar chart that displays the average salary of employees in each department.

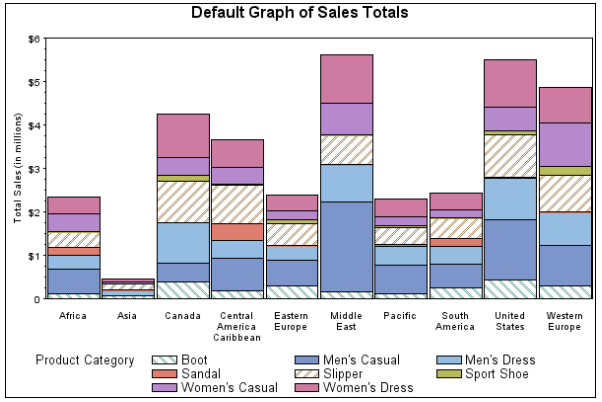
```
proc gchart data=univ.employees;
    vbar dept / sumvar=Salary type=mean;
    format Salary dollar8.;
run;
```

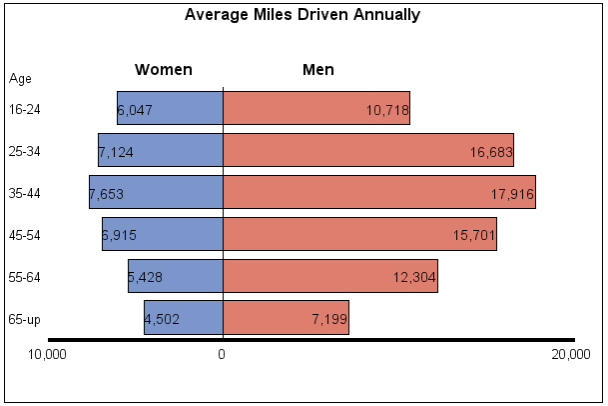


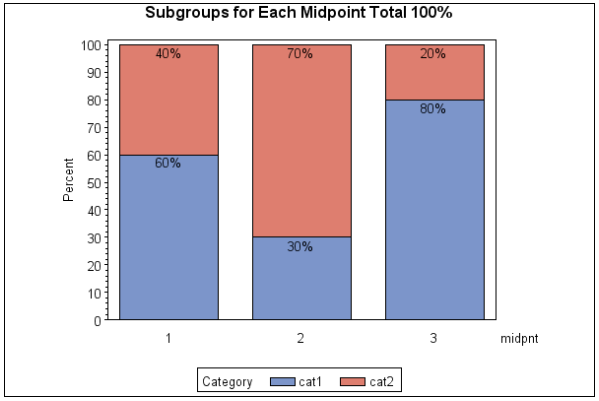
Další možnosti proc gchart

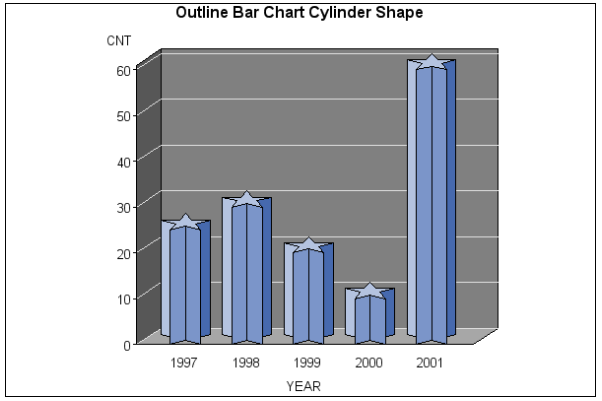
- Na adrese http://support.sas.com/sassamples/graphgallery/PROC_GCHART.html lze nalézt galerii možných typů grafů (**včetně kódů**!).

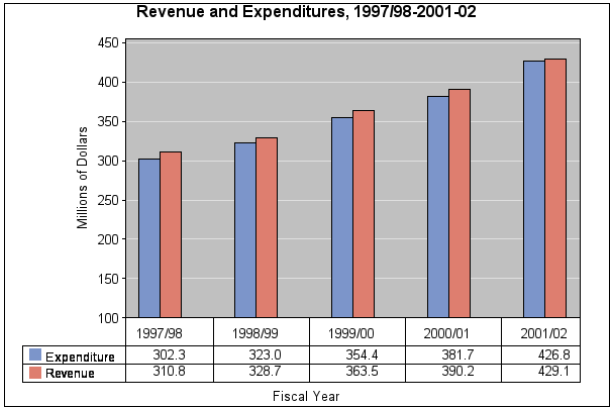






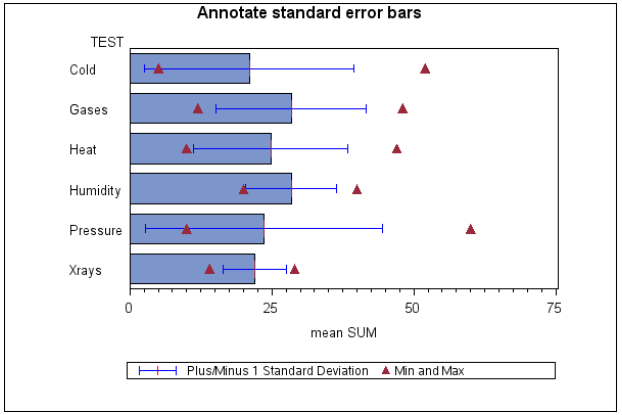


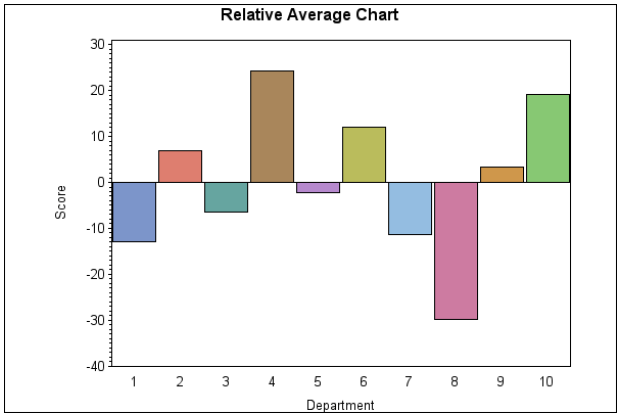


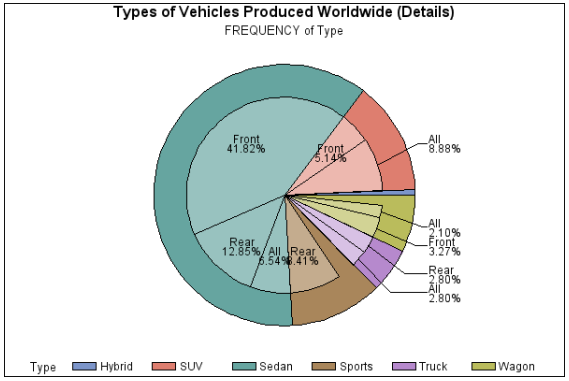


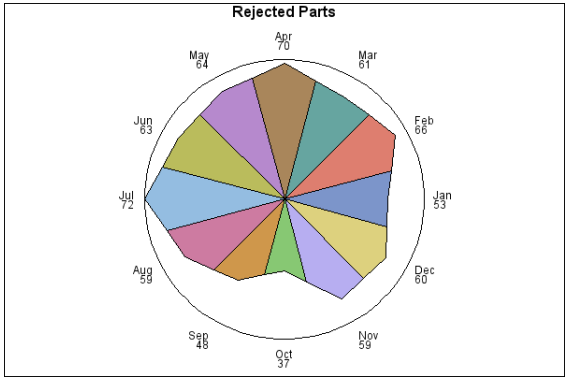
Další možnosti proc gchart

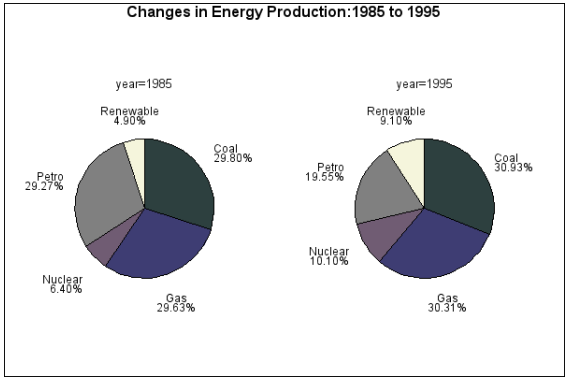
- A ještě několik typů...

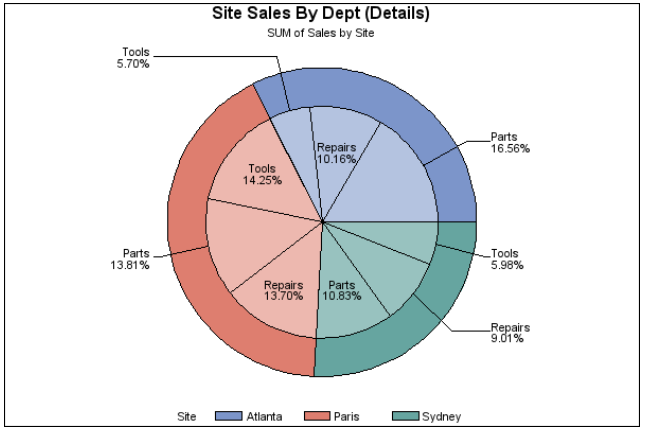












Producing Plots with the GPLOT Procedure

- You can use the GPLOT procedure to plot one variable against another within a set of coordinate axes.

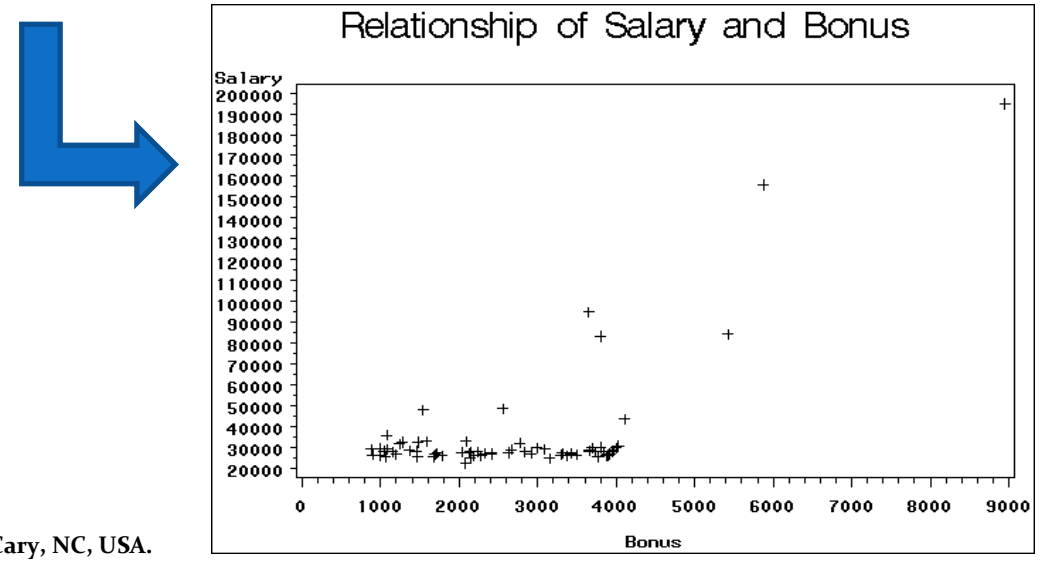
- General form of a PROC GPLOT step:

```
PROC GPLOT DATA=SAS-data-set;
      PLOT vertical-variable*horizontal-variable </ options>;
RUN;
QUIT;
```

The GPLOT Procedure

Produce a plot of salary versus bonus for each employee.

```
proc gplot data=univ.employees;
   plot Salary*Bonus;
   title 'Relationship of Salary and Bonus';
run;
```



SYMBOL Statement

- You can use the SYMBOL statement to do the following:
  - define plotting symbols
  - draw lines through the data points
  - specify the color of the plotting symbols and lines

- General form of the SYMBOL statement:

SYMBOL*n* options;

- The value of *n* can range from 1 to 255.
- If *n* is omitted, the default is 1.
- Symbol statement is global and additive:

global	After being defined, the statements remain in effect until changed or until the end of the SAS session.
additive	Specifying the value of one option does not affect the values of other options.

SYMBOL Statement Options

- You can specify the plotting symbol you want with the VALUE= option in the SYMBOL statement:

VALUE=*symbol* | **V=***symbol*

- Selected *symbol* values are shown below:

PLUS (default)	DIAMOND
STAR	TRIANGLE
SQUARE	NONE (no plotting symbol)

- You can use the I= option in the SYMBOL statement to draw lines between the data points.
- Selected *interpolation* values:

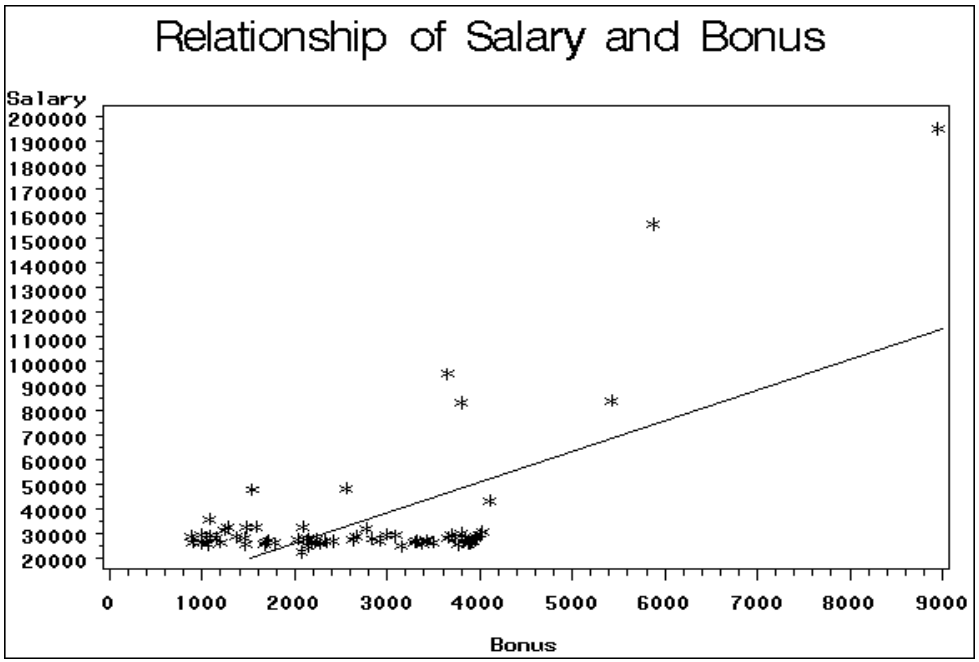
I=*interpolation*

JOIN	joins the points with straight lines.
SPLINE	joins the points with a smooth line.
NEEDLE	draws vertical lines from the points to the horizontal axes.
R	overlays a simple linear regression line on the plot.

SYMBOL Statement Options

- Use a star as the plotting symbol and superimpose a regression line on the plot.

```
plot Salary*Bonus;
  symbol value=star i=r;
run;
```

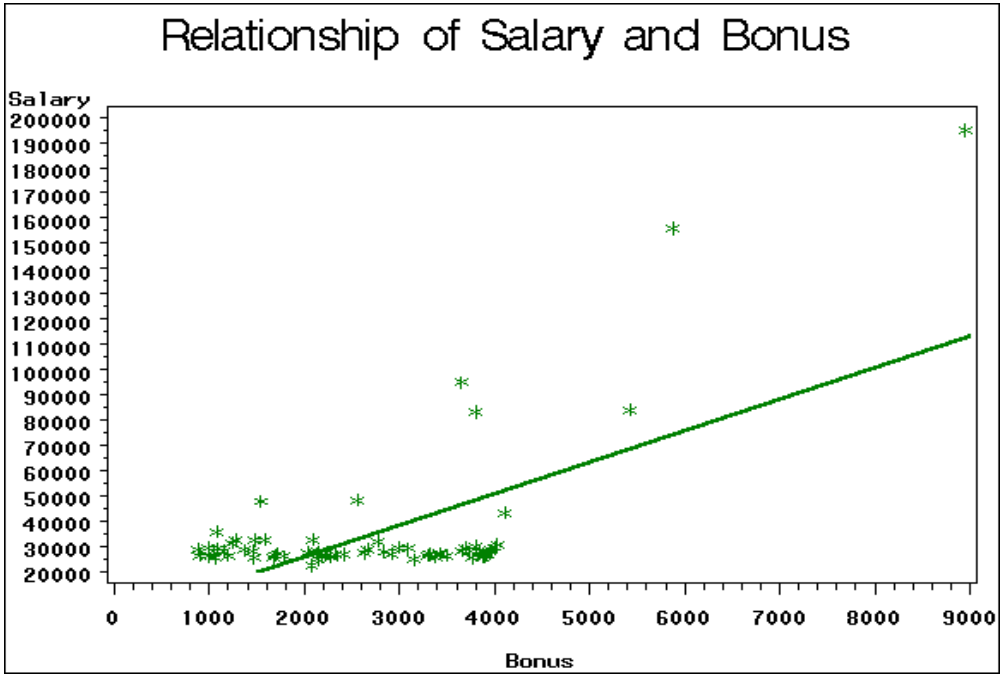


Additional SYMBOL Statement Options

- You can enhance the appearance of the plots with the following selected options:

WIDTH=*width* **W**=*width*	specifies the thickness of the line.
COLOR=*color* **C**=*color*	specifies the color of the line and plot symbols.

```
plot Salary*Bonus;
symbol c=green w=3;
run;
```



Canceling SYMBOL Statements

- You can cancel a SYMBOL statement by submitting a null SYMBOL statement.

```
symbol1;
```

- To cancel all SYMBOL statements, submit the following statement:

```
goptions reset=symbol;
```

- Zrušení všech předchozích voleb (návrat k defaultnímu nastavaní)

```
goptions reset=global;
```

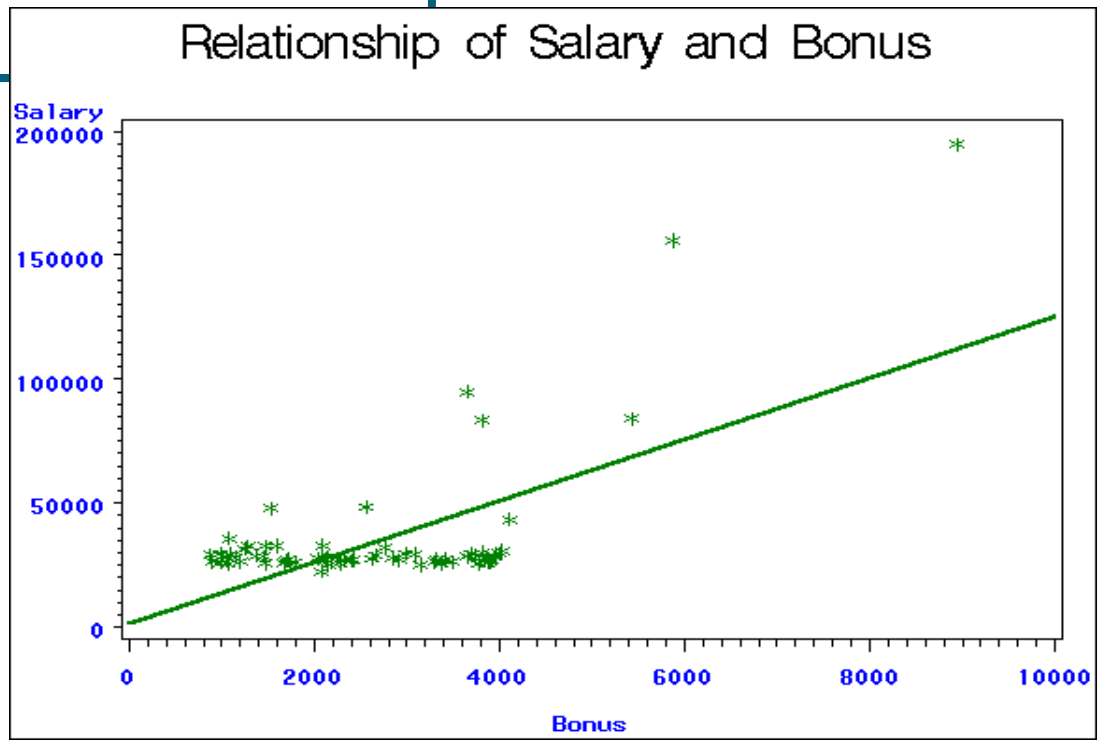
Controlling the Axis Appearance

- You can modify the appearance of the axes that PROC GPLOT produces with the following:
  - PLOT statement options
  - the LABEL statement
  - the FORMAT statement
- You can use PLOT statement options to control the scaling and color of the axes, and the color of the axis text.
- Selected PLOT statement options for axis control:

HAXIS=*values*	scales the horizontal axis.
VAXIS=*values*	scales the vertical axis.
CAXIS=*color*	specifies the color of both axes.
CTEXT=*color*	specifies the color of the text on both axes.

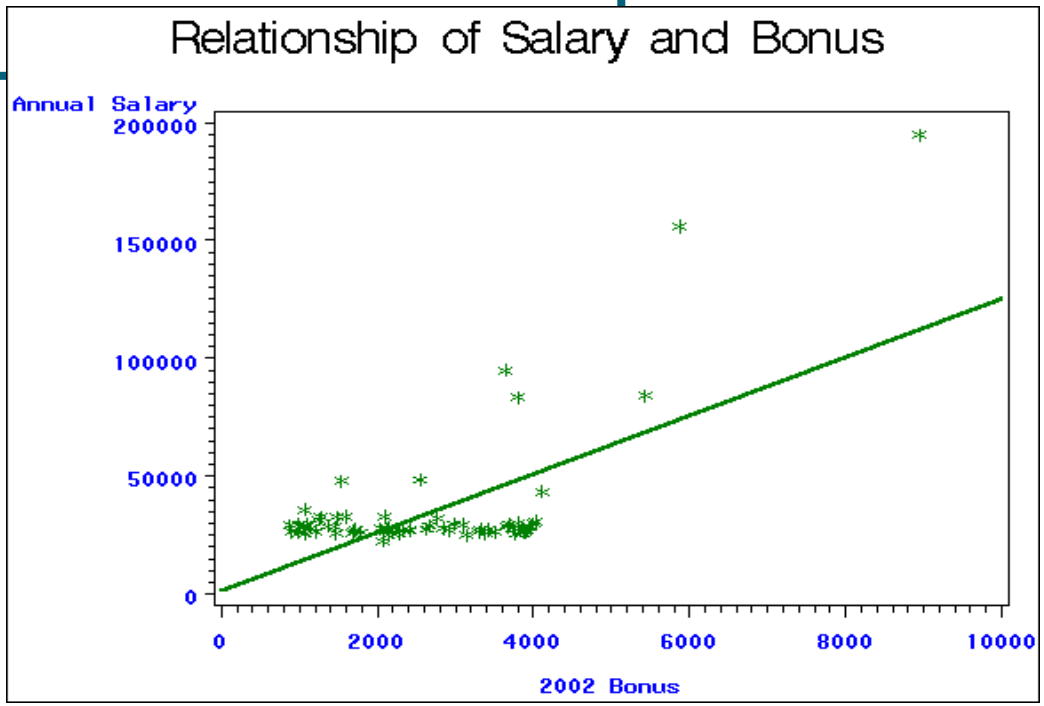
PLOT Statement Options

```
plot Salary*Bonus
    / vaxis=0 to 200000 by 50000
      haxis=0 to 10000 by 2000
      ctext=blue;
run;
```



LABEL Statement

```
plot Salary*Bonus /
    vaxis=0 to 200000 by 50000
    haxis=0 to 10000 by 2000 ctext=blue;
    label Salary='Annual Salary'
        Bonus='2002 Bonus';
run;
```



Gplot options – další možnosti

Plot <*Y Variable*>*<*X Variable*> / <*options*>;

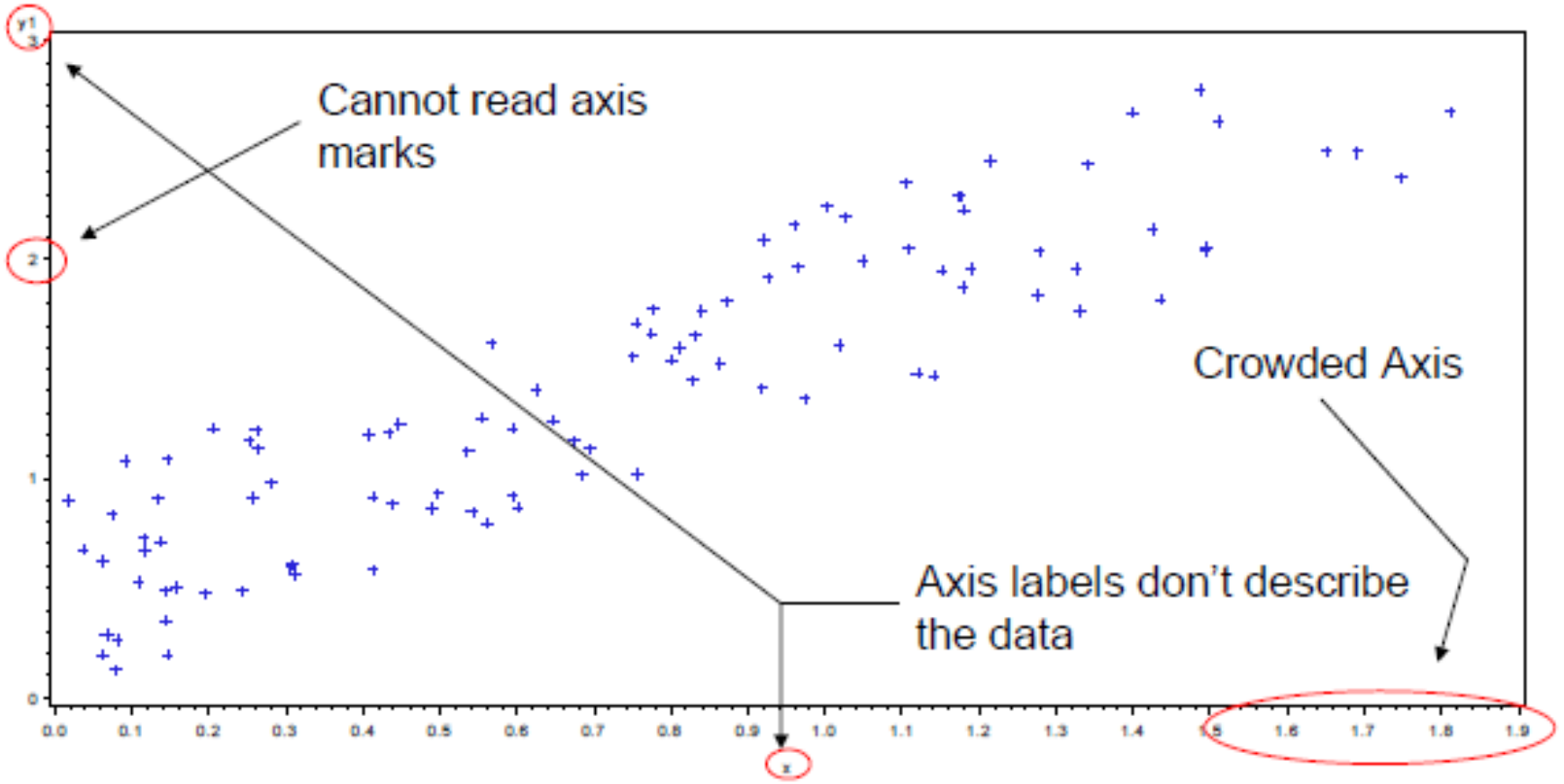
- Options for plotting
  - Plot options
    - •Legend= or nolegend: specifies figure legend options
    - •Overlay: allows overlay of more than one Y variable
    - •Skipmiss: breaks the plotting line where Y values are missing

  - •Appearance option
    - •Axis: Specifies axis label and value options
    - •Symbol: Specified symbol options
    - •href, vref: Draws vertical or horizontal reference lines on plot
    - •frame/fr or noframe/nofr: specifies whether or not to frame the plot
    - •caxis/ca, cframe/cfr, chref/ch, cvref/cv, ctext/c: specifies colors used for axis, frame, text or reference lines.

Gplot options – další možnosti

```
proc gplot data=twovar;
      plot y1*x;
run;
```

Very basic plot, below we get all of the default options. Not very exciting. Definitely not publication quality.



Gplot options – další možnosti

- AXIS<1..99> <options>;
  - Label Option;
    - Angle/a=degrees (0-359)
    - Color/c=text color
    - Font/f=font
    - Height/h=text height (default=1)
    - Justify=(left/center/right)
    - Label="text string"
  - Order Option;
    - Order=(a to b by c): major tick marks will show up at intervals based on c.
      - Example order=(0 to 3 by 1);
  - Value Option;
    - value("" "" ""): applies text label to each major tick.
      - Example Value=("Start" "Middle" "End")

- axis1 label=(a=90 c=black f="arial" h=1.2 "time" a=90 c=black f="arial" h=1.0 "hours");

Gplot options – další možnosti

Resets previous options → `goptions reset=global ;`

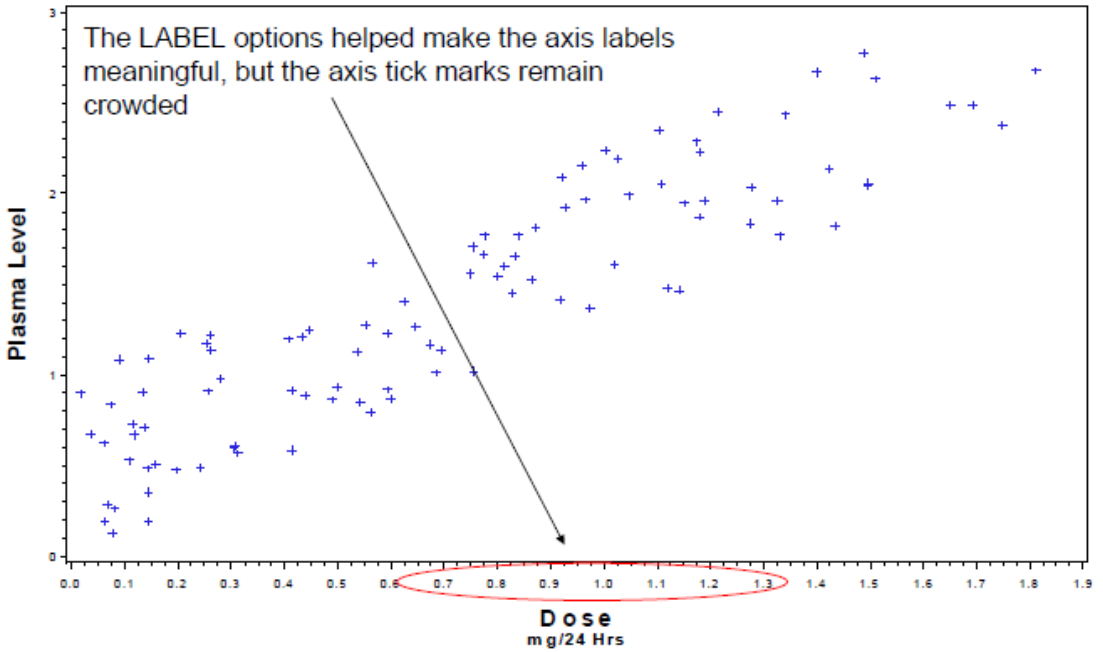
Horizontal axis (X Variable) → `axis1 label=(f='arial/bo' h=1.9 "Dose" justify=c f='arial/bo' h=1.3 "mg/24 Hrs" );`

Vertical axis (Y Variable) → `axis2 label=(a=90 f='arial/bo' h=1.9 "Plasma Level");`

```
proc gplot data=twovar;
     plot y1*x / haxis=axis1 vaxis=axis2;
run;
```

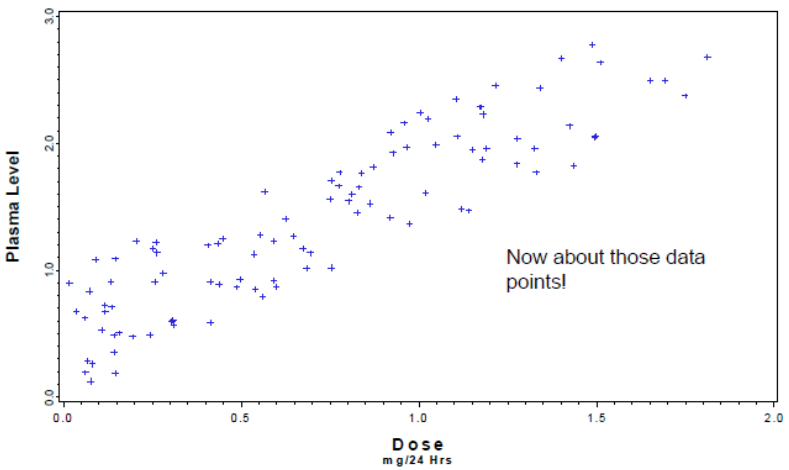
Call Axis statements

NOTE: you can also place the AXIS statements within the gplot proc



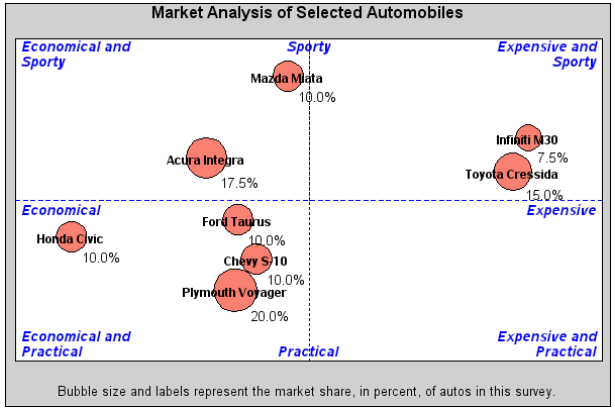
Added VALUE option to Axis statement

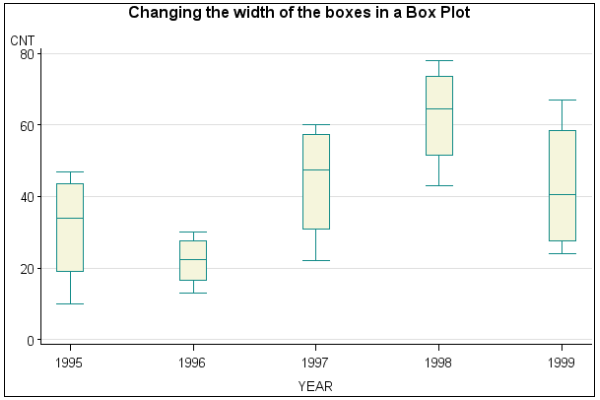
```
axis1 label=(f='arial/bo' h=1.9 "Dose" justify=c
             f='arial/bo' h=1.3 "mg/24 Hrs")
      order=(0 to 2 by 0.5)
      value=(f='arial' h=1.3 "0.0" "0.5" "1.0" "1.5" "2.0");

axis2 label=(a=90 f='arial/bo' h=1.9 "Plasma Level")
      order=(0 to 3 by 1)
      value=(a=90 f='arial' h=1.3 "0.0" "1.0" "2.0" "3.0");


proc gplot data=twovar;
     plot y1*x / haxis=axis1 vaxis=axis2;
run;
```

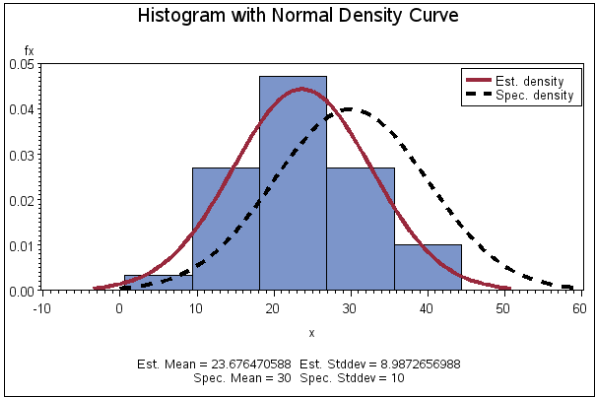


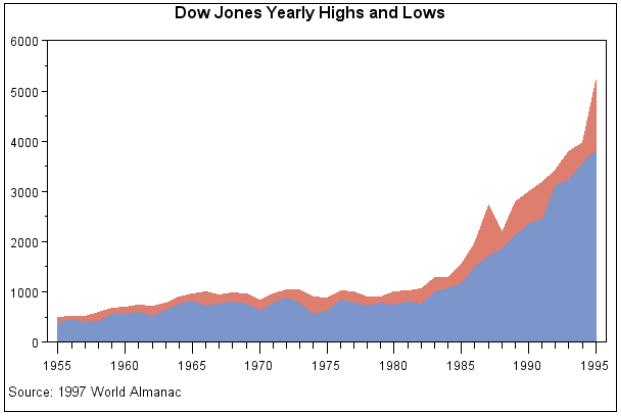
Další možnosti proc gplot

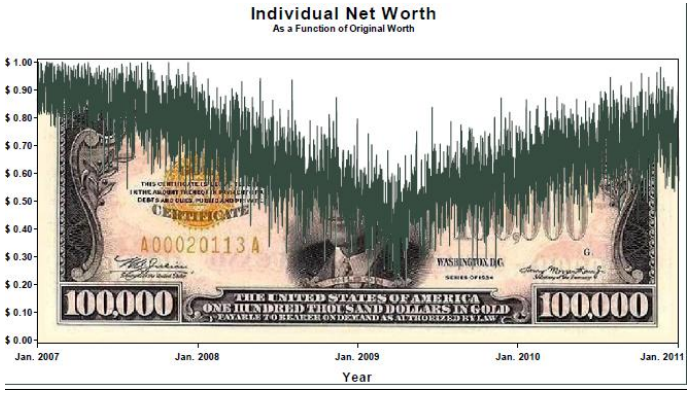
Na adrese http://support.sas.com/sassamples/graphgallery/PROC_GPLOT.html lze nalézt galerii možných typů grafů (**včetně kódů**!). Na adrese http://ebookbrowse.com/sas-gplot-slides-1-26-2011-ppt-d138883835 lze najít další návody a ukázky včetně kódů.











ODS Graphics

- Recall that the Output Delivery System (ODS), added in Version 8, manages all tabular output created by procedures and enables you to display it in a variety of destinations, such as HTML and RTF. SAS 9.1 introduced an extension to ODS, referred to as ODS Graphics, which—together with corresponding modifications to statistical procedures—equips these procedures to create graphics as automatically as tables. This eliminates the need for additional programming.

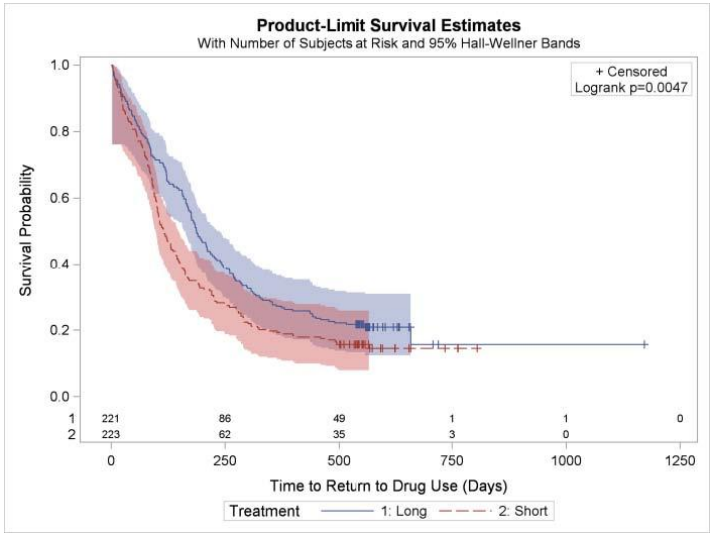
- Automaticky vytvářet grafické výstupy umí tyto procedury:

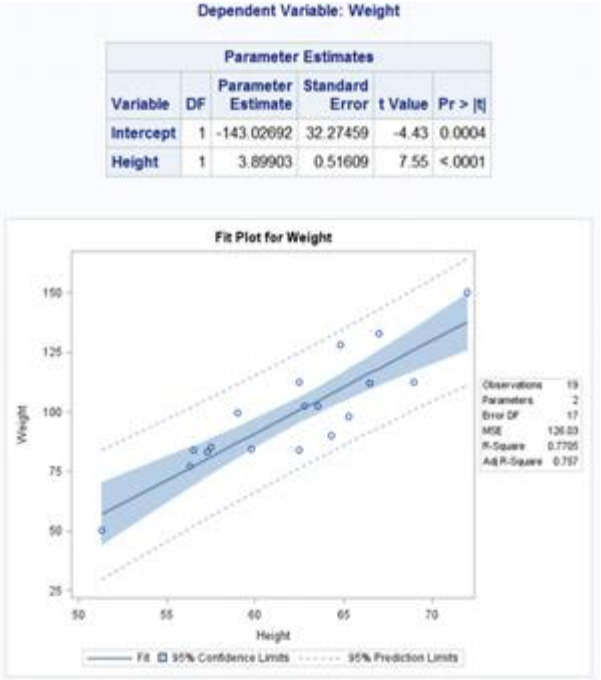
Base SAS	SAS/STAT			SAS/QC	SAS/ETS
CORR	ANOVA	KDRIGE2D	PRINCOMP	ANOM	ARIMA
FREQ	BOXPLOT	LIFEREG	PRINQUAL	CAPABILITY	AUTOREG
UNIVARIATE	CALIS	LIFETEST	PROBIT	CUSUM	ENTROPY
	CLUSTER	LOESS	QUANTREG	MACONTROL	EXPAND
	CORRESP	LOGISTIC	REG	PARETO	MODEL
	FACTOR	MCMC	ROBUSTREG	RELIABILITY	PANEL
	FREQ	MDS	RSREG	SHEWHART	RISK
	GAM	MI	SEQDESIGN		SIMILARITY
	GENMOD	MIXED	SEQTEST		SYSLIN
	GLIMMIX	MULTTEST	SIM2D		TIMESERIES
	GLM	NPAR1WAY	TCALIS		UCM
	GLMSELECT	PHREG	TRANSREG		VARMAX
	KDE	PLS	TTEST		X12
			VARIOGRAM		

Více viz:
1) http://support.sas.com/rnd/base/topics/statgraph/
2) http://susanslaughter.files.wordpress.com/2011/06/svsug-2011-handout_for_getting_started_with_ods_statistical_graphics.pdf
3) http://www2.sas.com/proceedings/sugi31/192-31.pdf

ODS Graphics

```
ods listing style=statistical;
ods graphics on;
proc lifetest data=grouped plots=survival(cb=hw test
atrisk=0 to 1500 by 250); time Time*Cens(0); strata
Treatment; by Site;
run;
```

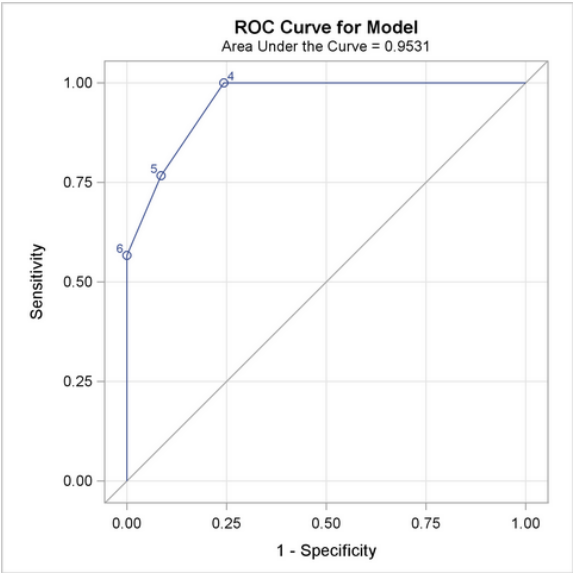


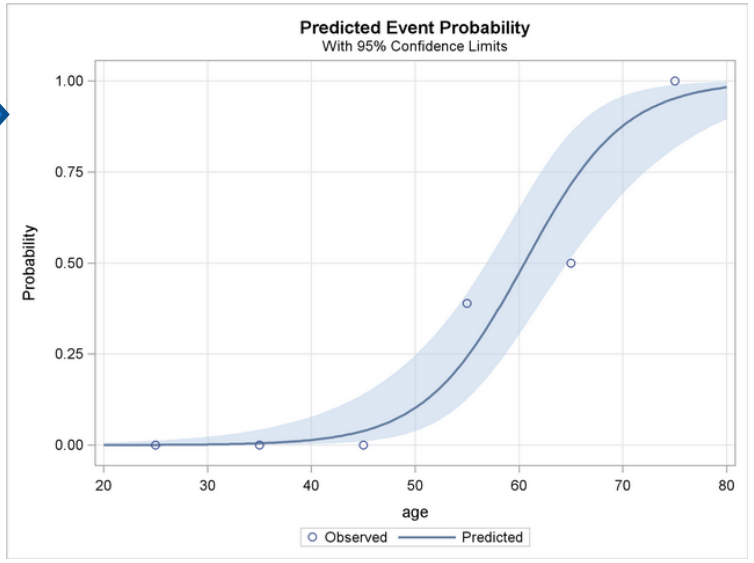



```
ods listing style=statistical;
ods graphics on;
proc reg data=sashelp.Class;
  model Weight=Height;
quit;
```

ODS Graphics

```
data Data1;
 input disease n age; datalines;
 0 14 25
 0 20 35
 0 19 45
 7 18 55
 6 12 65
17 17 75
 ;
run;

ods graphics on;
proc logistic data=Data1 plots(only)=(roc(id=obs) effect);
 model disease/n=age
/ scale=none clparm=wald clodds=pl rsquare; units age=10;
run;
ods graphics off;
```





Více viz PLOTS options u PROC Logistic: http://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_logistic_sect004.htm#statug.logistic.logiploteffect nebo na adrese: http://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_logistic_sect050.htm

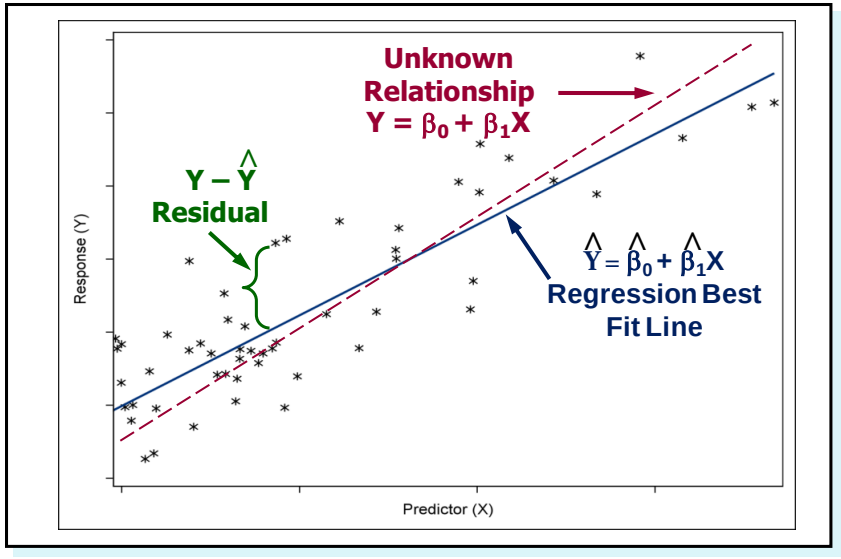
SAS makra – M.Friendly

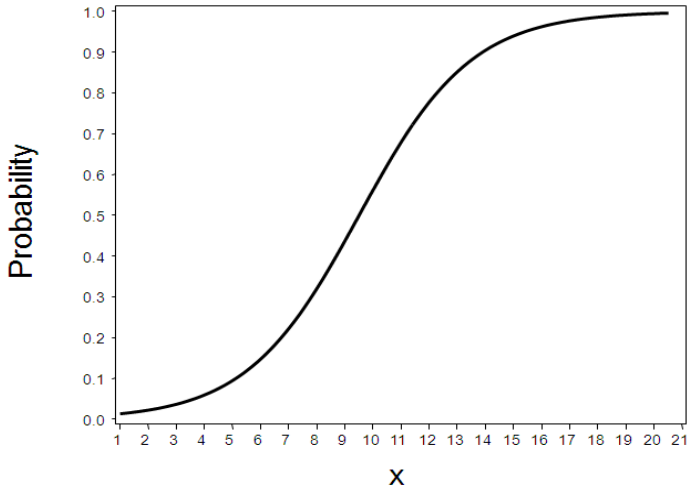
- **Michael Friendly,York University:**
SAS Graphic Programs and Macros 

  - Univariate displays
  - Bivariate displays
  - Multivariate displays
  - Cluster analysis
  - Maps

http://www.datavis.ca/sasmac/

9. Regrese. Logistická regrese





Overview

	Type of Predictors		
Type of Response	**Categorical**	**Continuous**	**Categorical and Continuous**
Continuous	Analysis of Variance	Linear Regression	Analysis of Covariance (Regression with dummy variables)
Categorical	Logistic Regression or Contingency Tables	Logistic Regression	Logistic Regression

Přehled procedur SASu pro regresi

- SAS/STAT:

  CATMOD, GAM, GENMOD, GLIMMIX, GLM, LIFEREG, LOESS, LOGISTIC, MIXED, NLIN, NLMIXED, ORTHOREG, PHREG, PLS, PROBIT, REG, ROBUSTREG, RSREG, SURVEYLOGISTIC, SURVEYPHREG, SURVEYREG, TRANSREG.

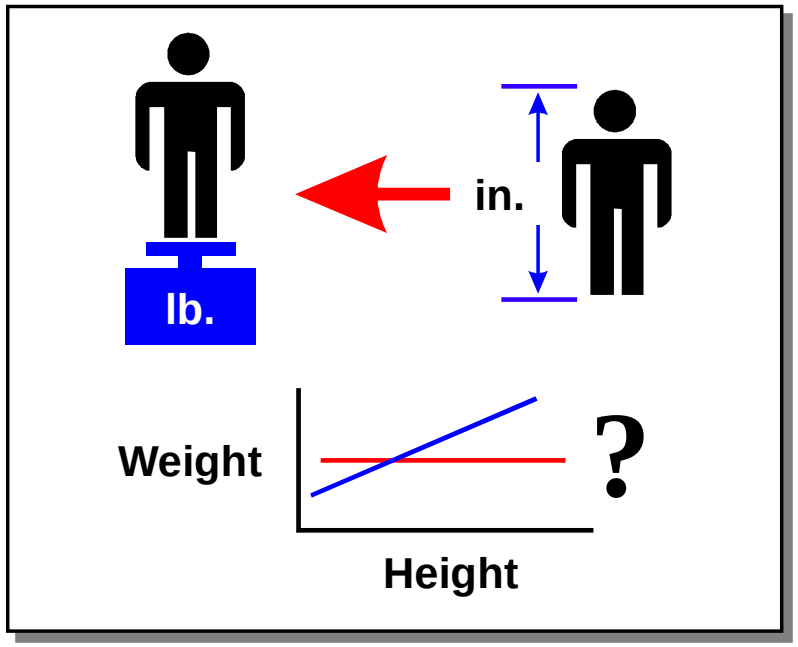
logistická regrese

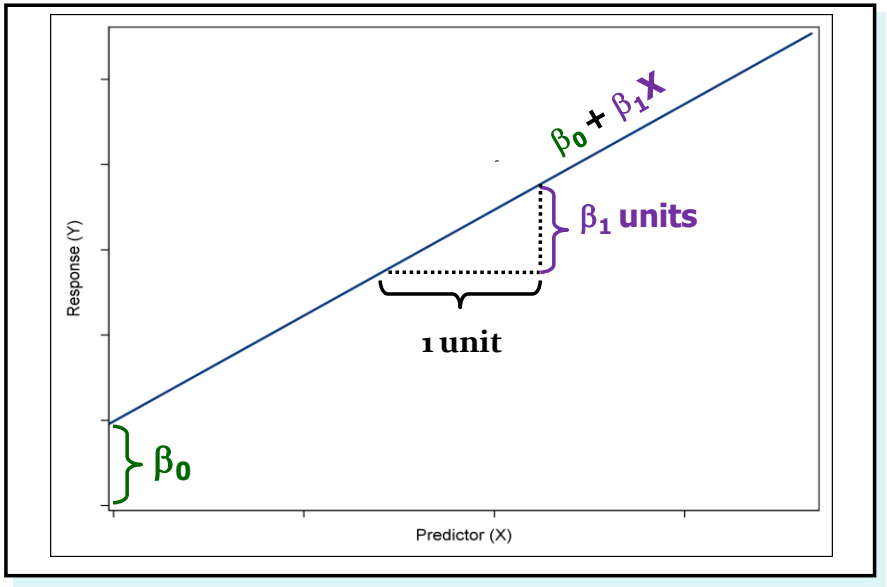
„klasická“ lineární regrese

- SAS/ETS:

  AUTOREG, COUNTREG, MODEL, PANEL, PDLREG, SYSLIN.

Simple Linear Regression Model





The CORR Procedure

- S regresní analýzou souvisí analýza korelační.
- Když pro nic jiného, tak alespoň v souvislosti s explorační analýzou je vhodné prozkoumat data pomoví procedury CORR.
- General form of the CORR procedure:

```
PROC CORR DATA=SAS-data-set <options>;
        VAR variables;
        WITH variables;
        ID variables;
RUN;
```

The CORR Procedure

- Scatter plots and scatter plot matrices are available through ODS Graphics.
- ID statement enables you to specify additional variables to identify observations in scatter plots and scatter plot matrices.
- Selected options:
  - **PLOTS** <(**ONLY**)> <= *plot-request*>
  - **PLOTS** <(**ONLY**)> <= (*plot-request* < *plot-request* >) >
    - **ALL**
    - **MATRIX** <(*matrix-options*)>
    - **SCATTER** <(*scatter-options*)>
    - **HIST** | **HISTOGRAM**
    - **NVAR=ALL** | *n*
    - **ELLIPSE=PREDICTION** | **CONFIDENCE** | **NO**

PROC CORR –příklad výstupu

Correlations and Scatter Plots with Oxygen_Consumption

The CORR Procedure

1 With Variables:	Oxygen_Consumption	
7 Variables:	RunTime Performance	Age Weight Run_Pulse Rest_Pulse Maximum_Pulse

Simple Statistics

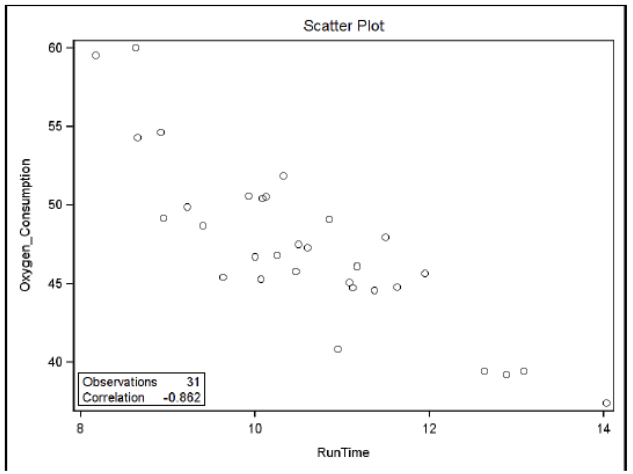
Variable	*N*	*Mean*	*Std Dev*	*Sum*	*Minimum*	*Maximum*
Oxygen_Consumption	31	47.37581	5.32777	1469	37.39000	60.06000
RunTime	31	10.58613	1.38741	328.17000	8.17000	14.03000
Age	31	47.67742	5.26236	1478	38.00000	57.00000
Weight	31	77.44452	8.32857	2401	59.08000	91.63000
Run_Pulse	31	169.64516	10.25199	5259	146.00000	186.00000
Rest_Pulse	31	53.45161	7.61944	1657	40.00000	70.00000
Maximum_Pulse	31	173.77419	9.16410	5387	155.00000	192.00000
Performance	31	56.64516	18.32584	1756	20.00000	94.00000

Pearson Correlation Coefficients, N = 31
Prob > |r| under H0: Rho=0

Oxygen_Consumption	RunTime	Performance	Rest_Pulse	Run_Pulse	Age	Maximum_Pulse	Weight
	-0.86219	0.77890	-0.39935	-0.39808	-0.31162	-0.23677	-0.16289
	<.0001	<.0001	0.0260	0.0266	0.0879	0.1997	0.3813

Correlations and Scatter Plots with Oxygen_Consumption

The CORR Procedure



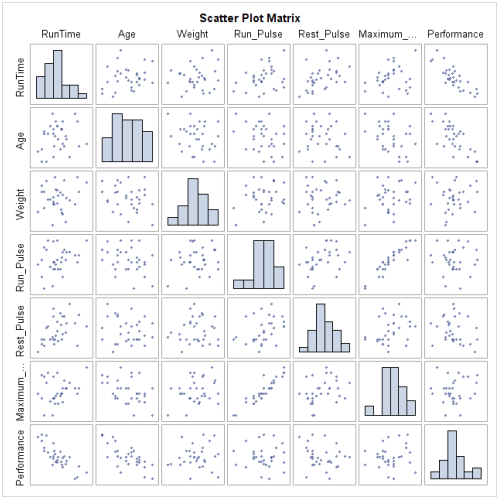
Correlations and Scatter Plot Matrix of Fitness Predictors

The CORR Procedure

7 Variables:	RunTime Age Weight Run_Pulse Rest_Pulse Maximum_Pulse Performance

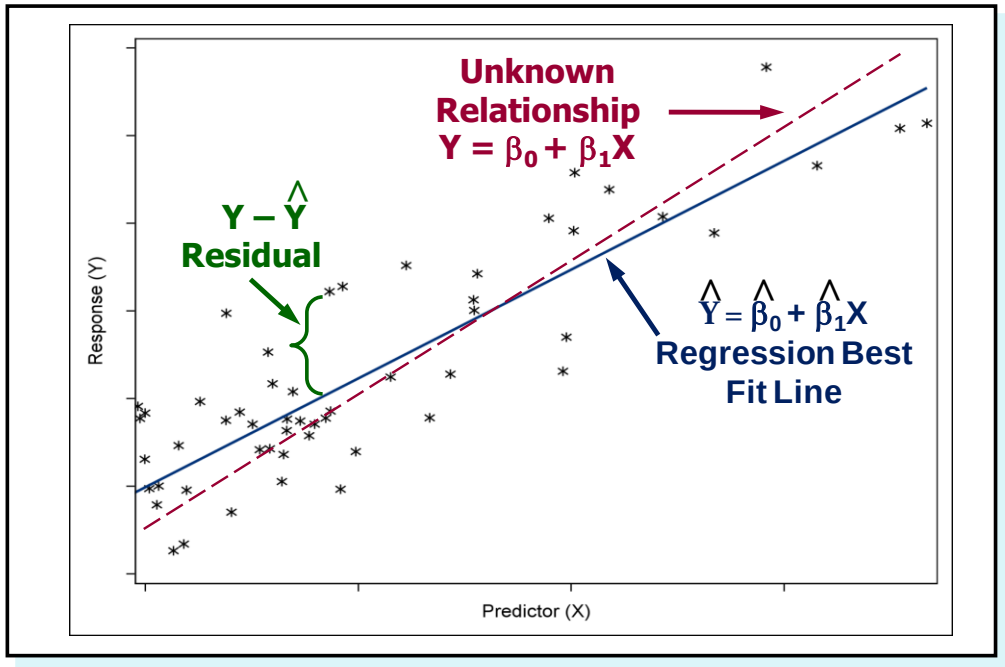
Pearson Correlation Coefficients, N = 31
Prob > |r| under H0: Rho=0

	RunTime	Age	Weight	Run_Pulse	Rest_Pulse	Maximum_Pulse	Performance
RunTime	1.00000	0.19523	0.14351	0.31365	0.45038	0.22610	-0.82049
		0.2926	0.4412	0.0858	0.0110	0.2213	<.0001
Age	0.19523	1.00000	-0.24050	-0.31607	-0.15087	-0.41490	-0.71257
	0.2926		0.1925	0.0832	0.4178	0.0203	<.0001
Weight	0.14351	-0.24050	1.00000	0.18152	0.04397	0.24938	0.08974
	0.4412	0.1925		0.3284	0.8143	0.1761	0.6312
Run_Pulse	0.31365	-0.31607	0.18152	1.00000	0.35246	0.92975	-0.02943
	0.0858	0.0832	0.3284		0.0518	<.0001	0.8751
Rest_Pulse	0.45038	-0.15087	0.04397	0.35246	1.00000	0.30512	-0.22560
	0.0110	0.4178	0.8143	0.0518		0.0951	0.2224
Maximum_Pulse	0.22610	-0.41490	0.24938	0.92975	0.30512	1.00000	0.09002
	0.2213	0.0203	0.1761	<.0001	0.0951		0.6301
Performance	-0.82049	-0.71257	0.08974	-0.02943	-0.22560	0.09002	1.00000
	<.0001	<.0001	0.6312	0.8751	0.2224	0.6301	



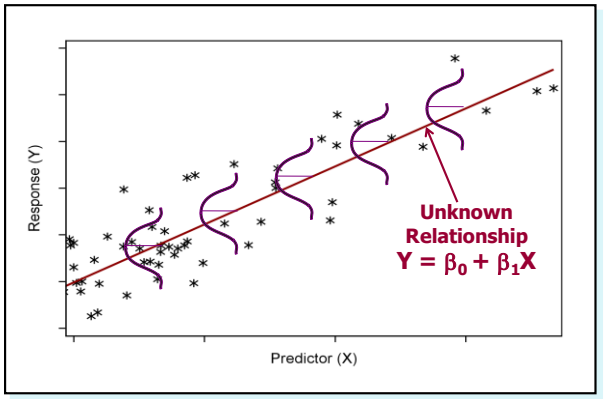
Simple Linear Regression Model

$$Y = \beta_0 + \beta_1 X_1 + \varepsilon$$



Assumptions:

- **The mean of the Ys is accurately modeled by a linear function of the Xs.**
- **The random error term, ε, is assumed to have a normal distribution with a mean of zero.**
- **The random error term, ε, is assumed to have a constant variance, σ^2.**
- **The errors are independent.**



Model Hypothesis Test

- **Null Hypothesis:**
  - The simple linear regression model does not fit the data better than the baseline model.
  - $\beta_1 = 0$
- **Alternative Hypothesis:**
  - The simple linear regression model does fit the data better than the baseline model.
  - $\beta_1 \neq 0$

Multiple Linear Regression with Two Variables

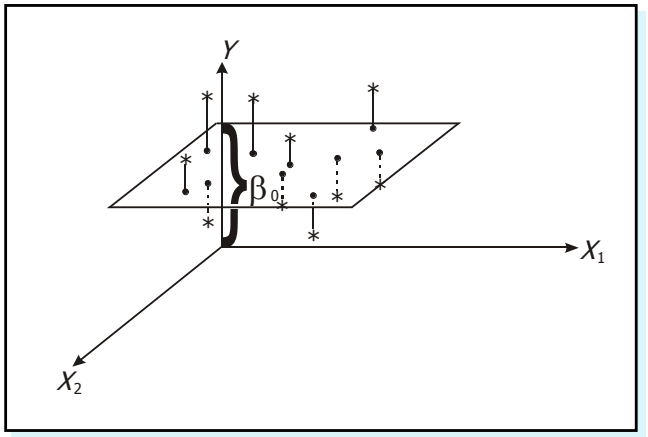
- Consider the two-variable model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon$$

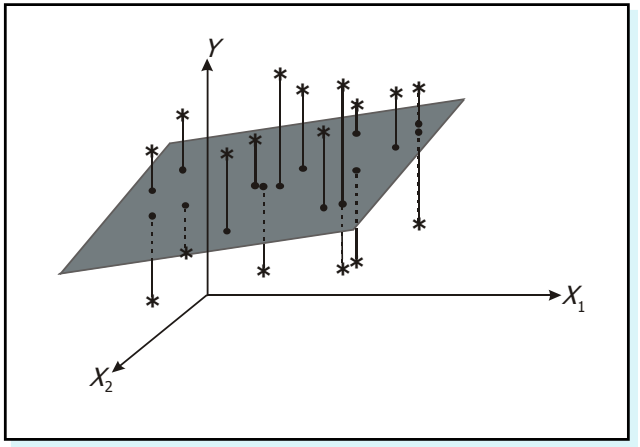
where

Y	is the dependent variable.
X_1 and X_2	are the independent or predictor variables.
ε	is the error term.
β_0, β_1, and β_2	are unknown parameters.

No relationship:



A relationship:



The Multiple Linear Regression Model

- In general, you model the dependent variable Y as a linear function of *k* independent variables, (the Xs) as

$$Y = \beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k + \varepsilon$$

Model Hypothesis test:

- **Null Hypothesis:**
  - The regression model does not fit the data better than the baseline model.
  - $\beta_1 = \beta_2 = \ldots = \beta_k = 0$

- **Alternative Hypothesis:**
  - The regression model does fit the data better than the baseline model.
  - Not all β_is equal zero.

Prediction

- The terms in the model, the values of their coefficients, and their statistical significance are of secondary importance.
- The focus is on producing a model that is the best at predicting future values of Y as a function of the Xs. The predicted value of Y is given by

$$\underline{\hat{Y}} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \ldots + \hat{\beta}_k X_k$$

Analytical or Explanatory Analysis

- The focus is on understanding the relationship between the dependent variable and the independent variables.
- Consequently, the statistical significance of the coefficients is important as well as the magnitudes and signs of the coefficients.

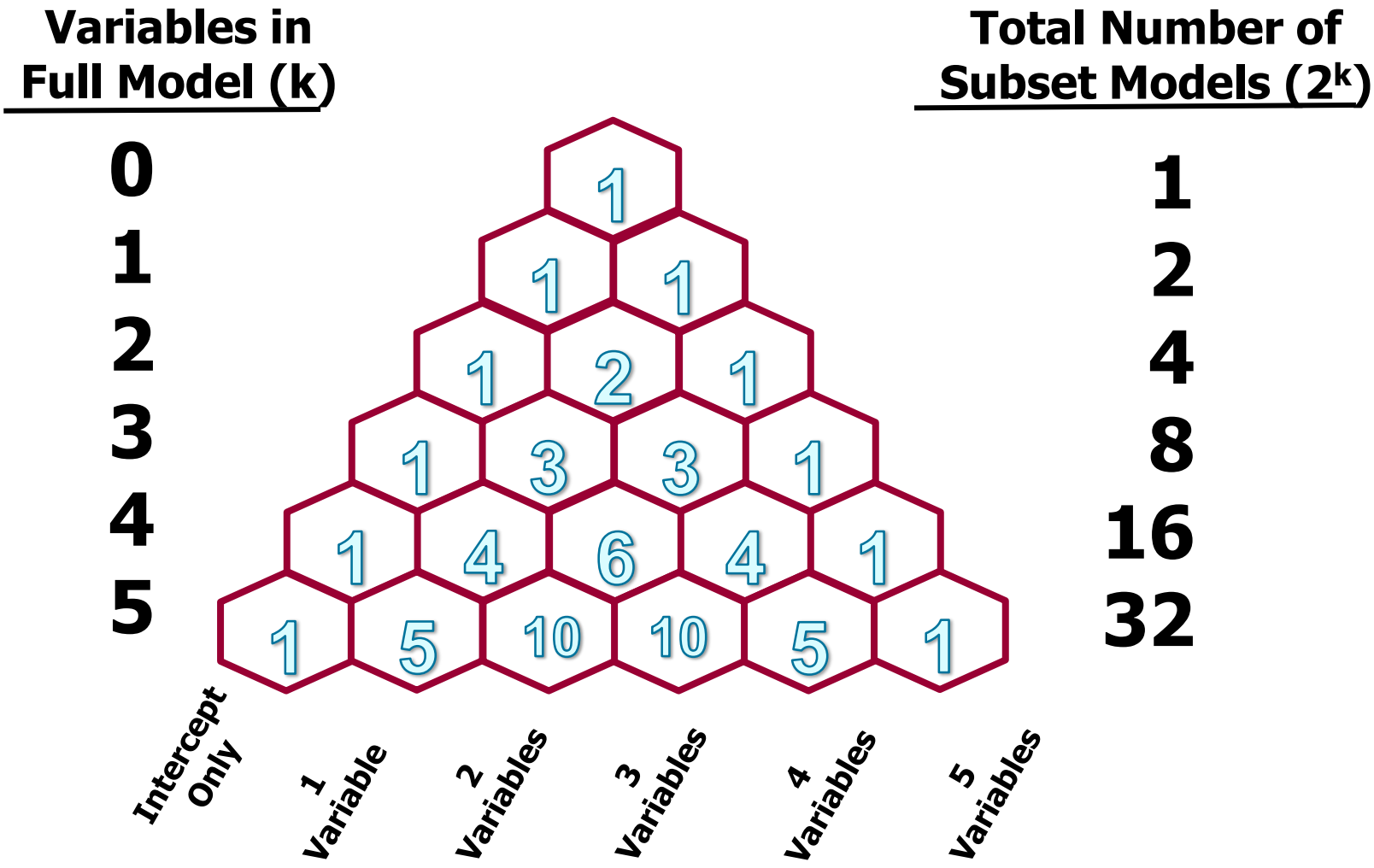
$$\hat{Y} = \underline{\hat{\beta}_0} + \underline{\hat{\beta}_1} X_1 + \ldots + \underline{\hat{\beta}_k} X_k$$

Model Selection Options

- The SELECTION= option in the MODEL statement of PROC REG supports these model selection techniques:
- **All-possible regressions ranked using**
  - RSQUARE, ADJRSQ or CP
- **Stepwise selection methods**
  - STEPWISE, FORWARD, or BACKWARD
-

SELECTION=NONE is the default.

RSQUARE, ADJRSQ, CP Selection Options



Mallows' C_p

- Mallows' C_p is a simple indicator of model bias. Models with a large C_p are biased.
- Look for models with $C_p \leq p$, where p equals the number of parameters in the model, including the intercept.

- Mallows recommends choosing the first (fewest variables) model where C_p approaches p.

Conservative Significance Levels

	Sample Size			
Evidence	**30**	**50**	**100**	**1000**
Weak	.076	.053	.032	.009
Positive	.028	.019	.010	.003
Strong	.005	.003	.001	.0003
Very Strong	.001	.0005	.0001	.00004

The REG Procedure

- General form of the REG procedure:

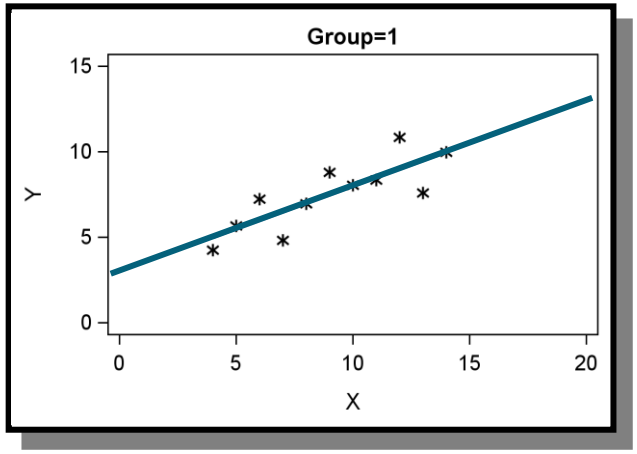
```
PROC REG DATA=SAS-data-set <options>;
      MODEL dependent(s)=regressor(s) </ options>;
RUN;
```

Popis + jednoduchý příklad:

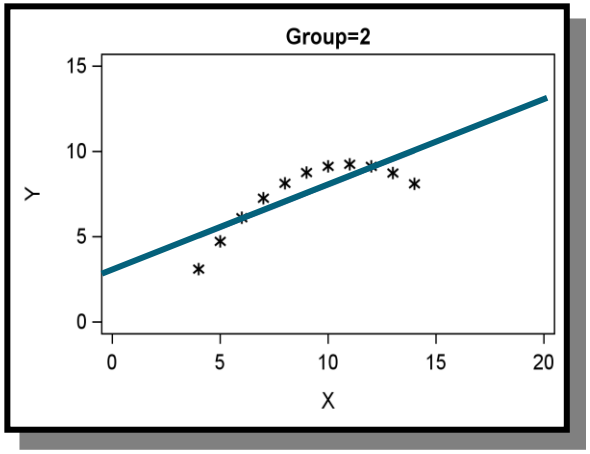
http://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_reg_sect003.htm

Analýza reziduí

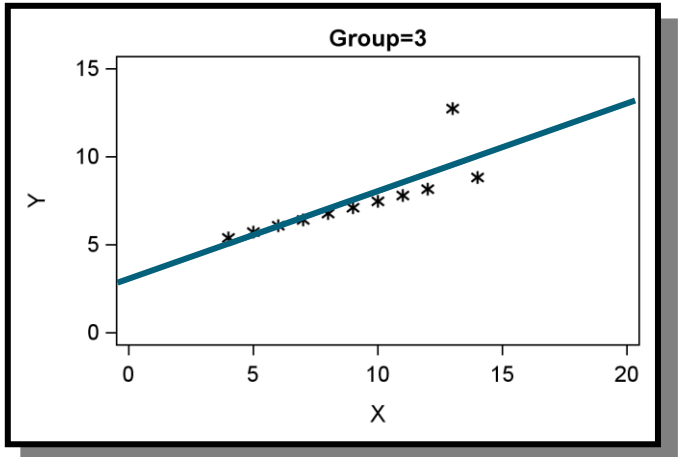
Scatter Plot of Correct Model



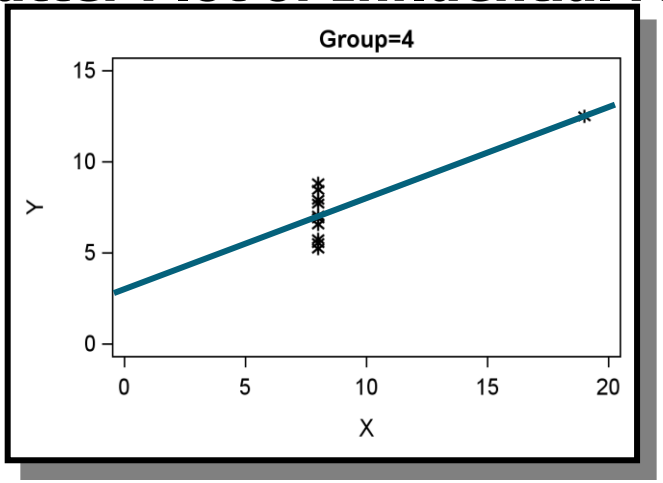
Scatter Plot of Curvilinear Model



Scatter Plot of Outlier Model

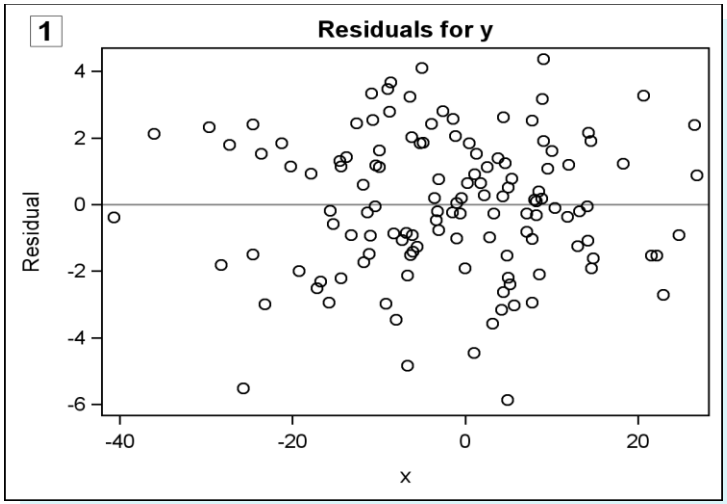


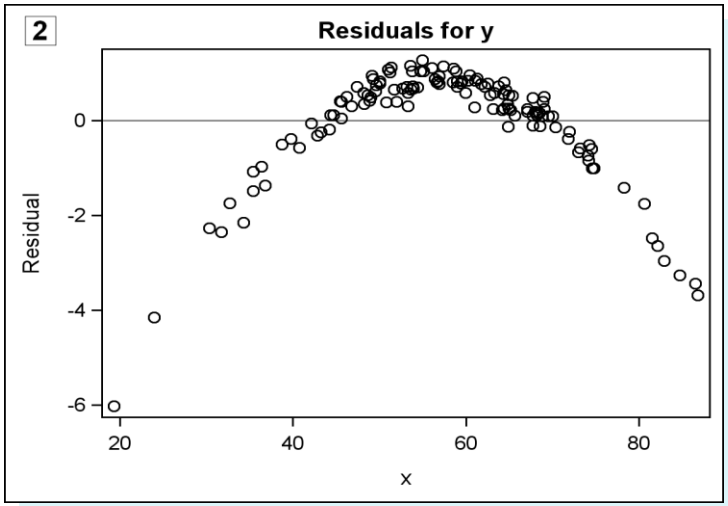
Scatter Plot of Influential Model

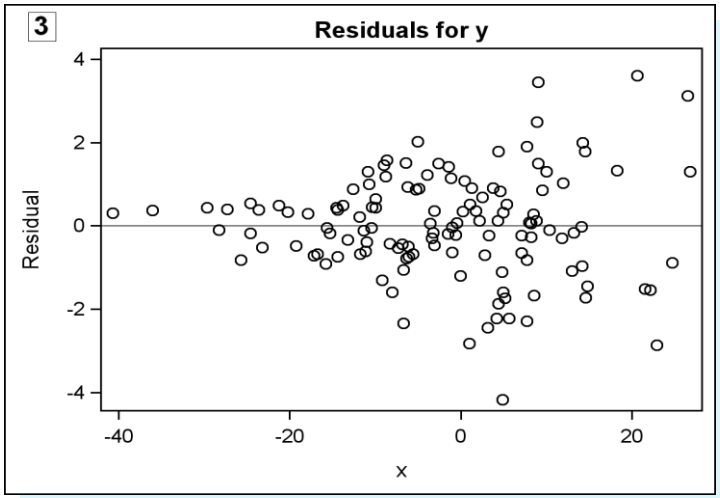


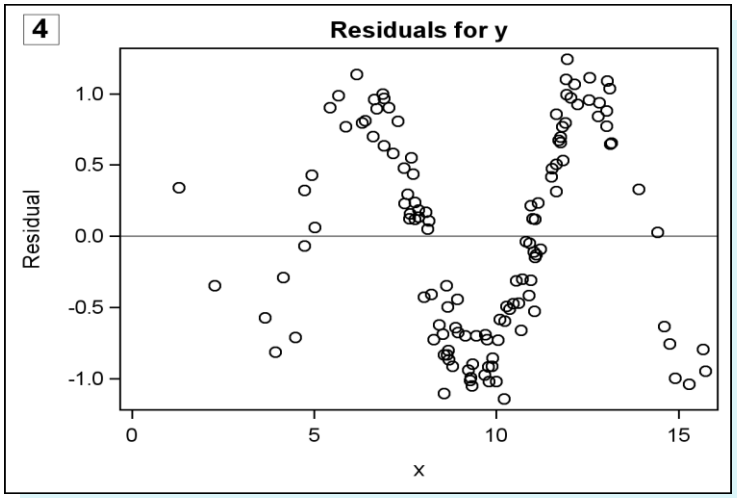
- $Y = 3.0 + 0.5X$
- $R^2 = 0.67$

Examining Residual Plots









Diagnostic Statistics

- Four statistics that help identify influential observations are
  - STUDENT residual
  - Cook's D
  - RSTUDENT residual
  - DFFITS.

Cook's D Statistic

- Cook's D statistic is a measure of the simultaneous change in the parameter estimates when an observation is deleted from the analysis.

- A suggested cutoff is $D_i > \frac{4}{n}$, where n is the sample size.

- If the above condition is true, then the observation might have an adverse effect on the analysis.

Studentized Residual

- Studentized residuals (SR) are obtained by dividing the residuals by their standard errors.
- Suggested cutoffs are as follows:
  - |SR| > 2 for data sets with a relatively small number of observations
  - |SR| > 3 for data sets with a relatively large number of observations

DFFITS

- DFFITS_i measures the impact that the i^{th} observation has on the predicted value.
-

$$\text{DFFITS}_i = \frac{\hat{Y}_i - \hat{Y}_{(i)}}{s(\hat{Y}_i)}$$

$\hat{Y}_i$ is the i^{th} predicted value.

$\hat{Y}_{(i)}$ is the i^{th} predicted value when the i^{th} observation is deleted.

$s(\hat{Y}_i)$ is the standard error of the i^{th} predicted value.

Lineární regrese – PROC REG

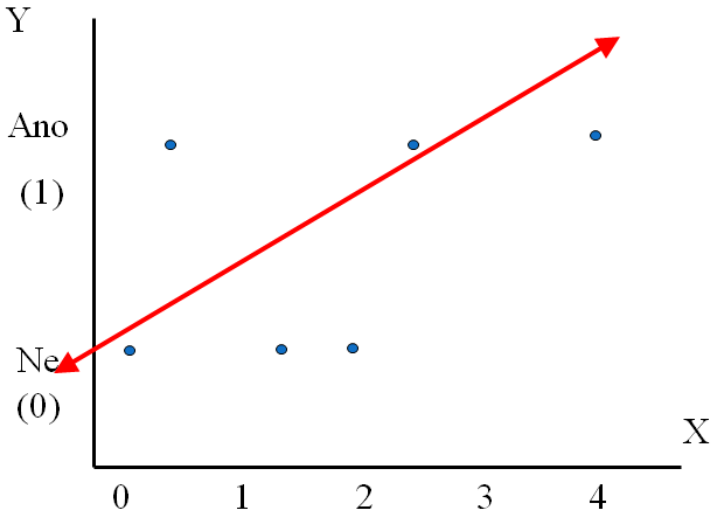
PROC REG <options> ;
<label:>MODEL dependents=<regressors> </ options> ;
BY variables ;
FREQ variable ;
ID variables ;
VAR variables ;
WEIGHT variable ;
ADD variables ;
DELETE variables ;
<label:>MTEST <equation, ...,equation> </ options> ;
OUTPUT <OUT=SAS-data-set>< keyword=names> <...keyword=names> ;
PAINT <condition | ALLOBS> </ options > | < STATUS | UNDO> ;
RESTRICT equation, ...,equation ;
REWEIGHT <condition | ALLOBS> </ options > | < STATUS | UNDO> ;
PLOT <yvariable*xvariable> <=symbol> <...yvariable*xvariable> <=symbol> </ options> ;
PRINT <options> <ANOVA> <MODELDATA> ;
REFIT ;
RESTRICT equation, ...,equation ;
REWEIGHT <condition | ALLOBS> </ options > | < STATUS | UNDO> ;
<label:>TEST equation,<,...,equation> </ option> ;

Více na: http://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_reg_sect001.htm

Modelování kategoriální responze

- Nastane default?

St.	X	Y
1	2.6	1
2	1.4	0
3	.65	1
4	4.1	1
5	.25	0
6	1.9	0



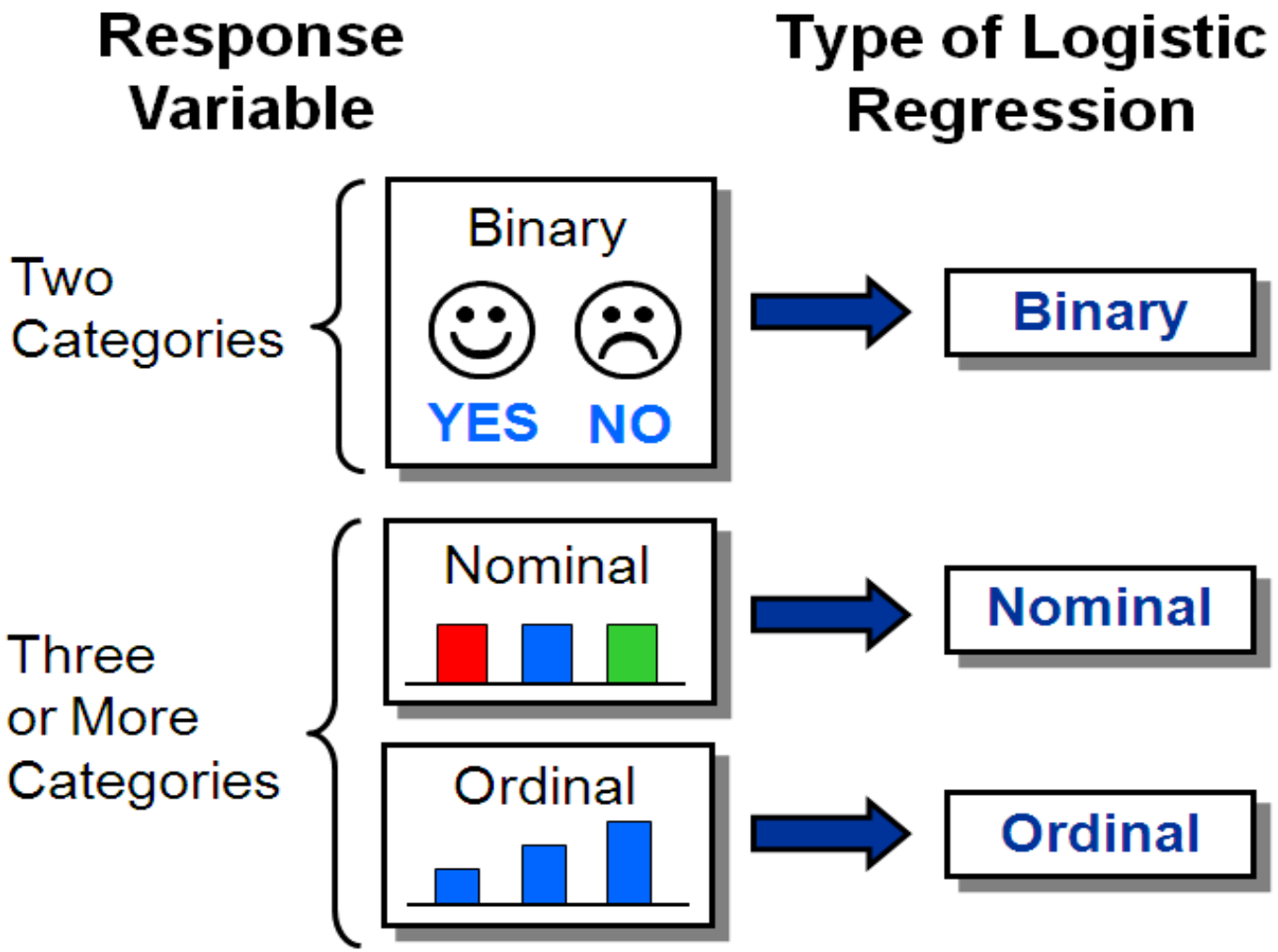
$Y_i = \beta_o + \beta_1 X_{1i} + \varepsilon_i$

„klasická“ regrese není vhodná

➡ používá se logistická regrese.

- If the response variable is categorical, then how do you code the response numerically?
- If the response is coded (1=Yes and 0=No) and your regression equation predicts 0.5 or 1.1 or -0.4, what does that mean practically?
- If there are only two (or a few) possible response levels, is it reasonable to assume constant variance and normality?

Types of Logistic Regression



Logit Transformation

- Logistic regression models transform probabilities called logits*.

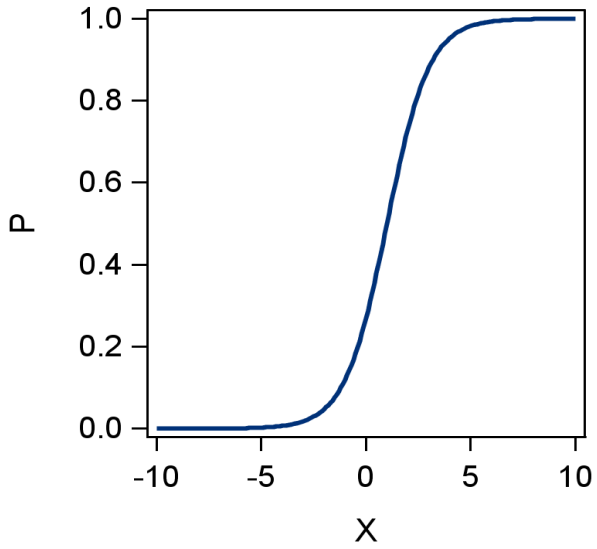
$$\text{logit}(p_i) = \ln\left(\frac{p_i}{(1-p_i)}\right)$$

where

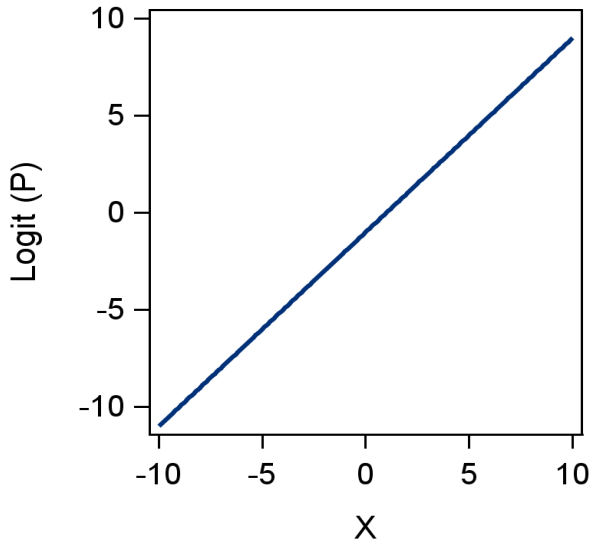
i	indexes all cases (observations)
p_i	is the probability the event (a default, for example) occurs in the i^{th} case
ln	is the natural log (to the base e).

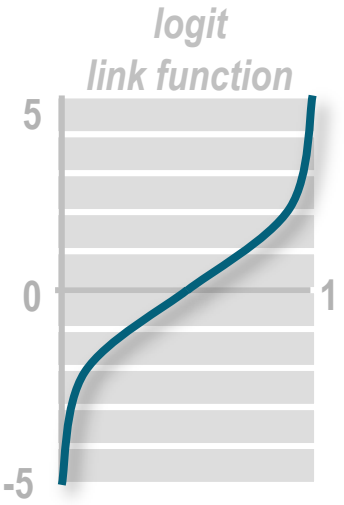
* The logit is the natural log of the odds.

Logit link function



Logit Transform





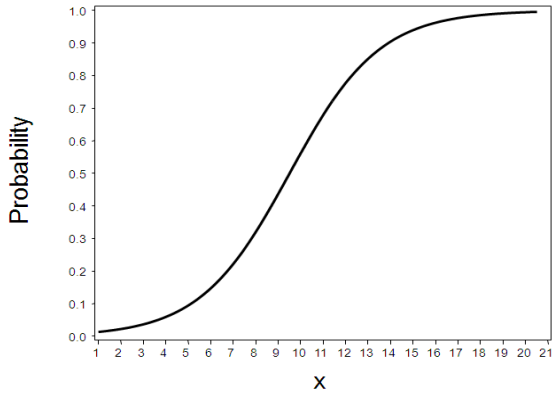
The logit link function transforms probabilities (between 0 and 1) to logit scores (between $-\infty$ and $+\infty$).

Logistic Regression Model

$$\text{logit}(p_i) = \beta_o + \beta_1 X_1 + \ldots + \beta_k X_k$$

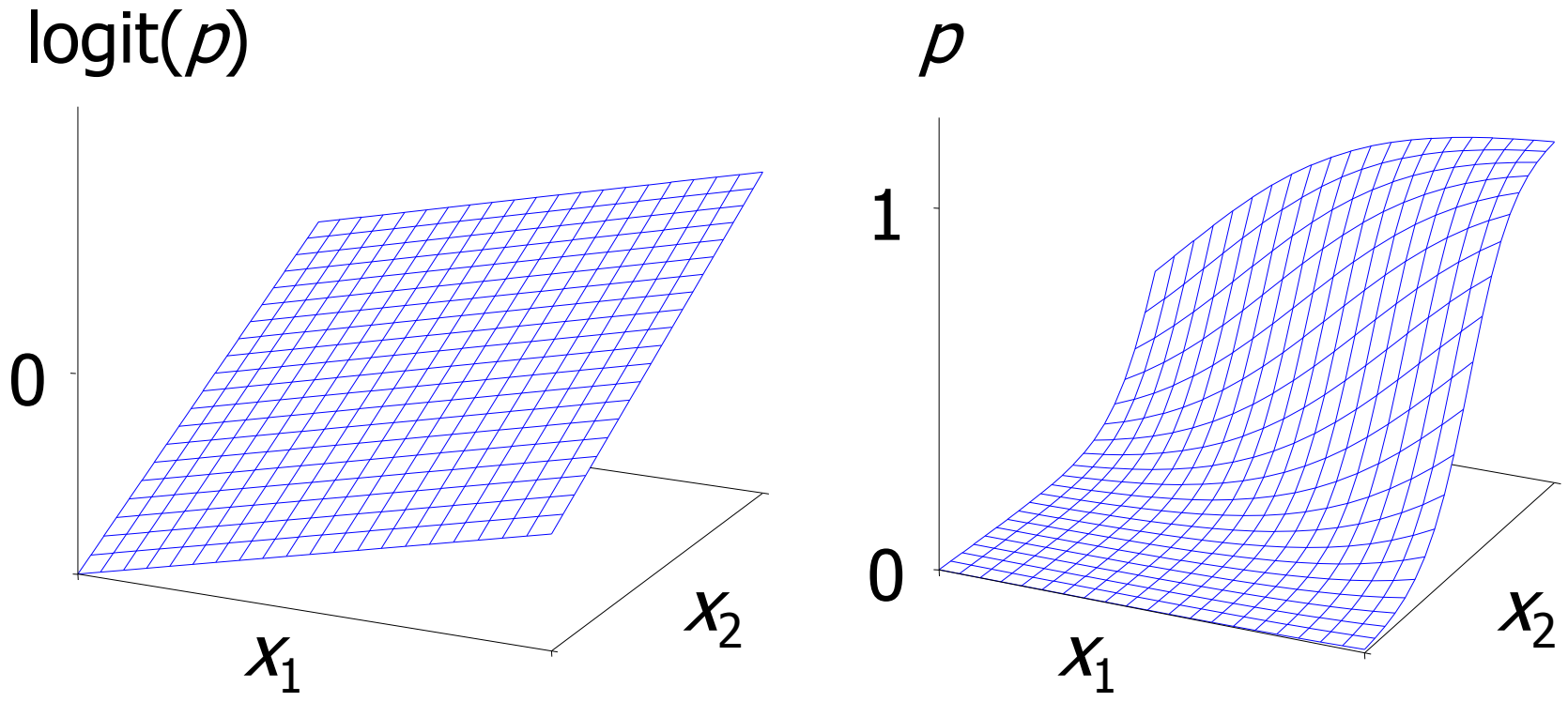
where

- logit (p_i)= logit of the probability of the event
- β_o=intercept of the regression equation
- β_k=parameter estimate of the k^{th} predictor variable

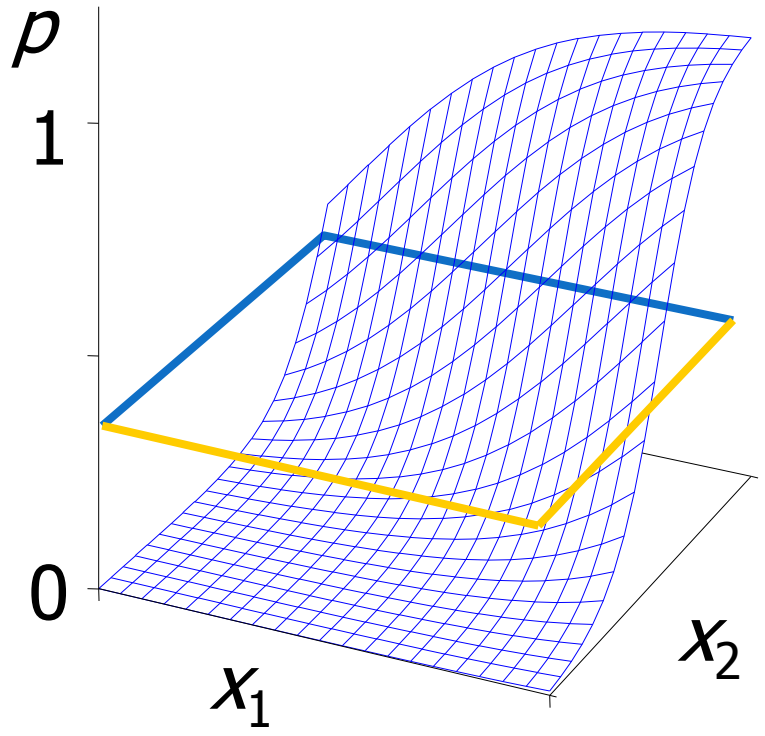


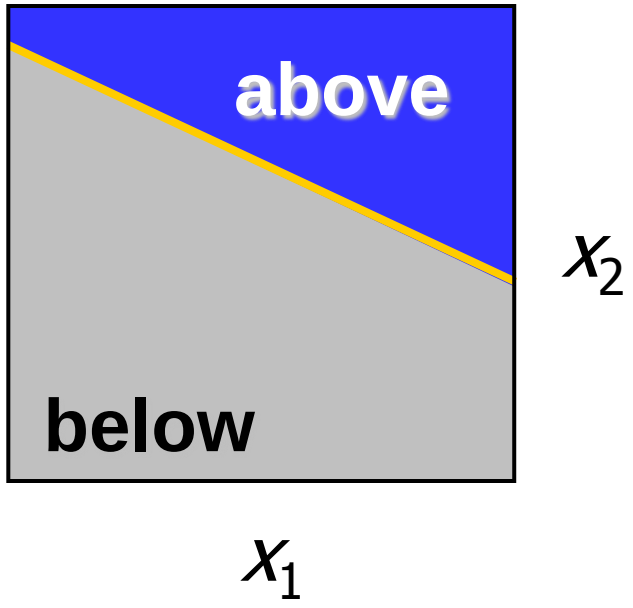
$$p_i = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \ldots + \beta_k X_k)}}$$

The Fitted Surface



Logistic Discrimination





Odhad parametrů

- Metoda maximální věrohodnosti vede na soustavu nelineárních rovnic.
- Tuto soustavu řešíme Netwon-Raphsonovou iterační metodou.
- Více na:
  - http://www.stat.cmu.edu/~cshalizi/402/lectures/14-logistic-regression/lecture-14.pdf
  - http://czep.net/stat/mlelr.pdf
  - http://www.stat.psu.edu/~jiali/course/stat597e/notes2/logit.pdf

Maximálně věrohodný odhad (MLE)

MLE is a general purpose method for parametric model estimation.
We will make use of it to estimate the logistic regression.

If we have a model with parametric structure $\boldsymbol{\theta}$, we can compute the **likelihood** that the model will generate a sequence of n observations $\mathbf{D} = (d_1, \ldots, d_n)$.

$$L(\boldsymbol{\theta}|\mathbf{D}) = P(\mathbf{D}|\boldsymbol{\theta})$$

The model which best fits the data is selected as the one which maximizes this likelihood.

$$\widehat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} L(\boldsymbol{\theta}|\mathbf{D})$$

If we *assume independence between the observations*, this then gives

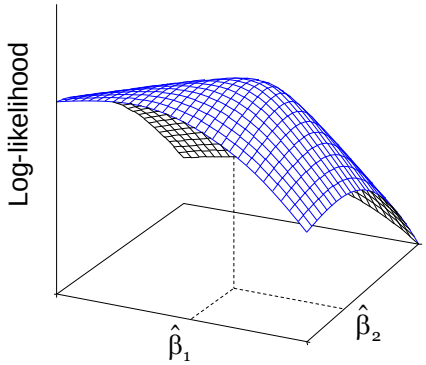
$$\widehat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} P(d_i|\boldsymbol{\theta})$$

Zdroj: http://www2.imperial.ac.uk/~abellott/Credit%20Scoring%202.pdf

Maximálně věrohodný odhad

This MLE can be expressed more conveniently in terms of log-likelihoods (since log is monotonic on its argument):

$$\widehat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log P(d_i|\boldsymbol{\theta})$$



Remember:

- We do not know the true value of the parameter $\boldsymbol{\theta}$, but we want to estimate it.
- To distinguish the estimate from the true value, in our notation, we put a "hat" on the estimate: $\widehat{\boldsymbol{\theta}}$.

MLE has several nice asymptotic properties:

- Consistency
- Asymptotic normality
- Efficiency.

Maximálně věrohodný odhad

Consider the training data set D_{train} with n observations (borrowers).

Remember

- $\mathbf{x}_i$ denotes values for predictor variables for observation i.
- y_i denotes the outcome for observation i, either 0 or 1.

Then the likelihood of the outcome for each observation i is given by

$$\begin{array}{ll} P(y_i = 0|\mathbf{x}_i, \boldsymbol{\beta}) & \text{if } y_i\text{=0,} \\ 1 - P(y_i = 0|\mathbf{x}_i, \boldsymbol{\beta}) & \text{if } y_i\text{=1} \end{array}$$

which is

$$P(y_i = 0|\mathbf{x}_i, \boldsymbol{\beta})^{1-y_i}\big(1 - P(y_i = 0|\mathbf{x}_i, \boldsymbol{\beta})\big)^{y_i}$$

giving log-likelihood for each observation:

$$(1 - y_i)\log P(y_i = 0|\mathbf{x}_i, \boldsymbol{\beta}) + y_i \log\big(1 - P(y_i = 0|\mathbf{x}_i, \boldsymbol{\beta})\big)$$

Maximálně věrohodný odhad

Assuming independence between observations, this gives the log-likelihood function for $\boldsymbol{\beta}$:

$$\log L(\boldsymbol{\beta}|D_{\text{train}}) = \sum_{i=1}^{n}(1-y_i)\log\left(\frac{1}{1+e^{-(\beta_0+\boldsymbol{\beta}\cdot\mathbf{x}_i)}}\right) + y_i \log\left(\frac{1}{1+e^{\beta_0+\boldsymbol{\beta}\cdot\mathbf{x}_i}}\right)$$

Differentiating by each coefficient in $\boldsymbol{\beta}$ and setting the derivative equal to zero to find the maxima gives

$$\sum_{i=1}^{n}\left(1-y_i-\left(\frac{1}{1+e^{-(\beta_0+\boldsymbol{\beta}\cdot\mathbf{x}_i)}}\right)\right)=0$$

and

$$\sum_{i=1}^{n}x_{ij}\left(1-y_i-\left(\frac{1}{1+e^{-(\beta_0+\boldsymbol{\beta}\cdot\mathbf{x}_i)}}\right)\right)=0$$

for each attribute j=1 to m.

These are non-linear equations that can be solved by computer intensive processes such as Newton-Raphson methods.

Standard errors on the MLE

Since $\widehat{\boldsymbol{\theta}}$ is only an estimate of the best model to explain the data, it is possible to derive standard errors $\hat{\boldsymbol{s}}$ on the estimates.

Asymptotic normality for MLE is such that

$$\frac{(\hat{\theta}_j - \theta_j)}{\hat{s}_j} \rightarrow N(0,1) \text{ as } n \rightarrow \infty$$

where $\hat{\theta}_j$, θ_j and $\hat{s}_j$ are the jth components of $\widehat{\boldsymbol{\theta}}$, $\boldsymbol{\theta}$ and $\hat{\boldsymbol{s}}$ respectively and $N(0,1)$ is the standard normal distribution.

This property then allows us to generate:-

- Generate a hypothesis tests using the Wald chi-square statistic;
- Generate confidence intervals around the estimate.

MLE- testování hypotéz

We test the hypothesis that an estimated coefficient is not zero against the null hypothesis that it is zero. That is, we testing if a parameter has a genuine effect in the model.

- Null hypothesis: $H_0: \theta_j = 0$
- Alternative hypothesis: $H_1: \theta_j \neq 0$

The Wald test says reject H_0 if $\frac{|\hat{\theta}_j|}{\hat{s}_j} > z_{\alpha/2}$ for some significance level α, where $z_{\alpha/2} = \Phi^{-1}(1 - \alpha/2)$ and Φ is the CDF for the standard normal distribution.

MLE – konfidenční intervaly

The asymptotic normality property also allows us to compute confidence intervals (CIs):

$$P\left(\hat{\theta}_j - z_{\alpha/2}\hat{s}_j < \theta_j < \hat{\theta}_j + z_{\alpha/2}\hat{s}_j\right) \to 1 - \alpha$$

as $n \to \infty$.

This is a range of possible values of the parameter within a given confidence level $1 - \alpha$.

Note: the larger the confidence level, the broader the confidence interval.

Likelihood Ratio Test

The maximized likelihood gives a measure of how well the model fits the data (1=perfect fit, 0=no fit). The ratio of likelihoods between two models, A "nested" in B, can be used to test whether the fit of A improves on B.

Definitions

Suppose we have two models A and B with the same structure except A has more parameters than B:

$$\boldsymbol{\theta}_A = (\theta_1, \ldots, \theta_{m+r}) \text{ and } \boldsymbol{\theta}_B = (\theta_1, \ldots, \theta_m)$$

Then A *is nested in* B.

The *likelihood ratio statistic* is $\lambda = 2\log\left(\frac{L(\hat{\boldsymbol{\theta}}_A)}{L(\hat{\boldsymbol{\theta}}_B)}\right)$.

Newton-Raphsonova metoda

- Základní princip metody:

$$p(x,\beta)=\frac{1}{1+e^{-\beta^T x}} \qquad L(\beta)=\sum_{i=1}^{n} y_i \beta^T x_i - \log(1+e^{\beta^T x_i}) \qquad \beta^{new}=\beta^{old}-\frac{\partial^2 L(\beta)}{\partial\beta\partial\beta^T}^{-1}\frac{\partial L(\beta)}{\partial\beta}$$

- Maticový zápis:

$$\beta^{new}=(X^T W X)^{-1} X^T W (X\beta^{old}+W^{-1}(y-p))$$

y ... vektor pozorování vysvětlované proměnné y_i

X ... matice plánu, typu $n\times(p+1)$

p ... vektor pravděpodobností $p(x_i,\beta^{\text{old}})$

W ... $n\times n$ diagonální matice vah, s diag. prvky $p(x_i,\beta^{\text{old}})\cdot(1-(x_i,\beta^{\text{old}}))$

- Jde o numerickou iterační metodu -> je třeba zkontrolovat, zda byla splněna podmínka konvergence (metoda „dokonvergovala“ k optimálnímu řešení)

Výhody logistické regrese

- Málo parametrů
- Snadné použití i interpretace
- Lze snadno začlenit i diskrétní prediktory
- Funguje dobře i na datech, která se poměrně značně liší od gaussovských směsí
- A především většinou dobře funguje, pokud věnujeme odpovídající pozornost přípravě dat
  - praktická zkušenost: ve čtyřech případech z pěti je logistická regrese na datech, která analyzuji, buď nejlepší nebo zhruba stejně dobrá jako jiné metody.

Interpretace, rozdíly proti OLS

- Regresní koeficienty b: kladné znamenají, že proměnná svým růstem zvyšuje šanci zařazení do skupiny kódované číslem 1, a naopak záporné indikují pokles této šance
- Často se používá $\exp(b_i)$: je to faktor, kterým se násobí šance $p/(1-p)$ při jednotkovém nárůstu x_i a neměnných ostatních x_k
  - Pozor na různá měřítka, v nichž x_i mohou být měřena;

- Místo F-testu celkové validity nyní máme chí-kvadrátový test pro totéž
- Místo t-testu signifikance proměnných v modelu jsou Waldovy statistiky; je to v podstatě totéž a čteme to stejně
- Místo R^2 jsou jen pseudo-R^2

Příklad

The following logistic regression output was produced on a data set of 40,000 credit cards.

Likelihood Ratio = 1819 (p-value < 0.001)

Variable	Coefficient	Estimate	Standard error	Wald chi-square	P > chi-square
Intercept	β_0	-0.181	0.084	4.6	0.032
Age	β_1	+0.0353	0.0013	757.6	<0.001
Income (log)	β_2	-0.0164	0.0100	2.67	0.10
Residential phone	β_3	+0.622	0.030	430.8	<0.001
Home owner *		0			
Renter	β_4	-0.155	0.039	15.6	<0.001
Lives with parents	β_5	+0.256	0.045	32.1	<0.001
Months in residence	β_6	-0.00025	0.00011	5.4	0.020
Months in current job	β_7	+0.00210	0.00025	72.9	<0.001

* Notice that the Home owner category is set as base residency category and so has no coefficient estimate. We will discuss this in a later lecture.

Zdroj: http://www2.imperial.ac.uk/~abellott/Credit%20Scoring%202.pdf

Příklad

We have used logistic regression to model the negative outcome (ie $y = 0$).

- This may seem odd given that the outcome of interest is the positive one (eg default).
- However, this model ensures the log-odds scores are the right way round: ie increasing scores imply increasing creditworthiness.
- There is no material difference. If we had modelled $y = 1$, the signs on the coefficient estimates would be reversed but everything else would be the same.

Interpretations:

- The estimates (highlighted) form the scorecard.
- Estimates greater than 0 indicate relative decrease in risk.
- Estimates less than 0 indicate relative increase in risk.
- Small p-values indicate coefficients that are statistically significantly different to zero (how small?).
- Large p-values indicate coefficients that have a good chance of actually being zero.

Příklad

Remember in the exercise in Chapter 1 we gave details of six borrowers. You were asked to select three to accept and three to reject.

Here the scores assigned by the model above are shown. The observations with the three lowest scores are rejected by the model. The actual outcome in each case is also shown. *How does your performance compare with the model?*

Age	Monthly Income (£)	Residential phone?	Residence type?	Months in residence	Months in current job	Score	Model accept or reject?	Actual outcome
22	1,145	Yes	Home owner	48	12	1.11	Reject	Good
46	15,500	Yes	Renter	48	192	2.14	Accept	Good
71	900	Yes	Renter	96	12	2.68	Accept	Good
32	5,000	Yes	Renter	48	168	1.61	Accept	Bad
25	1,385	Yes	Renter	12	0	1.05	Reject	Bad
43	3,145	No	Home owner	96	36	1.25	Reject	Bad

Příklad

Variable	Value	Coefficient	Estimate	Value × Estimate
Intercept	n/a	β_0	-0.181	-0.181
Age	22	β_1	+0.0353	+0.777
Income (log)	log(1145) =7.04	β_2	-0.0164	-0.116
Residential phone	1	β_3	+0.622	+0.622
Home owner *	1		0	0
Renter	0	β_4	-0.155	0
Lives with parents	0	β_5	+0.256	0
Months in residence	48	β_6	-0.00025	-0.012
Months in current job	12	β_7	+0.00210	0.025
Score (sum)				**+1.115**

Compute the PD of the borrower.

Score = 1.115

$$P(y = 1|s) = \frac{1}{1+e^s} \approx 0.25.$$

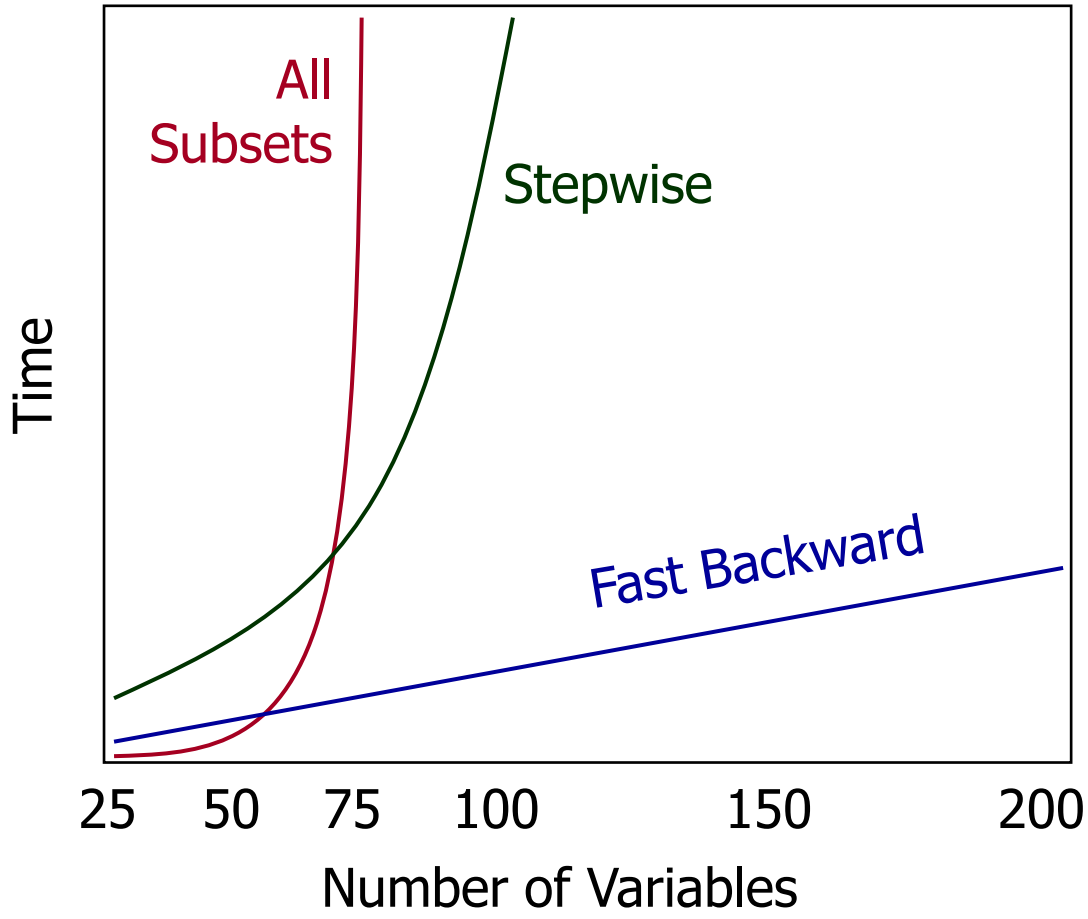
Logistic Regression with Sequential Steps

- Forward regression
  - starts with a baseline model (intercept-only)
  - searches all variables and finds the strongest one
  - keeps adding variables in order of strength until no significant improvement is achieved in the model.
- Backwards regression
  - starts with a full model using all variables
  - removes the weakest input variable provided that taking it out does not cause a significant reduction in the fit of the model
  - continues removing the weakest input variables in order unless there is a significant reduction in the fit of the model; at which point the algorithm stops.

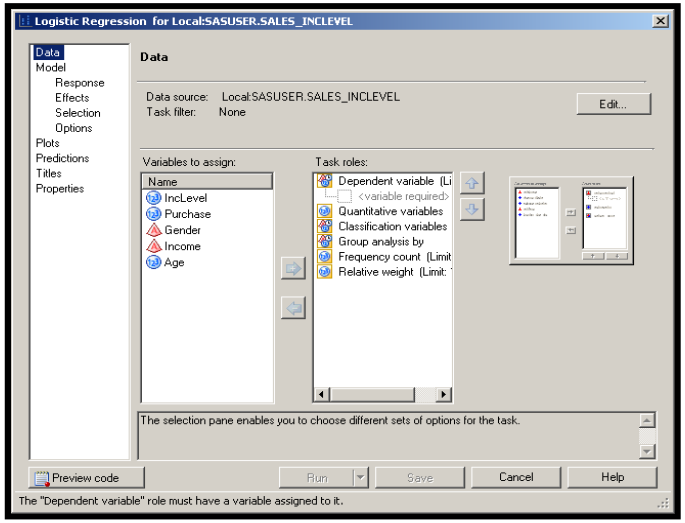
Logistic Regression with Sequential Steps

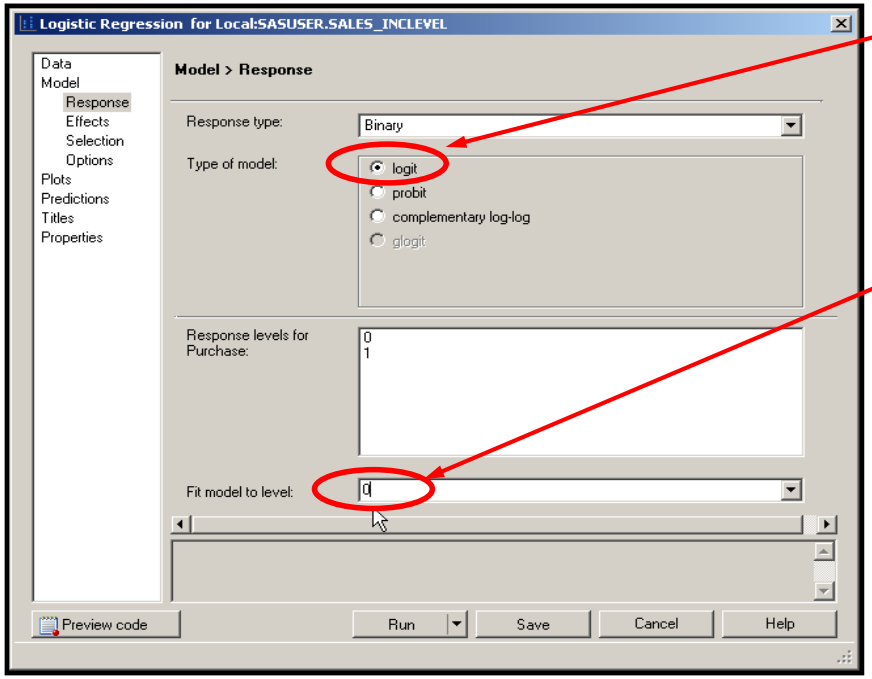
- Stepwise regression
  - is a combination of forward and backward regression
  - begins the same way as forward
  - re-evaluates the statistical significance of all included variables after each new variable is added.
- ✎ If a previously included variable becomes statistically insignificant when a new variable is added, that variable is then removed.
- ✎ The algorithm stops when no more variables can be found that add significantly to the fit of the model **and** all variables remaining in the model are statistically significant.

Scalability in PROC LOGISTIC



The Logistic Regression Task





Volba linkovací funkce.

Specify the level of the response variable that you want to model.
For example, do you want to model the probability of a 0 or a 1?

LOGISTIC Procedure

- General form of the LOGISTIC procedure:

```
PROC LOGISTIC <options>;
        CLASS variable</v-options>;
        MODEL response = <effects></options>;
        ODDSRATIO <'label'> variable </ options>;
        ROC <'label'> <specification> </ options>;
        ROCCONTRAST <'label'><contrast></ options>;
        SCORE <options>;
        UNITS predictor1=list1 </option>;
        OUTPUT <OUT=SAS-data-set> keyword=name…
                keyword=name></option>;
RUN;
```

Více např. na: http://www.okstate.edu/sas/v8/sashtml/onldoc.htm

http://www.okstate.edu/sas/v8/saspdf/stat/chap39.pdf

ODS Statistical Graphics

- PROC LOGISTIC can now create graphs as automatically as they create tables.
- The graphics have ODS styles designed for statistical work.
- The PLOTS= option is used to specify which graphs to create.
- You can specify where you want your graphs displayed by using ODS destination statements.

```
ODS GRAPHICS ON;

  statistical procedure code

ODS GRAPHICS OFF;
```

LOGISTIC Procedure - příklady

```
ods html file="logistic_vyvoj.html" style=sasweb;
proc logistic data=dm1.data_vyvoj descending;
model good4=goods_type_w phone_w a_uver_w
        fam_state_w income_w credit_w vek_w
            ;
run;
ods html close;
```

```
proc logistic data=dm1.score_base
        outest=work.model_def;

CLASS AGE_d EDUCATION_d CAR_AGE_d / param=glm;
MODEL def_bad = AGE_d EDUCATION_d CAR_AGE_d
total_income_d(init_pay_by_INCOME_d)

/ SELECTION=FORWARD HIERARCHY=MULTIPLECLASS;
score  out=work.tab_scored_def;
run;
```

LOGISTIC Procedure - příklady

```
proc logistic
data=dm1.score_base outest=work.model_def namelen=200;
where client_type="1-Novy";
CLASS sex_k child_num_k fam_state_k age_k;
MODEL def_bad = AGE_w EDUCATION_w AGE_w*EDUCATION_w
          sex_k|child_num_k|fam_state_k|age_k@4

/selection=stepwise slentry=0.6 slstay=0.1 details corrb
;
run;
```

```
proc logistic
data=dm1.score_base  inest=hc.modelSU namelen=200;
CLASS sex_k child_num_k fam_state_k age_k;
MODEL def_bad = AGE_w EDUCATION_w AGE_w*EDUCATION_w
          sex_k|child_num_k|fam_state_k|age_k@4

/selection=none maxiter=0;
output out=dm1.data_all_scr (keep=id_credit score def_bad compress=yes)
prob=score;
run;
```

What Happens to Classification Variables?

- The Logistic Regression task assumes a linear relationship between predictors and the logit for the response.
  - For categorical variables, that assumption cannot be met.
- Specification as a Classification variable creates "design variables" representing the information in the categorical variables.
  - The design variables are the ones actually used in model calculations.
  - There are many possible "parameterizations" of the design variables.

Effects (default) Coding: An Example

Design Variables

CLASS	Value	Label	1	2
IncLevel	1	Low Income	1	0
	2	Medium Income	0	1
	3	High Income	-1	-1

$$\text{logit}(p) = \beta_0 + \beta_1 * D_{\text{Low income}} + \beta_2 * D_{\text{Medium income}}$$

β_o = the average value of the logit across all categories

β_1 = the difference between the logit for Low income and the average logit

β_2 = the difference between the logit for Medium income and the average logit

Analysis of Maximum Likelihood Estimates						
Parameter		DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept		1	-0.5363	0.1015	27.9143	<.0001
IncLevel	1	1	-0.2259	0.1481	2.3247	0.1273
IncLevel	2	1	-0.2200	0.1447	2.3111	0.1285

Reference Cell Coding: An Example

			Design Variables	
CLASS	Value	Label	1	2
IncLevel	1	Low Income	1	0
	2	Medium Income	0	1
	3	High Income	0	0

$$\text{logit}(p) = \beta_0 + \beta_1 * D_{\text{Low income}} + \beta_2 * D_{\text{Medium income}}$$

β_0 = the value of the logit when income is High

β_1 = the difference between the logits for Low and High income

β_2 = the difference between the logits for Medium and High income

Analysis of Maximum Likelihood Estimates						
Parameter		DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept		1	-0.0904	0.1608	0.3159	0.5741
IncLevel	1	1	-0.6717	0.2465	7.4242	0.0064
IncLevel	2	1	-0.6659	0.2404	7.6722	0.0056

Odds Ratio Calculation from the Current Logistic Regression Model

- Logistic regression model:

$$\text{logit}(\text{p}) = \log(odds) = \beta_0 + \beta_1 * (\text{gender})$$

- Odds ratio (females to males):

$$\text{odds}_{\text{females}} = e^{\beta_0 + \beta_1}$$

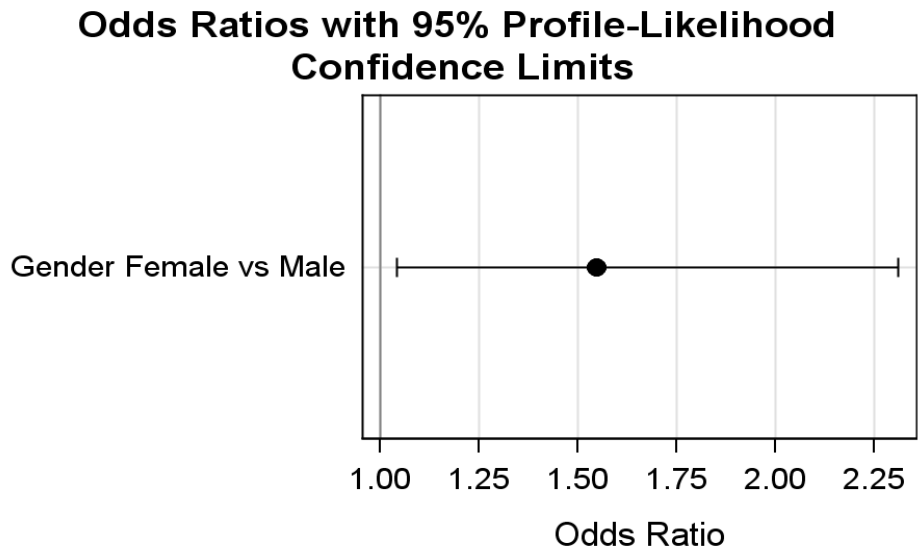
$$\text{odds}_{\text{males}} = e^{\beta_0}$$

$$\text{odds ratio} = \frac{e^{\beta_0 + \beta_1}}{e^{\beta_0}} = e^{\beta_1}$$

Odds Ratios for Categorical Predictors

Odds Ratio Estimates			
Effect	Point Estimate	95% Wald Confidence Limits	
Gender Female vs Male	1.549	1.040	2.305

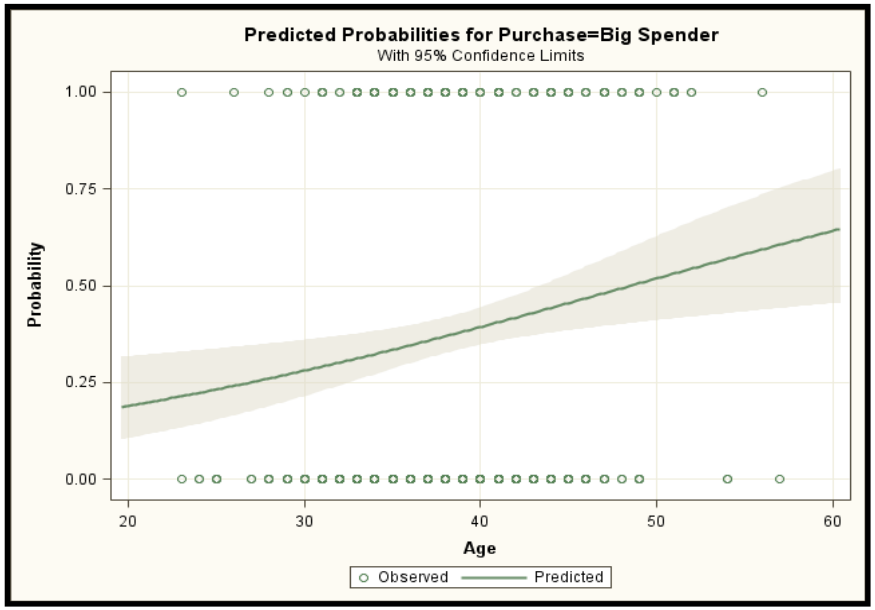
Profile Likelihood Confidence Interval for Odds Ratios				
Effect	Unit	Estimate	95% Confidence Limits	
Gender Female vs Male	1.0000	1.549	1.043	2.312



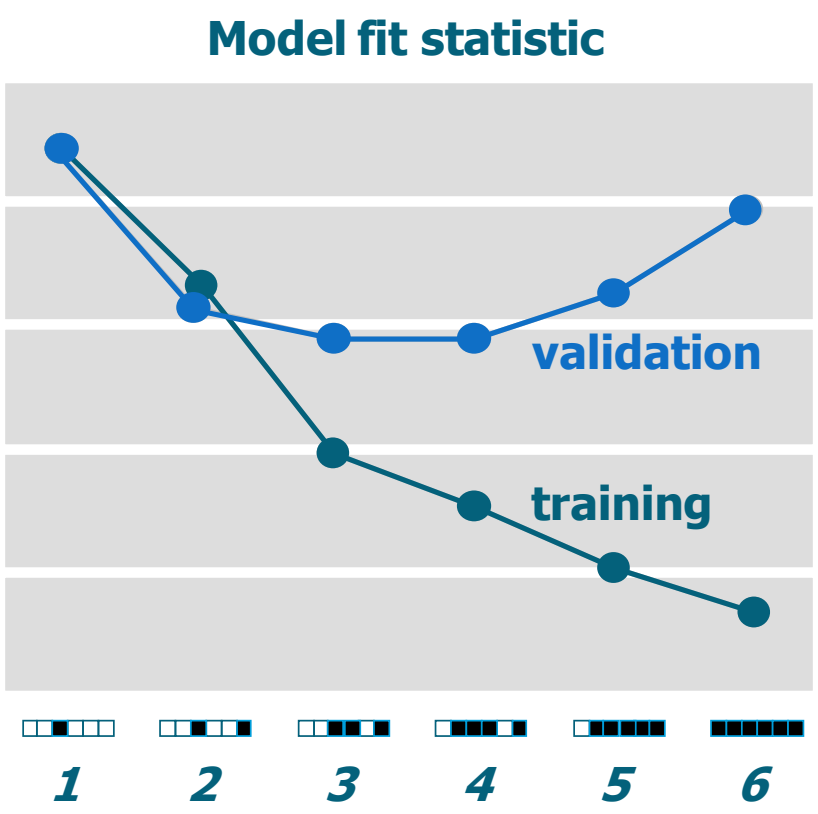
Odds Ratios for Continuous Predictors

Odds Ratio Estimates			
Effect	Point Estimate	95% Wald Confidence Limits	
Age	1.052	1.016	1.090

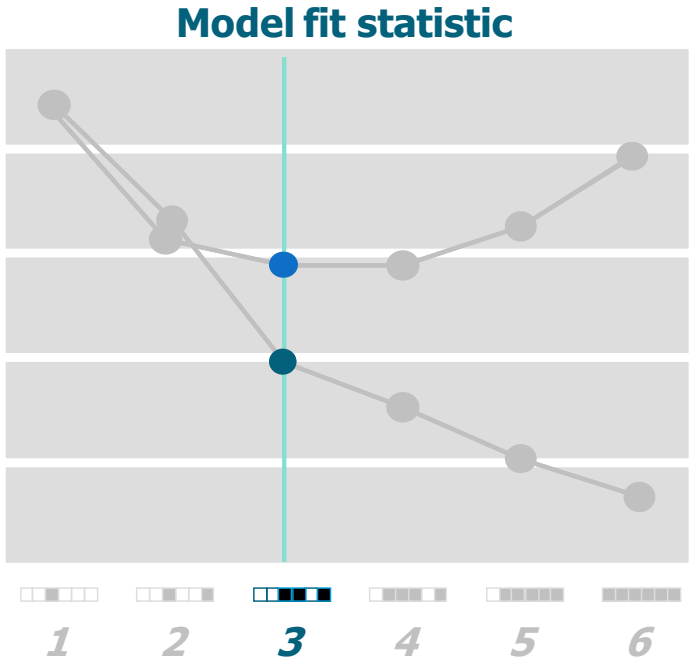
Profile Likelihood Confidence Interval for Odds Ratios				
Effect	Unit	Estimate	95% Confidence Limits	
Age	10.0000	1.663	1.176	2.373



Model Fit versus Complexity



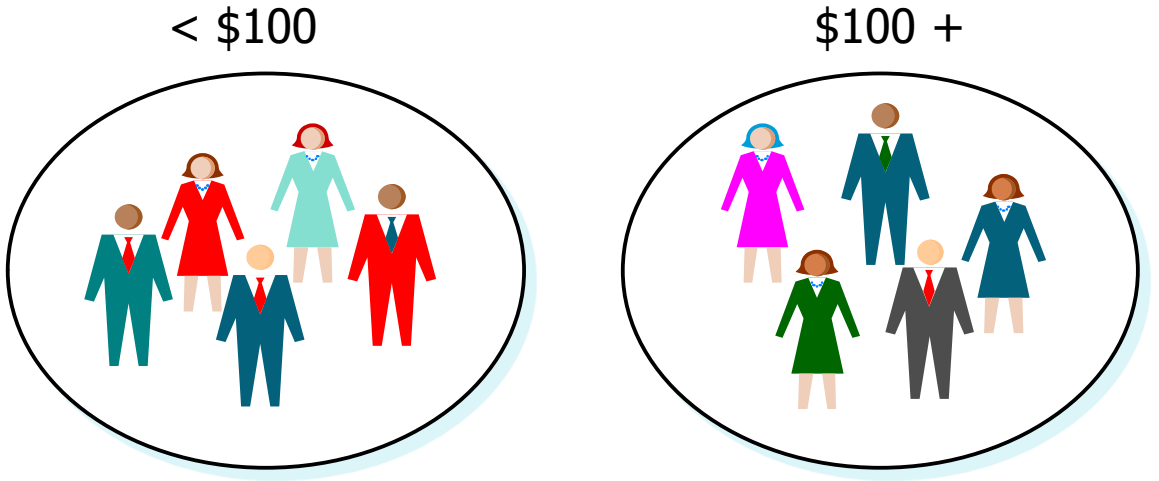
Choose simplest optimal model.



Model Assessment: Comparing Pairs

- Counting concordant, discordant, and tied pairs is a way to assess how well the model predicts its own data and therefore how well the model fits.
- In general, you want a high percentage of concordant pairs and low percentages of discordant and tied pairs.

- Následuje příklad určení těchto párů na modelu predikujícím zda daná osoba nakoupí zboží za více než 100$.



Comparing Pairs

< $100 $100 +


P(100+) = .32 P(100+) = .42

The actual sorting agrees with the model.
This is a **concordant** pair.

< $100 $100 +


P(100+) = .42 P(100+) = .42

The model cannot distinguish between the two.
This is a **tied** pair.

< $100 $100 +

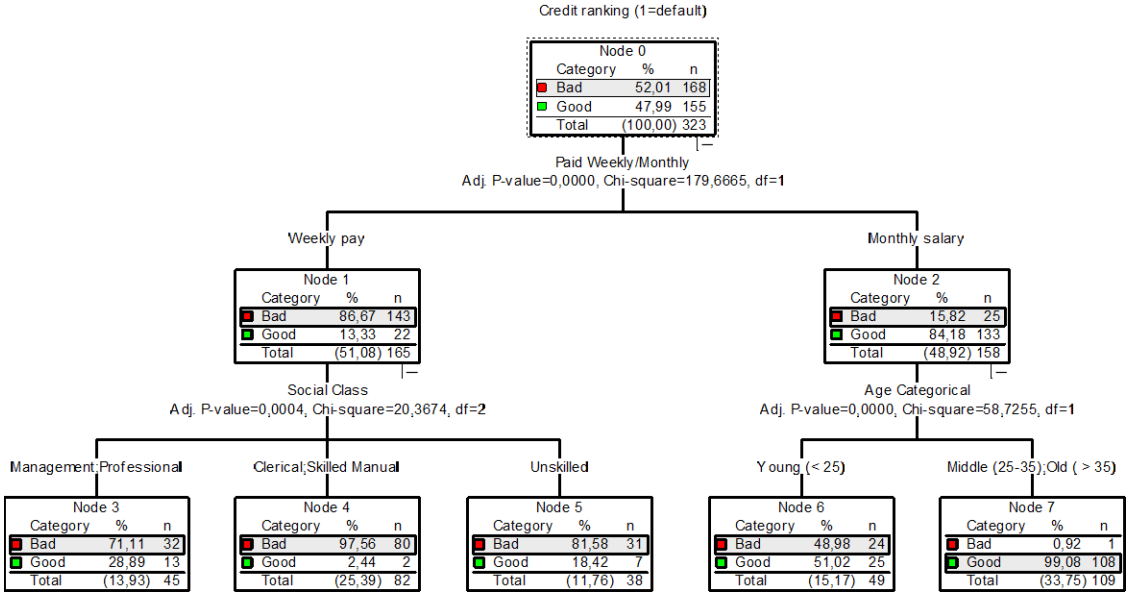

P(100+) = .42 P(100+) = .32

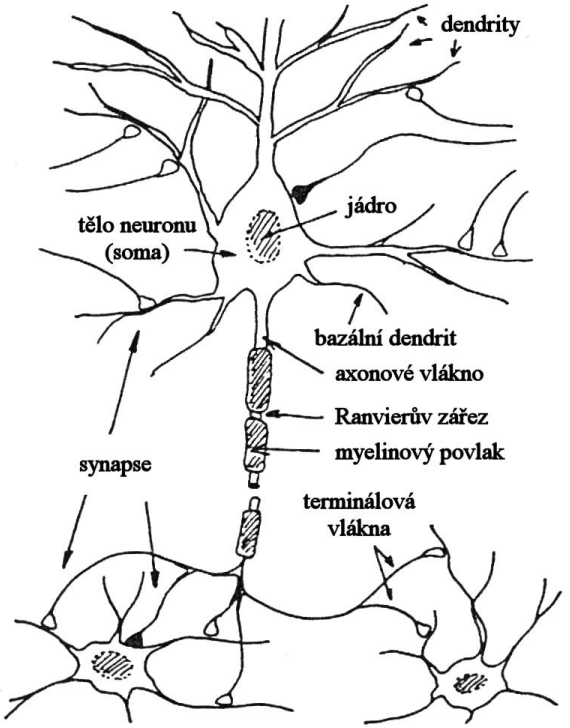
The actual sorting disagrees with the model.
This is a **discordant** pair.

- **PROC Logistic standardně nabízí četnosti (relativní) jednotlivých typů párů a z nich odvozené statistiky kvality modelu:**

Association of Predicted Probabilities and Observed Responses			
Percent Concordant	30.1	Somers' D	0.107
Percent Discordant	19.5	Gamma	0.215
Percent Tied	50.4	Tau-a	0.050
Pairs	43578	c	0.553

10. Rozhodovací stromy, neuronové sítě





Princip rozhodovacích stromů

DIVIDE ET IMPERA !

- Rozděl a panuj: vhodně rozdělím zkoumané objekty do skupin…
- a v každé skupině opět postupuji stejně (rekurze)…
- dokud nedojdu k malým skupinkám, na něž stačí zcela jednoduchý model.

Historie metody

- DIVIDE ET IMPERA je staré římské přísloví, ale…
- jeho použití v analýze dat ve smyslu rozhodovacích stromů bylo navrženo až roku 1959 W. A. Belsonem
  - W. A. Belson: britský sociolog a metodolog, zabýval se především kriminalitou mládeže
- Původní citace: William A. Belson: Matching and Prediction on the Principle of Biological Classification, Applied Stat., VIII:65-75, 1959.
  - *Již předtím (minimálně od 30. let 20. stol.) se však statistici zabývali problémy kategorizace spojitých proměnných a dělením populací, ovšem v jiném kontextu (Yule, Fisher…)*

Historie metody (pokrač.)

- První počítačově implementovaný algoritmus se jmenoval AID – vznikl roku 1963
- Citace: James N. Morgan, John A. Sonquist: Problems in the Analysis of Survey Data, and a Proposal, Journal of the American Statistical Association, 58:415-435, 1963.
- AID byl založen na analýze rozptylu (sumy čtverců) – pomůcka pro přípravu ANOVA => základ statistického směru teorie rozh. stromů (CHAID, SEARCH aj.)

Portrét: James N. Morgan

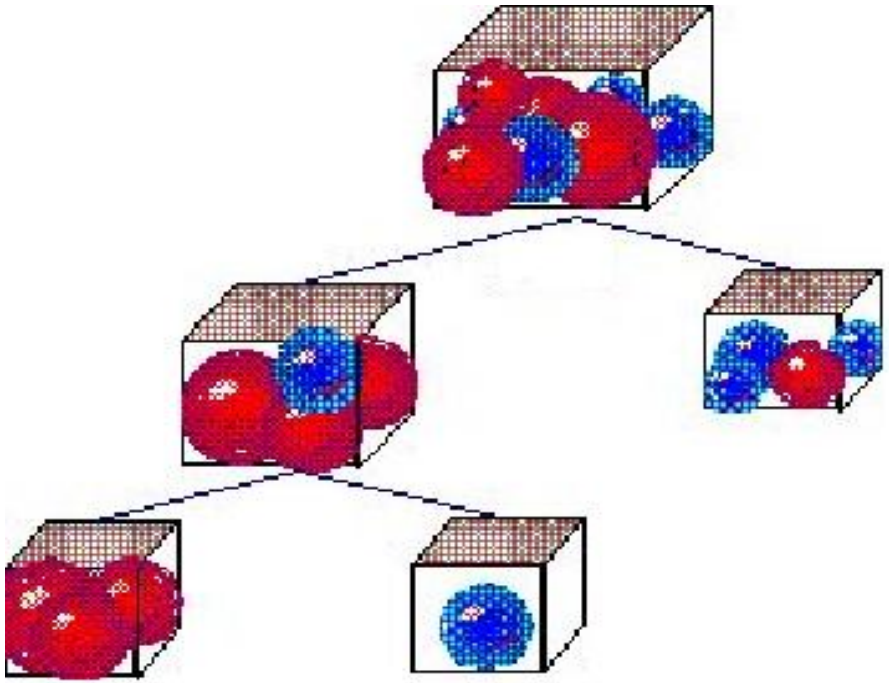
Historie metody (pokrač.)

- Označení „rozhodovací strom“ (Decision Tree) je snad z r. 1966 (Experiments in Induction – E. B. Hunt, J. Marinová a P. J. Stone) => směr zakotvený v teorii umělé inteligence.
- Zde vyšli z teorie informace – rozdělení na podskupiny má přinést „informační zisk“, snížit entropii (implementováno např. v dnes užívaných algoritmech ID3, C4.5 a C5).
- Rozvoj aplikací a uplatnění i mimo oblast teoretické vědy přinesl nástup rychlých PC a rozvoj data miningu (cca polovina 90. let 20. století)

Portrét: Earl B. Hunt

Proč se hovoří o stromech?

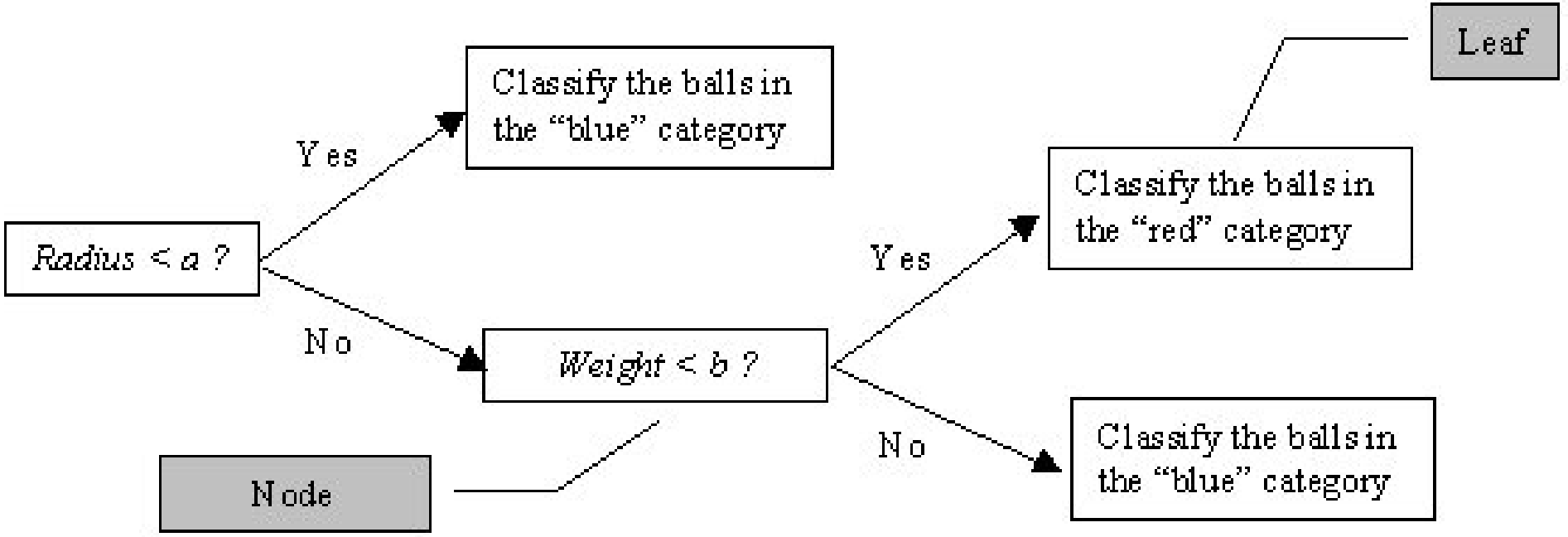
- Postupné dělení skupin zkoumaných případů lze znázornit stromovým schématem
- Kořen – větve – listy: terminologie teorie grafů



www.elseware.fr/storm

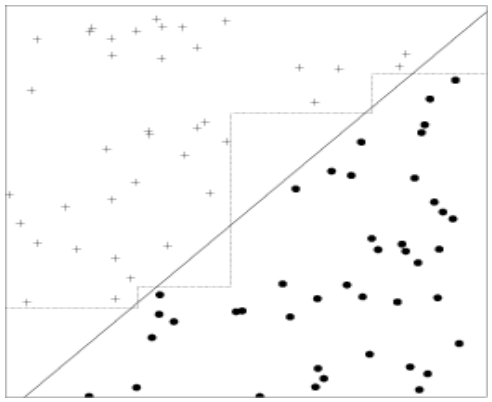
Proč „rozhodovací“?

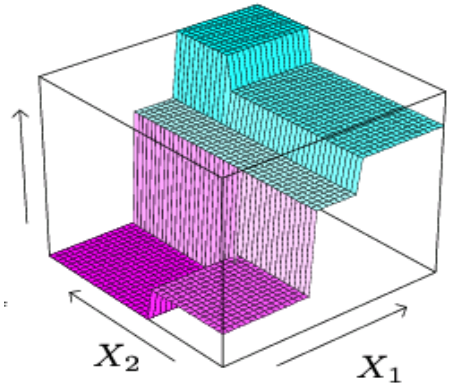
- Strom lze vyjádřit pomocí schémat *jestliže - pak*
- Lze snadno aplikovat do rozhodovacích procesů



Co je lepší: stromy, regrese…?

- Neexistuje obecné pravidlo, kdy volit jaký typ algoritmu – nejlepší bývá vyzkoušet jich několik
- Neexistují data vyloženě vhodná pro jeden typ (a vyloženě nevhodná pro jiný)





- Často však v praxi dosáhnou všechny metody podobnou přesnost => rozhodne interpretovatelnost, snadnost použití, stabilita výsledků, objem potřebných vstupních dat…

Co je lepší? (pokrač.)

- An Empirical Comparison of Decision Trees and Other Classification Methods (Tjen-Sien Lim, Wei-Yin Loh, Yu-Shan Shih, 1998) – srovnání 33 různých metod na 32 datových množinách
- Hlavní závěr: Průměrné chybovosti většiny klasifikátorů se od sebe statisticky významně neliší. Značné rozdíly jsou však ve výpočetním čase, který jednotlivé klasifikátory spotřebují.
- Nejlepší metody s přijatelným časem: polytomická logistická regrese a rozhodovací strom QUEST

  ☺ *Nutno dodat, že obě programovali autoři článku*

Binární nebo obecné stromy?

Binární stromy

- Např. CART, C5, QUEST
- Z uzlu vždy 2 větve
- Rychlejší výpočet (méně možností)
- Je třeba mít více uzlů
- Zpravidla přesnější

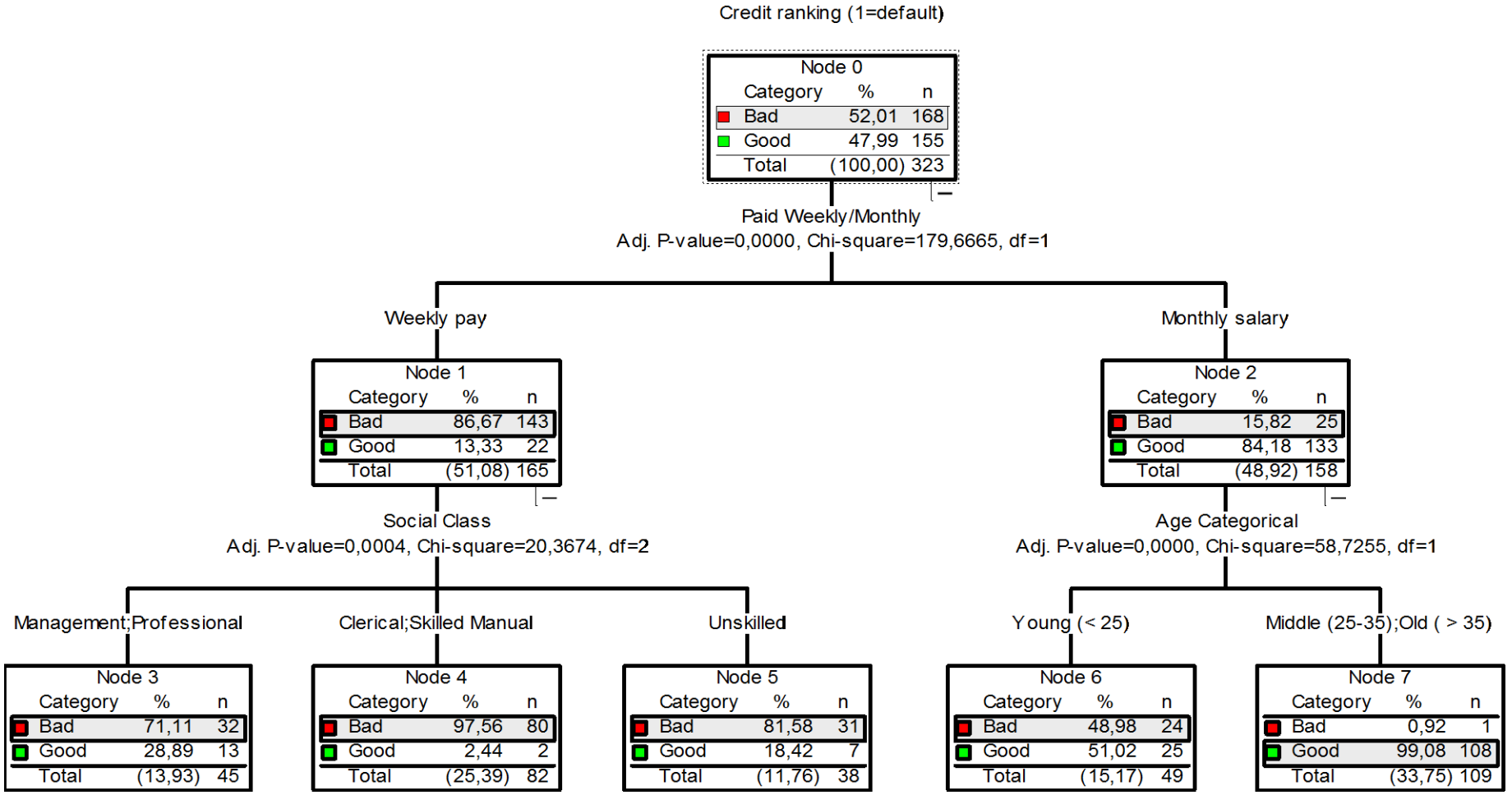
=> Data Mining, skóry

Obecné stromy

- Např. CHAID, Exhaustive CHAID
- Počet větví libovolný
- Interpretovatelnost člověkem je lepší
- Strom je menší
- Zpravidla logičtější

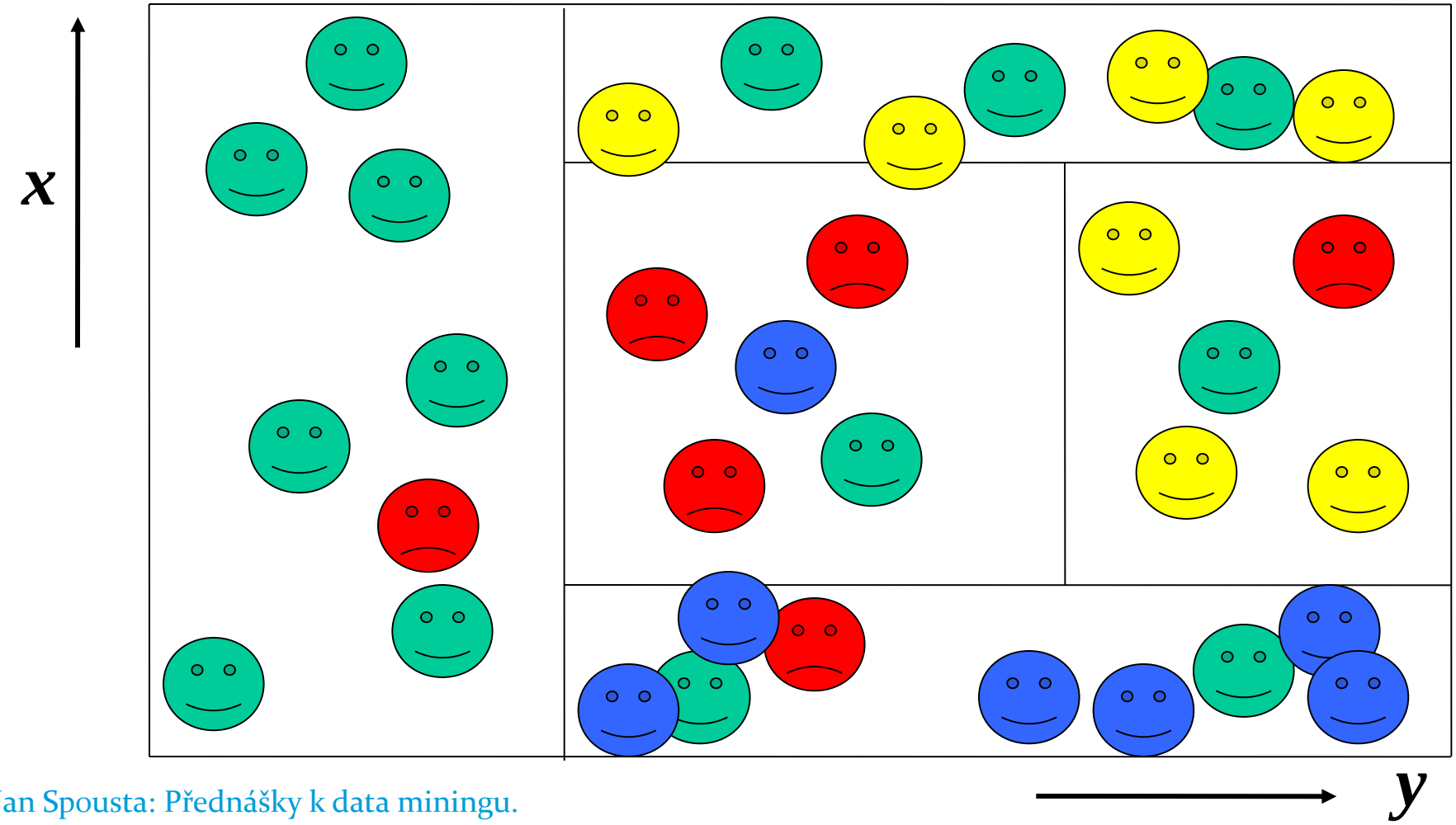
=> segmentace, mrktg.

Klasická prezentace: dendrogram



Alternativní prezentace: box chart

Vhodné pro malý počet použitých prediktorů (zde *x* a *y*)

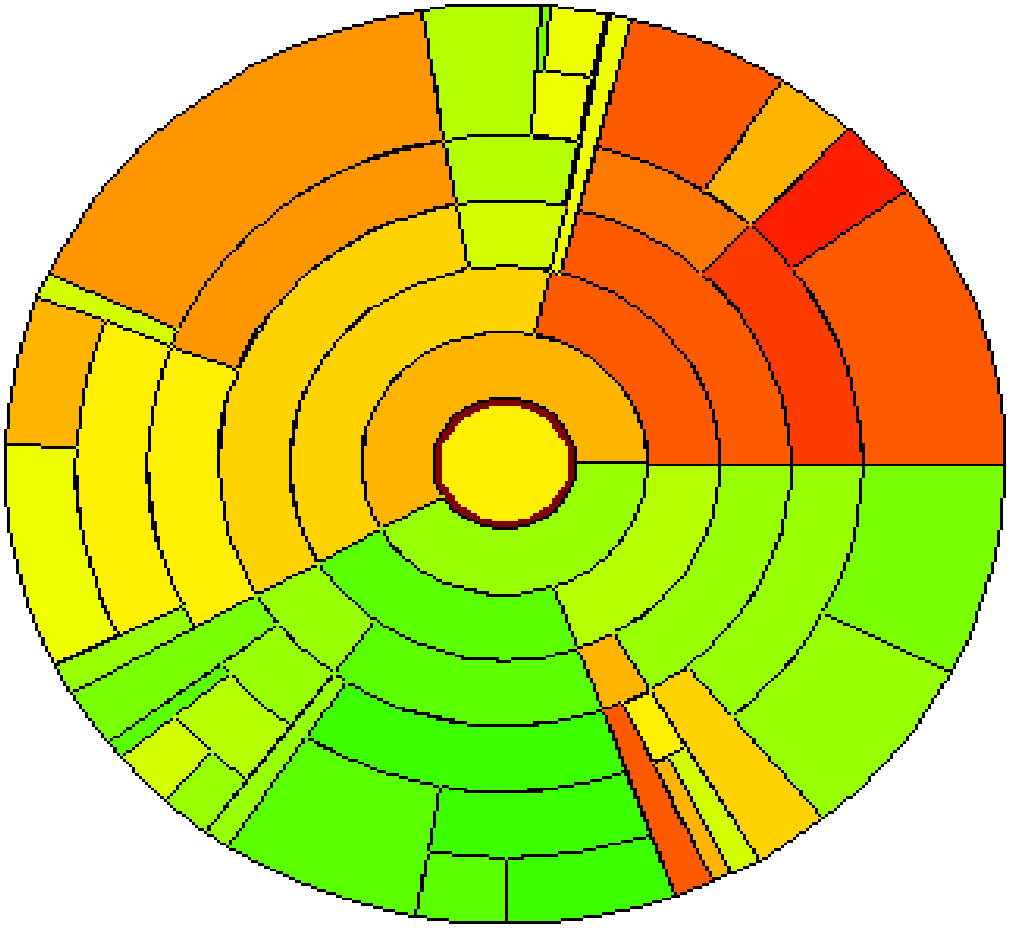


Alternativní prezentace: výseče

Snadno vidíme podíl jednotlivých větví na celém počtu případů.

Barva znázorňuje podíl hledané kategorie nebo míru homogenity uzlu.

Méně vhodné, jde-li nám o rozhodovací pravidla.



Alternativní prezentace: text

Jednoduché, ale hůře čitelné a málo výrazné

```
Group 1: All Cases N=5235, Mean(Y)=$13.02
  Group 2 EDUCATION 1989 H, NO COLLEGE DEGREE N=4186, Mean(Y)=$10.81
    Group 10 MARRIED MEN N=2535, Mean(Y)=$12.59
      Group 12 EDUCATION 12 GRADES OR LESS N=1420, Mean(Y)=$11.01*
      Group 13 EDUCATION 13+ NO COLL DEGREE N=1115, Mean(Y)=$14.45
        Group 14 AGE 18-34, N=703, Mean(Y)=$12.58*
        Group 15 AGE 35+ N=412 Mean(Y)=$16.24
          Group 18 CITY OF 25,000+NEARBY, N=287, Mean(Y)=$17.83*
          Group 19 NO CITY OF 25,000+ NEARBY N=125, Mean(Y)=$12.82*
    Group 11 SINGLE MAN OR WOMAN N=1651, Mean(Y)=$8.55*
  Group 3 COLLEGE GRAD OR MORE N=1049, Mean(Y)=$19.48
    Group 4 AGE 18-29 N=374, Mean(Y)=$14.17*
    Group 5 AGE 30+ N=675, Mean(Y)=$22.00
      Group 6 MARRIED MEN N=530, Mean(Y)=$24.34
        Group 8 AGE 30-39 N=329, Mean(Y)=$20.77
          Group 16 LIVING IN SAME STATE GREW UP N=174, Mean(Y)=$16.86*
          Group 17 LIVING IN DIFFERENT STATE N=155, Mean(Y)=$25.30*
        Group 9 AGE 40+ N=201, Mean(Y)=$28.04*
      Group 7 SINGLE MEN, WOMEN N=145, Mean(Y)=$16.3825*
```

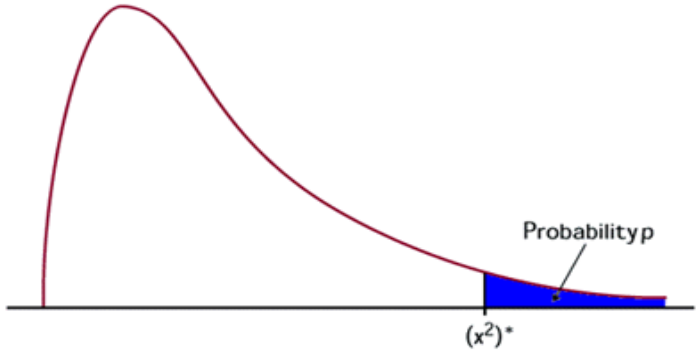
Algoritmus CHAID – úvod

- **CH**i-squared **A**utomatic **I**nteraction **D**etector
- Jeden z nejrozšířenějších rozhodovacích stromů v komerční oblasti (vedle QUEST a C4.5 / C5)
- Kass, Gordon V. (1980). An exploratory technique for investigating large quantities of categorical data. Applied Statistics, Vol. 29, pp. 119-127.
  - Založeno na autorově disertaci na University of Witwatersrand (Jihoafrická rep.)
  - Předchůdci: AID – Morgan a Sonquist, 1963; THAID – Morgan a Messenger, 1973

Připomenutí: Test nezávislosti χ^2

- Nezávislost testujeme na základě výrazu

$$\chi^2 = \sum_i \sum_j \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$



- Jsou-li *x* a *y* nezávislé, má tento výraz Pearsonovo chí-kvadrát rozdělení s df = $(r - 1)(s - 1)$
- Test: plocha pod grafem „nad“ pozorovanou hodnotou (~signifikance p) < α => zamítnu hypotézu nezávislosti *x* a y
- Současné testování více hypotéz
  => nutno adjustovat α (Bonferroni)

Algoritmus CHAID: idea

- Začíná se u celého souboru
- Postupné větvení / štěpení souboru (přípustné je rozdělení na libovolný počet větví vycházejících z jednoho uzlu)
- Algoritmus je rekurzivní – každý uzel se dělí podle stejného předpisu
- Zastaví se, pokud neexistuje statisticky signifikantní rozdělení => vzniká list
  - Obvykle je navíc podmínka minimálního počtu případů v uzlu a/nebo v listu, příp. maximální hloubky stromu
  - http://support.spss.com/ProductsExt/SPSS/Documentation/Statistics/algorithms/14.0/TREE-CHAID.pdf

CHAID: postup v uzlu

- Pro všechny prediktory
  - Vytvoř kontingenční tabulku target x prediktor (rozměr k x l)
  - Pro všechny dvojice hodnot prediktoru spočti chí-kvadrátový test podtabulky (k x 2)
  - „Podobné“ (=ne signifikantně odlišné) dvojice postupně spojuj (počínaje nejnižšími hodnotami chí-kvardrátu) a přepočítávej výchozí kontingenční tabulku. Zastav se, když signifikance všech zbylých podtabulek je vyšší než stanovená hodnota.
  - Zapamatuj si spojené kategorie a signifikanci chí-kvadrátu výsledné tabulky s redukovanou dimenzionalitou
- Vyber prediktor, kde je tato signifikance nejnižší
- Pokud jsou splněny podmínky štěpení, rozděl případy v uzlu podle již „spojených“ kategorií

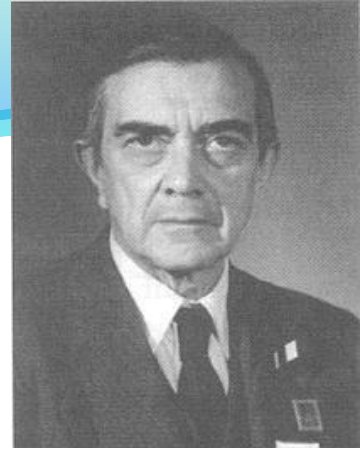
CHAID: zhodnocení

- Pokud je počet kategorií prediktoru n, tak je třeba provádět jen řádově n^2 testů
- Kdyby se testovala všechna možná rozdělení, rostl by počet testů exponenciálně s růstem n
- CHAID tím šetří výpočetní čas, ale zároveň není zaručeno, že najde optimální řešení uzlu (greedy search v uzlu)
- Ordinální znak: lze spojit jen sousední kategorie
- Spojitý znak: nutná je kategorizace
  - Zde existují lepší i horší implementace

Další algoritmy

- Existují desítky příbuzných algoritmů, často navzájem dost podobných
- Zde pouze naznačíme vlastnosti několika z nich (často používaných a/nebo zajímavých)
  - CART
  - ID3 a C5
  - QUEST
  - TreeNet

CART / C&RT



- **C**lassification **A**nd **R**egression **T**ree
- Algoritmus je založen na počítání míry diverzity („nečistoty“) uzlu: chci maximalizovat

$$div(matka) - (\mathbf{div(dcera\ A) + div(dcera\ B)})$$

konst.

s tím, že sčítance vážíme podílem případů v uzlech

- Giniho míra diverzity (inspirace z ekonomie, kde se podobně měří nerovnosti v distribuci majetku a příjmů)

$$div_{\text{Gini}} = 1 - \Sigma\, p_i^2$$

- p_i jsou u nás relativní četnosti v uzlech

Portrét: Corrado Gini

ID3, C4.5, C5 (See5)

- Místo Giniho míry užívají entropii

$$div_{\text{entrop}} = -\sum p_i \ln_2 p_i$$

= *střední počet bitů potřebných pro zakódování případu v daném uzlu*

- Binární stromy
- Zabudovaný algoritmus pro zjednodušení množiny odvozených pravidel – lepší interpretovatelnost
- Ross Quinlan: Induction of decision trees (1986); týž: C4.5: Programs for Machine Learning, (1993); týž: C5.0 Decision Tree Software (1999)
- http://www.rulequest.com/see5-info.html

Portrét: Ross Quinlan

QUEST

- **Q**uick, **U**nbiased and **E**fficient **S**tatistical **T**ree
- Loh, W.-Y. and Shih, Y.-S. (1997), Split selection methods for classification trees, Statistica Sinica, vol. 7, pp. 815-840
- Výběr štěpící proměnné na základě statistického testu nezávislosti prediktor x target => mírně suboptimální, ale rychlé, navíc výběr štěpící proměnné je nevychýlený
- Jen nominální target (=závisle proměnná)
- Binární strom, pruning
- Používá se imputace chybějících hodnot

Portrét: Wei-Yin Loh

TreeNet

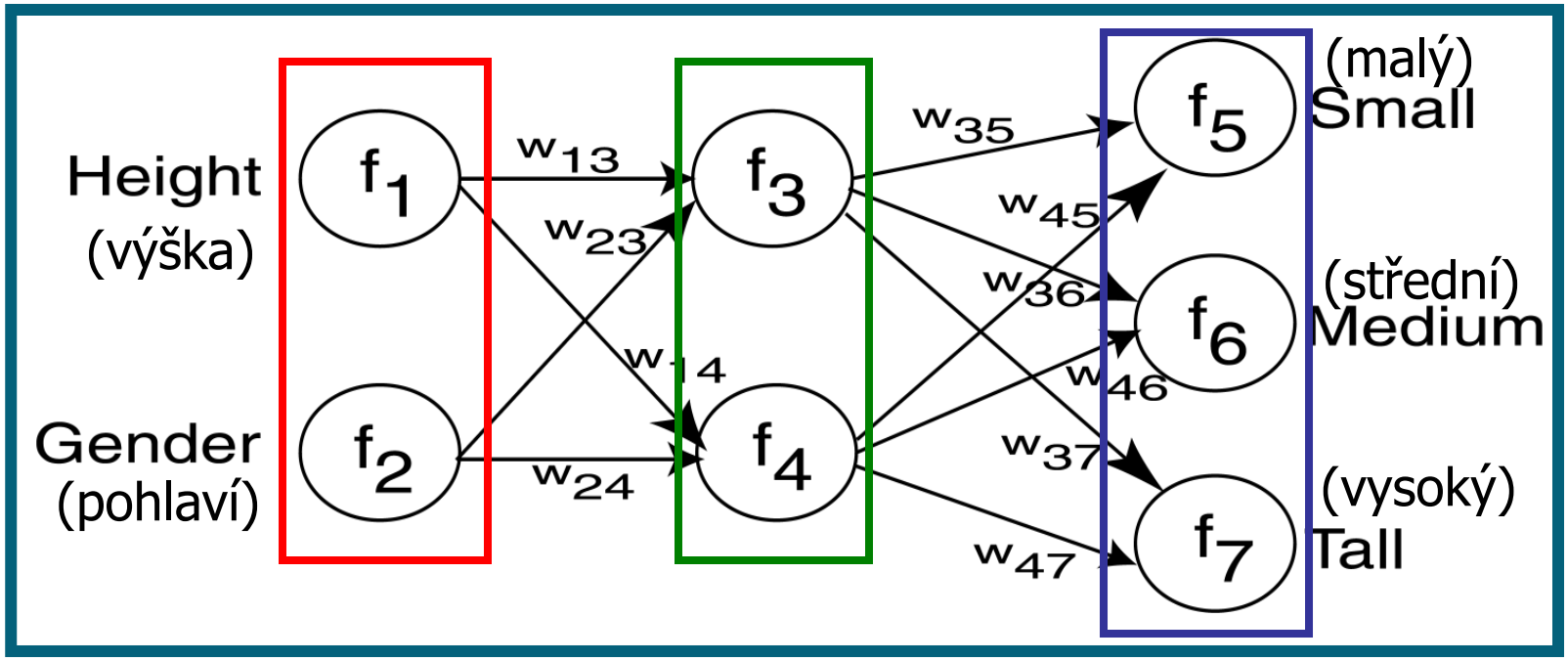
- Friedman, J. H. (1999): Greedy Function Approximation: A Gradient Boosting Machine, Technical report, Dept. of Statistics, Stanford Univ.
- Namísto jednoho velkého stromu „les“ malých
- Výsledná predikce vzniká váženým součtem predikcí jednotlivých složek
- Analogie Taylorova rozvoje: rozvoj do stromů
- Špatně interpretovatelné (černá skříňka), ale robustní a přesné; nižší nároky na kvalitu a přípravu dat než neuronová síť nebo boosting běžných stromů
- Komerční, www.salford-systems.com

Portrét: Jerome H. Friedman

Neuronové sítě (Neural Networks)

- Někdy se také uvádí název ***Artificial Neural Networks (ANN)***, *tj. umělé neuronové sítě.*
- Založené na pozorované funkcionalitě lidského mozku.
- Ovšem v porovnání s mozkem jde o velmi zjednodušený matematický model.
- Často jde u NN o adaptivní systém, který mění svou strukturu na základě vnějších či vnitřních informací získaných v průběhu učící fáze.
- Využívají se např. při vyhledávání vzorů v datech, rozpoznávání řeči nebo klasifikačních problémech.
- http://en.wikipedia.org/wiki/Artificial_neural_network

Příklad neuronové sítě

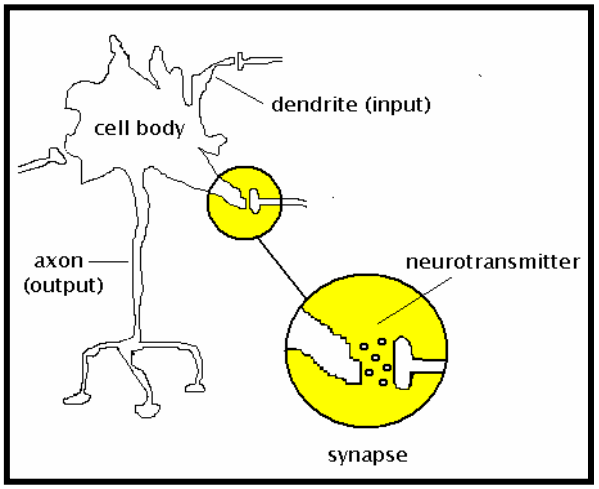


vstup (input)

skrytá vrstva (hidden layer)

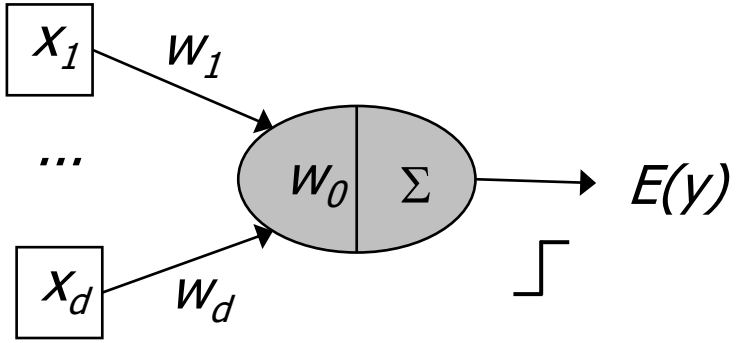
výstup (output)

The Neuron



- Excitatory (+) and inhibitory (-) inputs, arriving at the dendrites, are weighted by adaptable synapses.
- The weighted inputs are added together.
- If the sum is greater than an adaptable threshold (bias) value, the neuron sends activation down its axon.

The McCulloch-Pitts Neuron



- A McCulloch-Pitts neuron with d inputs is formally defined by the following equation:

$$E(y) = f\left(w_0 + \sum_{i=1}^{d} w_i x_i \right)$$

- The step function, $f(.)$, turns each McCulloch-Pitts neuron into a linear classifier/discriminator.

The Hebb Rule

- The strength of the connection between neurons i and j should be adjusted in accordance with the equation:

$$\Delta w_{ij} = \eta \, \hat{y}_i x_j$$

- *The eta (η) term* is the neuron's *learning rate*, which scales the amount of weight adjustment.
- Permitted learning rate values range from 0 to 1.
- Large learning rate values risk divergence.

The Widrow-Hoff Delta Rule

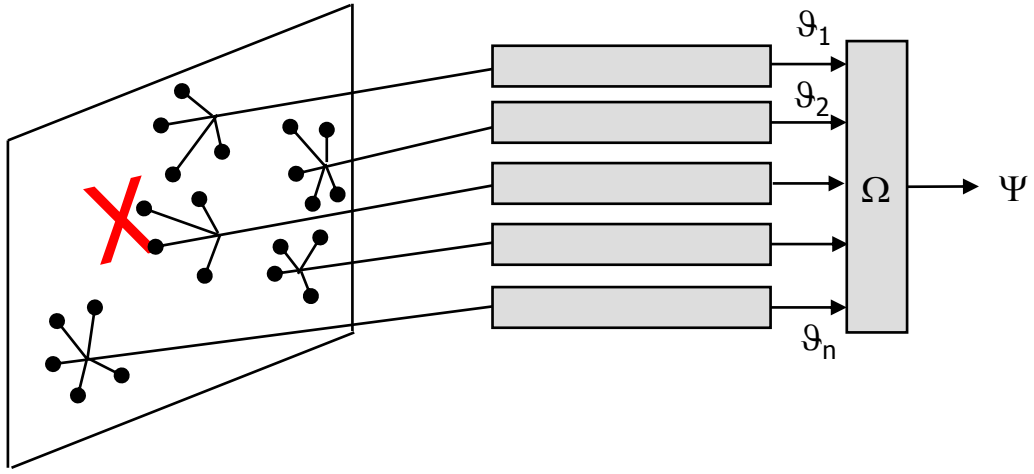
- Hebb's learning rule is unstable.
- Widrow and Hoff proposed a variant of Hebb's rule, one that is stable under a range of learning rates:

$$\Delta w_{ij} = \eta(y_i - \hat{y}_i)x_j$$

- They called their learning model the *delta rule*.
- Because the delta rule reduces the sum of squared error, it is also known as the *least mean squares rule*.

The Perceptron

- The perceptron is a pattern-recognition machine invented in the 1950s for optical character recognition.

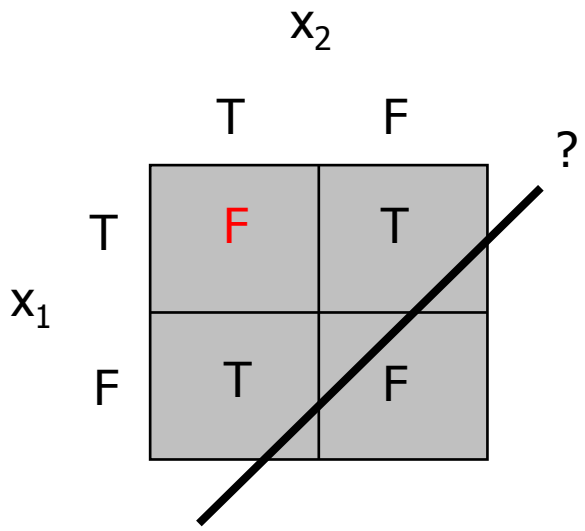


- Each processing unit is a *McCulloch-Pitts neuron*.
- A perceptron with *n* outputs is a discriminator function that divides the input space into *n* distinct regions.

The Limitations of a Simple Perceptron

- The simple (linear) perceptron can only solve linearly separable problems.
- The EXLUSIVE OR truth table (below) is an example of a problem that is **not** linearly separable.

Inputs		Output
x_1	x_2	
F	F	F
T	F	T
F	T	T
T	T	F



(Ne)Výhody NN

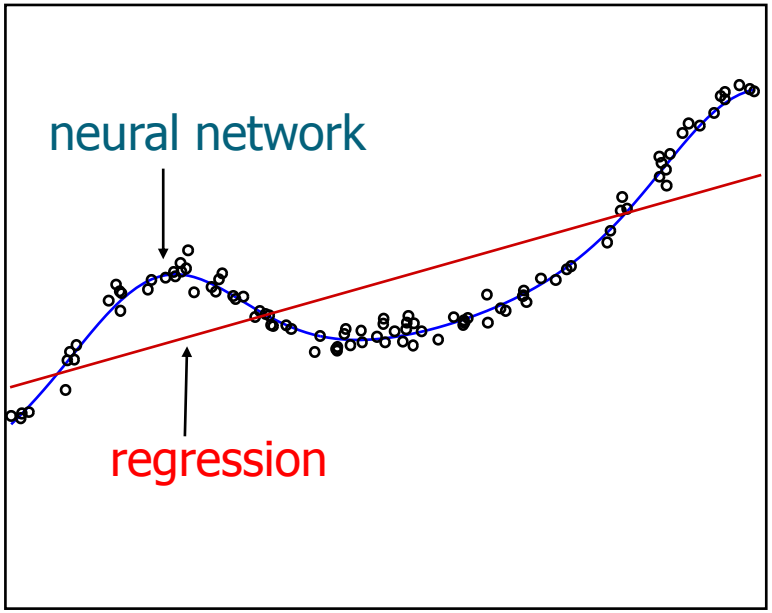
Výhody:

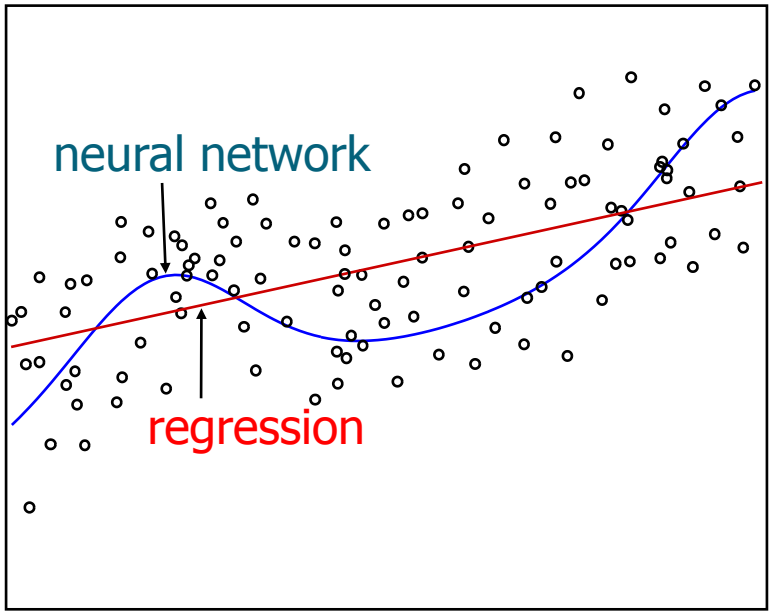
- Schopnost učení.
- Snadná parametrizace.
- Robustnost.
- Řeší mnoho problémů.

Nevýhody:

- Nesnadné porozumění/interpetace.
- Můžou trpět přeučením (overfitting).
- Vstupy musí být numerické.
- Obtížná verifikace.

The Impact of Noisy Data





Typy neuronových sítí

- Existuje celá řada typů neuronových sítí, přičemž každý z nich se hodí na jinou třídu úlohy.

- Podle přítomnosti „učitele“ dělíme neuronové sítě na
  - sítě s učitelem (srovnávání výstupu s požadovaným)
  - sítě bez učitele (bez vnějšího arbitru).

Typy neuronových sítí podle zpracování signálu

- V tabulce je základních devět typů sítí:
  - optimální lineární asociativní paměť (Optimum Linear Associative Memory – OLAM),
  - Hebbova síť (HEBB),
  - Hammingova síť (HAMM),
  - vícevrstvá síť s bipolárními neurony (Multi Layer Perceptron 1 – MLP1),
  - vícevrstvá síť se spojitým chováním (MLP2), Kohonenovy mapy (SOM),
  - síť s radiální bází (RBF),
  - modulární síť (MOD) a
  - síť se zpětným šířením (counterpropagation – COUNT).
- Další sítě lze vytvářet jejími kombinacemi.

Typ sítě	Vrstvy			Autoři
	Vstupní	Skrytá	Výstupní	
OLAM	*	–	L(+Z)	Haykin
HEBB	*	–	L+Z	Hopfield
HAMM	*	L+MAX	L+Z	Lipmann
MLP1	*	L+Z	L+Z	Widrow, Hoff
MLP2	*	L+S	L+S	Rummelhart
SOM	*	–	V+MIN	Kohonen
RBF	*	V+G	L	Poggio, Girosi
MOD	*	L+E	L	Jacobs, Jordan
COUNT	*	V+MIN	L	Nielsen

Symbol	Způsob zpracování signálu
–	chybí vrstva
*	žádné
L	lineární kombinace
V	vzdálenost
Z	znaménko
S	sigmoida
G	Gaussova funkce
E	exponenciála
MIN	nejmenší vyhrává
MAX	největší vyhrává

Asociativní neuronové sítě

- U asociativní paměti probíhá vybavení příslušné informace na základě její částečné znalosti (asociace).

- Rozlišujeme sítě s pamětí
  - autoasociativní (upřesnění či zúplnění vstupní informace na základě již naučeného)
  - heteroasociativní (vybavení si sdružené informace na základě vstupní asociace)

Učení neuronových sítí

- Algoritmus učení je různý, nicméně obecně má tyto kroky:
  - inicializace vah (malé náhodné hodnoty)
  - předložení nového vzoru (vektor reálných hodnot X)
  - výpočet aktuálního vstupu (podle f aktivační funkce)
  - přizpůsobení vah (přepočtení vah podle zjištěné odchylky)
  - opakování procesu učení (až do stabilizace vah w_i)

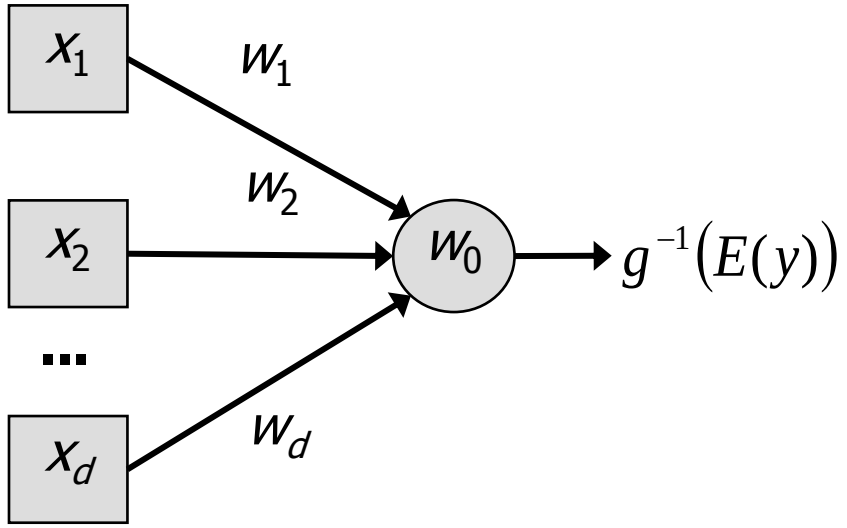
- Fáze učení sítě se nazývá adaptivní a po naučení je síť ve fázi vybavování (aktivní fázi).

Využití neuronových sítí

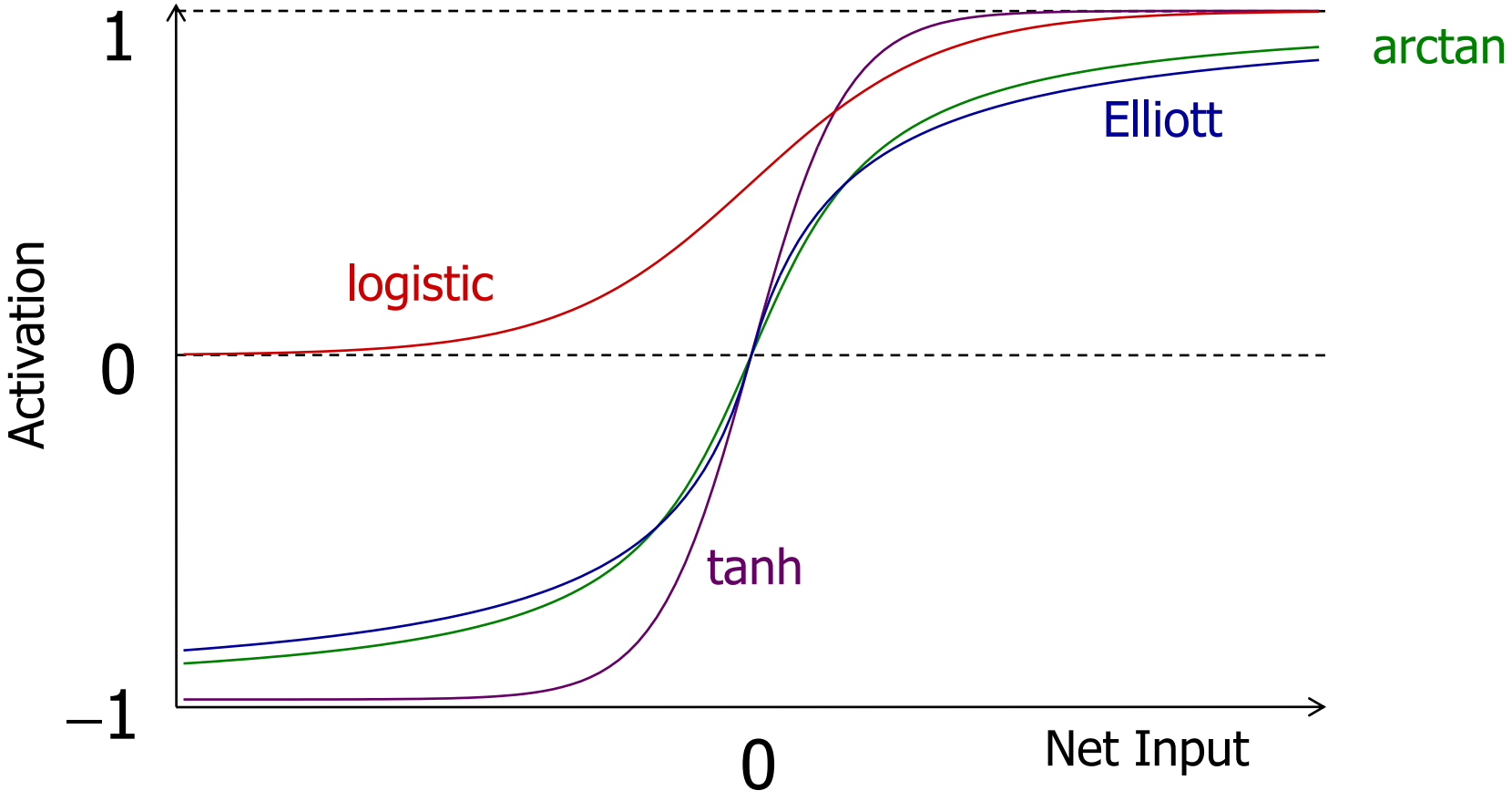
Úloha	Vhodné neuronové sítě
logické obvody	HEBB, HAMM, MLP1
odstranění šumu	MLP1, MLP2, RBF, MOD
řeč a výslovnost	MLP2, SOM
komprese	COUNT
data mining	OLAM, HEBB, SOM
optické rozpoznávání znaků	HEBB, OLAM, HAMM, MLP1, MLP2, RBF, SOM

Linear Perceptron

$$g^{-1}(E(y)) = w_0 + \sum_{i=1}^{d} w_i x_i$$

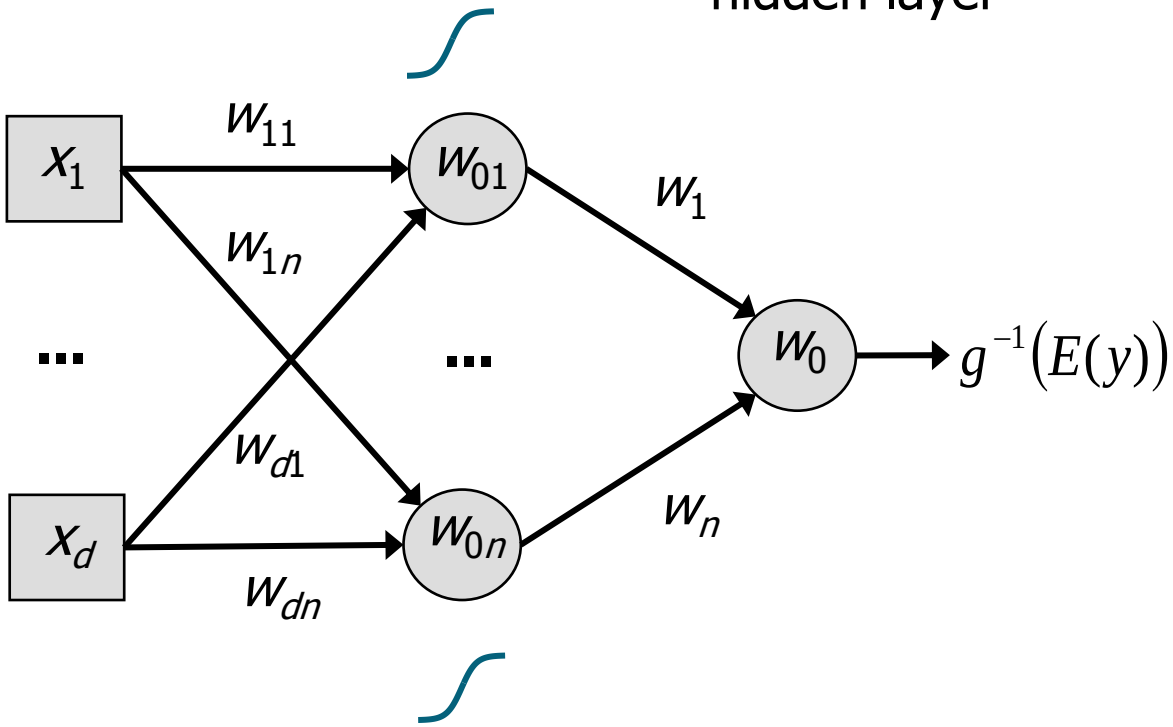


Activation Functions



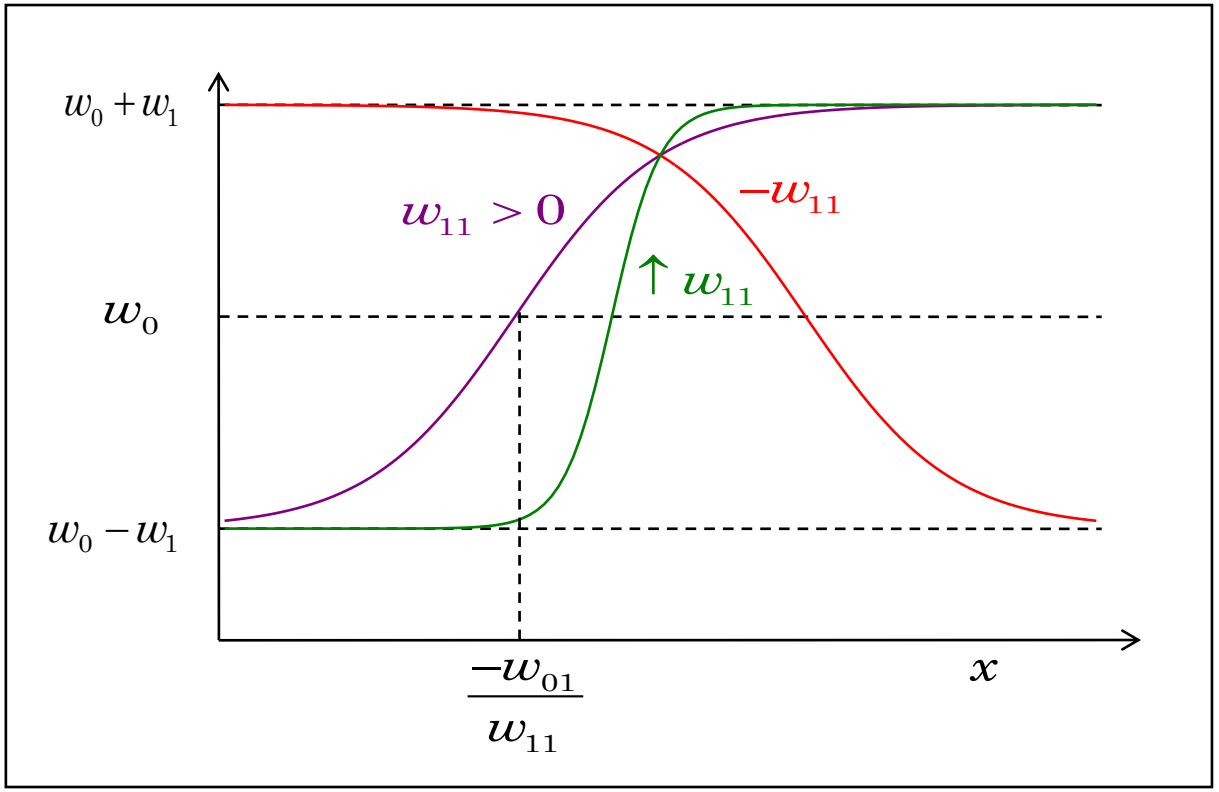
Multilayer Perceptron

$$g^{-1}(E(y)) = w_0 + \sum_{i=1}^{h} w_i g_i \underbrace{\left(w_{0i} + \sum_{j=1}^{d} w_{ij} x_j \right)}_{\text{hidden layer}}$$

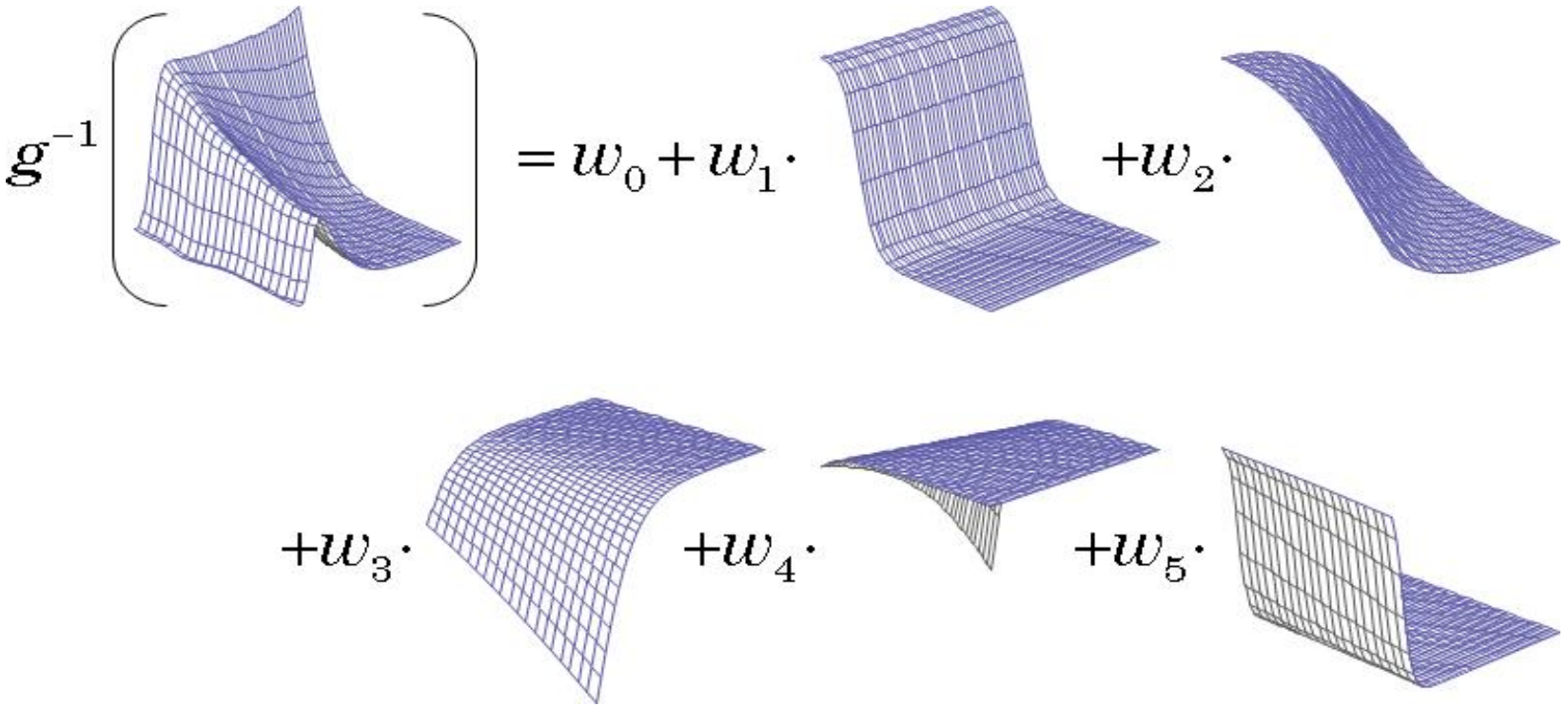


Shaping the Sigmoid

$$w_0 + w_1 \tanh(w_{01} + w_{11}x)$$

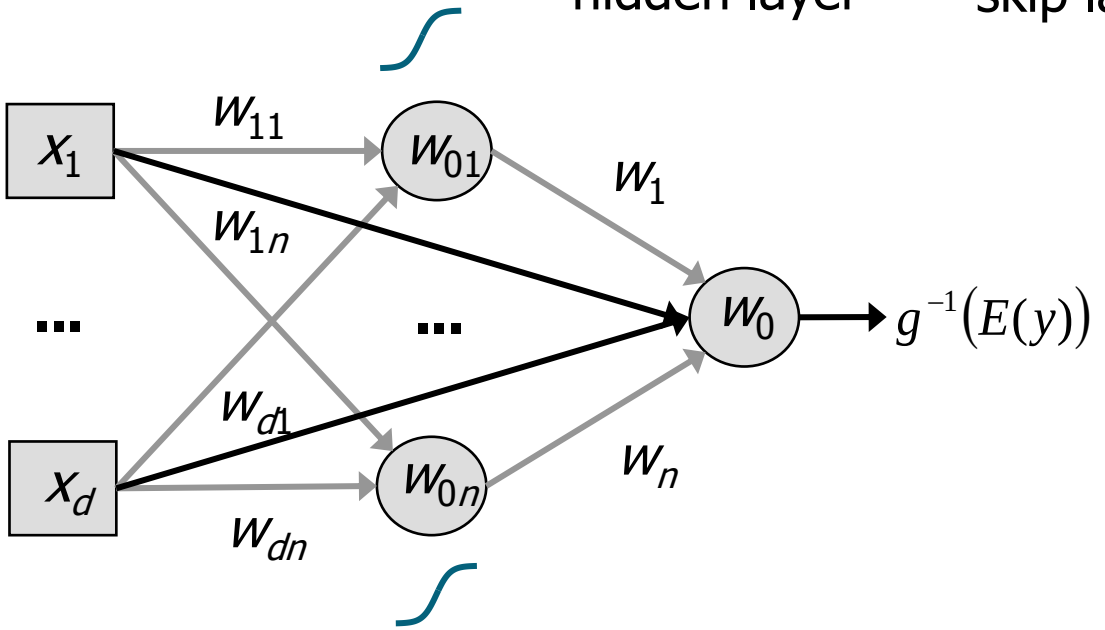


Sigmoidal Basis Functions



Skip-Layer Perceptron

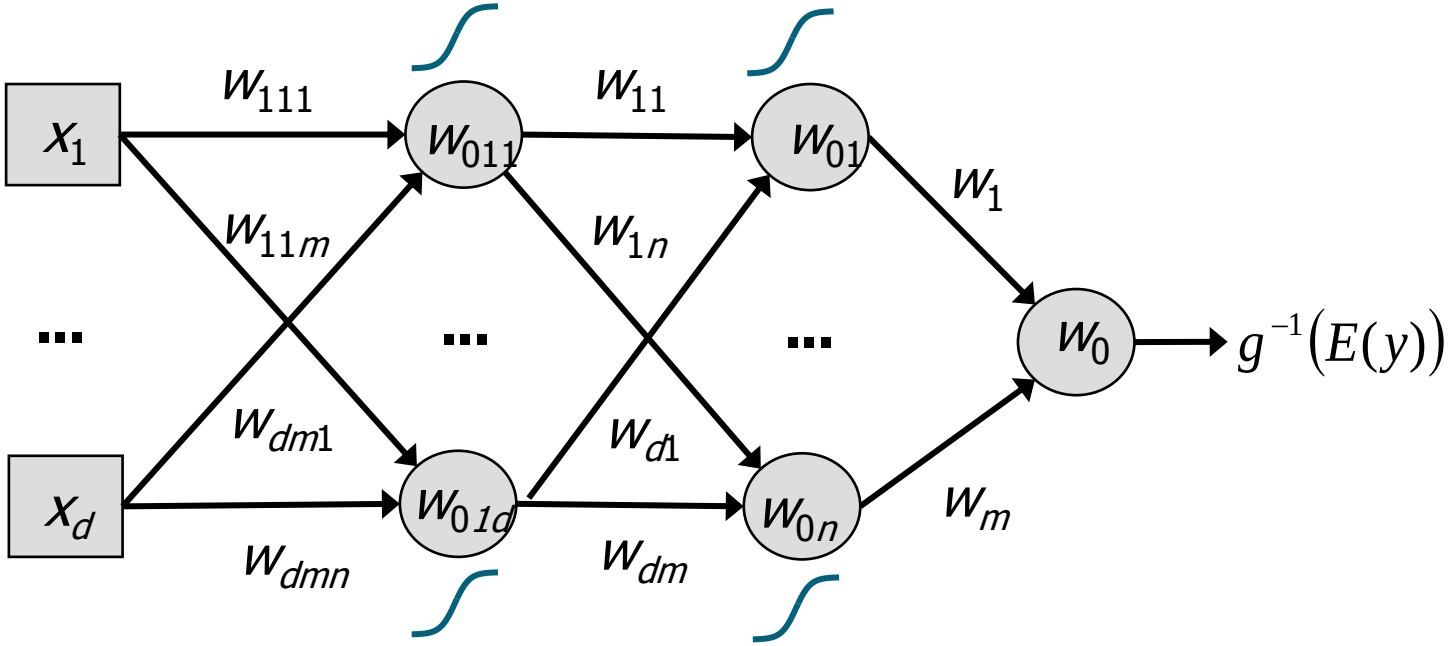
$$g^{-1}(E(y)) = w_0 + \underbrace{\sum_{i=1}^{h} w_i g_i\left(w_{0i} + \sum_{j=1}^{d} w_{ij} x_j\right)}_{\text{hidden layer}} + \underbrace{\sum_{k=1}^{d} w_k x_k}_{\text{skip layer}}$$



MLP with Two Hidden Layers

$$g^{-1}(E(y)) = w_0 + \sum_{k=1}^{m} w_k g_k \left(w_{0k} + \sum_{j=1}^{n} w_{jk} \; g_j \left(w_{0jk} + \sum_{i=1}^{d} w_{ijk} x_i \right) \right)$$

nested hidden layers

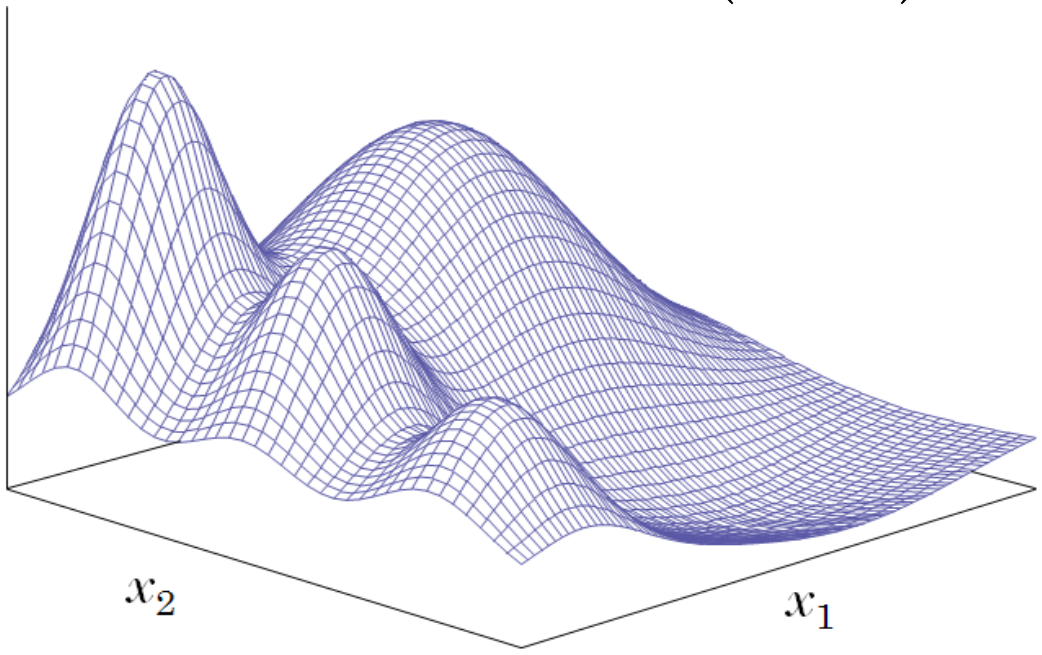


How Many?

- A single hidden layer network models any **continuous** relationship between the inputs and outputs.
- Two hidden layers model **discontinuous** relationships.
- The number of hidden units that will be required in each defined hidden layer is problem specific.

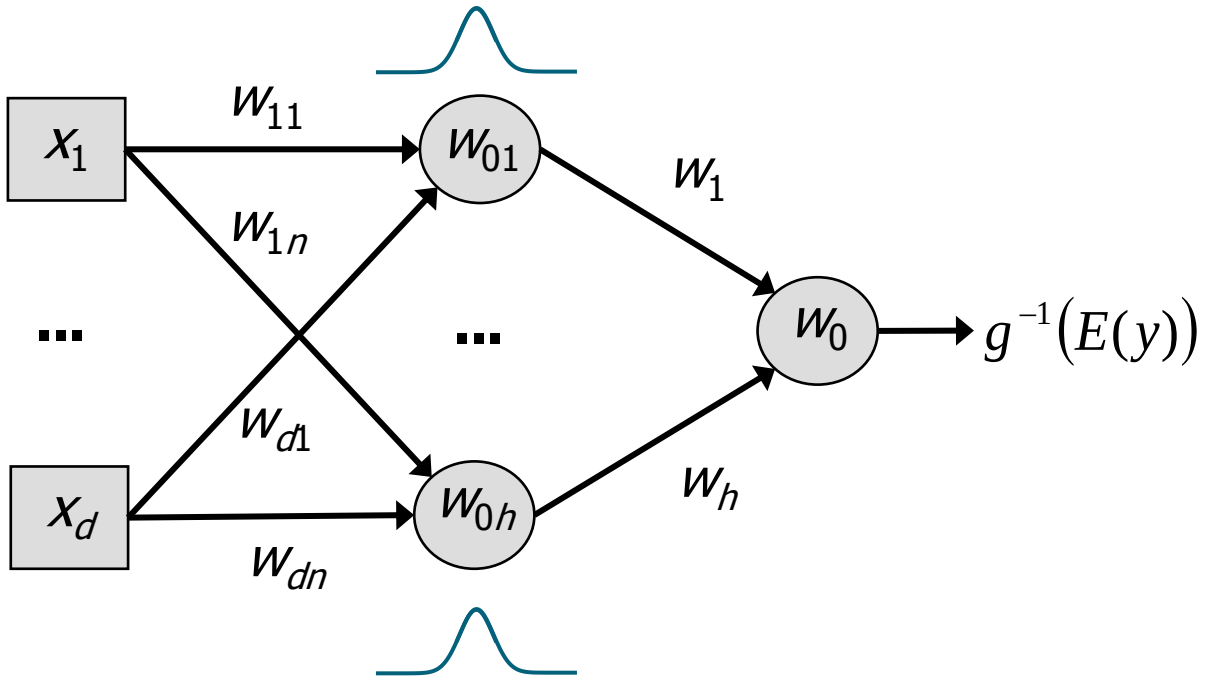
Overview of Radial Basis Functions

- Ordinary Radial Basis Functions (ORBF).
- Normalized Radial Basis Functions (NRBF).



Ordinary Radial Basis Functions

$$g^{-1}(E(y)) = w_0 + \sum_{i=1}^{h} w_i \underbrace{\exp\left[-w_{0i}\left(\sum_{j=1}^{d}(x_j - w_{ij})^2\right)\right]}_{\text{hidden unit}}$$

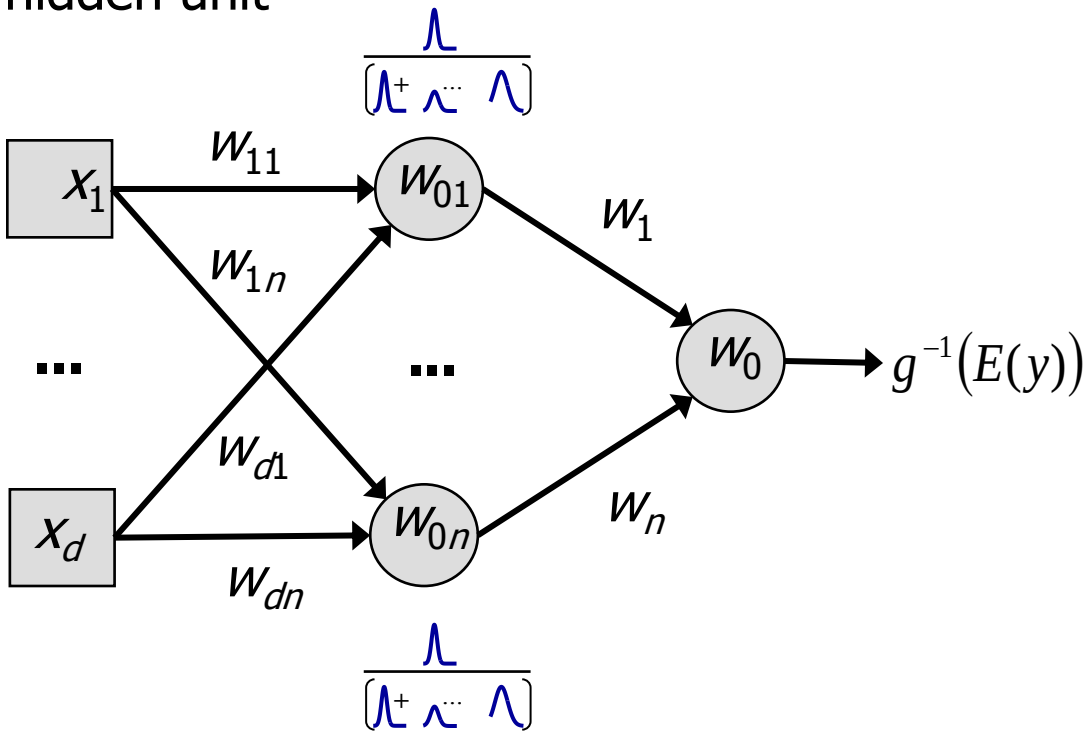


RBF Combination Functions

- XRADIAL Unequal Heights and Widths.
- EQRADIAL Equal Heights and Widths.
- EWRADIAL Equal Widths.
- EHRADIAL Equal Heights.
- EVRADIAL Equal Volumes.

Normalized Radial Basis Functions

$$g^{-1}(E(y)) = w_0 + \sum_{i=1}^{h} w_i \underbrace{\left(\frac{e_i}{\sum_{j=1}^{k} e_j} \right)}_{\text{hidden unit}} \quad \text{where} \quad e_i = \exp\left[f \cdot \ln(a_i) - w_{0i}^2 \left(\sum_{j=1}^{d} (x_j - w_{ji})^2 \right) \right]$$



Constructing Custom Neural Networks

```
PROC NEURAL DATA=<data> DMDBCAT=<catalog>;
      INPUT <inputs> / LEVEL=<input level>;
     TARGET <targets> / LEVEL=<target level>;
     ARCHI <architecture-name>;
     PRELIM <starts> MAXITER=<iterations>;
     TRAIN;
RUN;
```

PROC NEURAL

- underlies the Neural Network node
- enables you to construct virtually any feed-forward neural network architecture.

The PROC NEURAL/Architecture Statement

PROC NEURAL DATA=<*libref.*>*SAS-data-set*
DMDBCAT=*catalog* <*option-list*>**;**

- The PROC NEURAL statement invokes the neural network procedure.
- Options include the ability to read in saved networks and to assign validation and test data sets.
- The SAS data set must already have been cataloged by means of the DMDB procedure.

ARCHI *architecture-name* <**HIDDEN=***n*> <**DIRECT**>**;**

- The ARCHITECTURE statement constructs a network with either zero (a linear model) or one hidden layers.
- The statement sets the appropriate COMBINE= and ACT= functions, based on the specified *architecture-name*.

The TARGET/TRAIN Statement

TARGET | OUTPUT *variable-list* /
<**ACT**=*activation-function*>
<**BIAS|NOBIAS** >
<**COMBINE**=*combination-function*>
<**ERROR**=*keyword*>
<**ID**=*name*>
<**LEVEL**=*value*>
<**MESTA**=*number*>
<**MESTCON**=*number*>
<**SIGMA**=*number*>
<**STD**=*method*>**;**

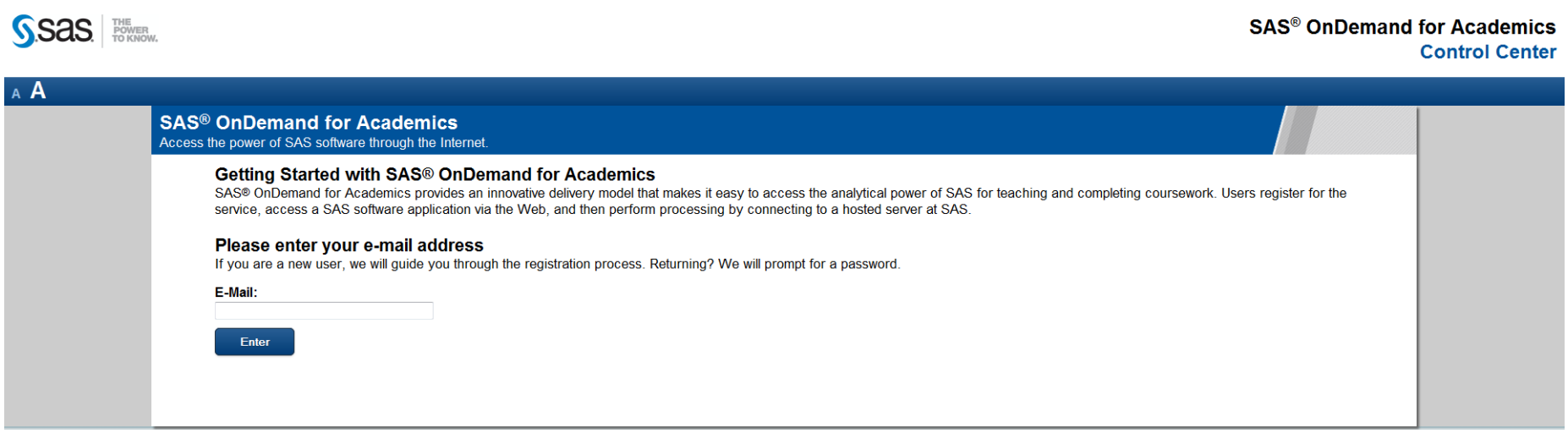
- The TARGET statement identifies the target variables.
- It is also used to specify the target layer activation and error functions.

TRAIN OUT=<*libref.*>*SAS-data-set*
OUTEST=<*libref.*>*SAS-data-set*
OUTFIT=<*libref.*>*SAS-data-set*
<**ACCEL|ACCELERATE**=*number*>
<**DECEL|DECELERATE**=*number*>
<**DUMMIES | NODUMMIES**>
<**ESTITER**=*i*>
<**LEARN**=*number*>
<**MAX|MAXMOMENTUM**=*number*>
<**MAXITER**=*integer*>
<**MAXLEARN**=*number*>
<**MAXTIME** =*number*>
<**MINLEARN**=*number*>
<**MOM|MOMENTUM**=*number*>
<**TECHNIQUE**=*name*>**;**

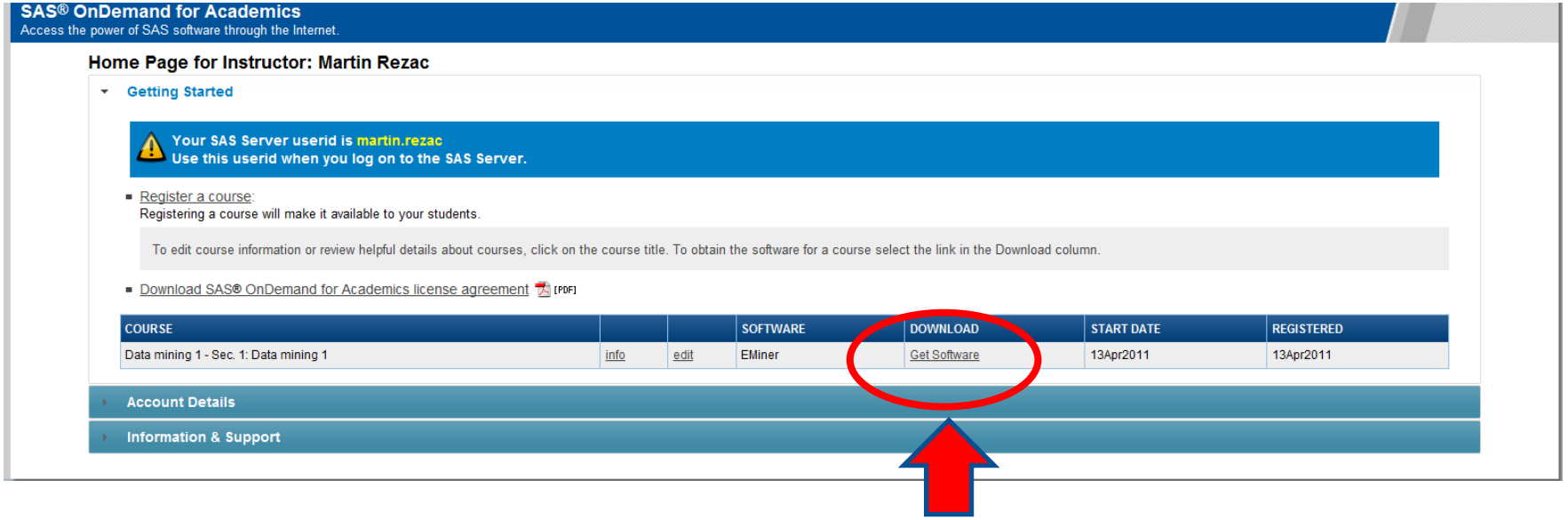
SAS OnDemand for Academics – Enterprise Miner

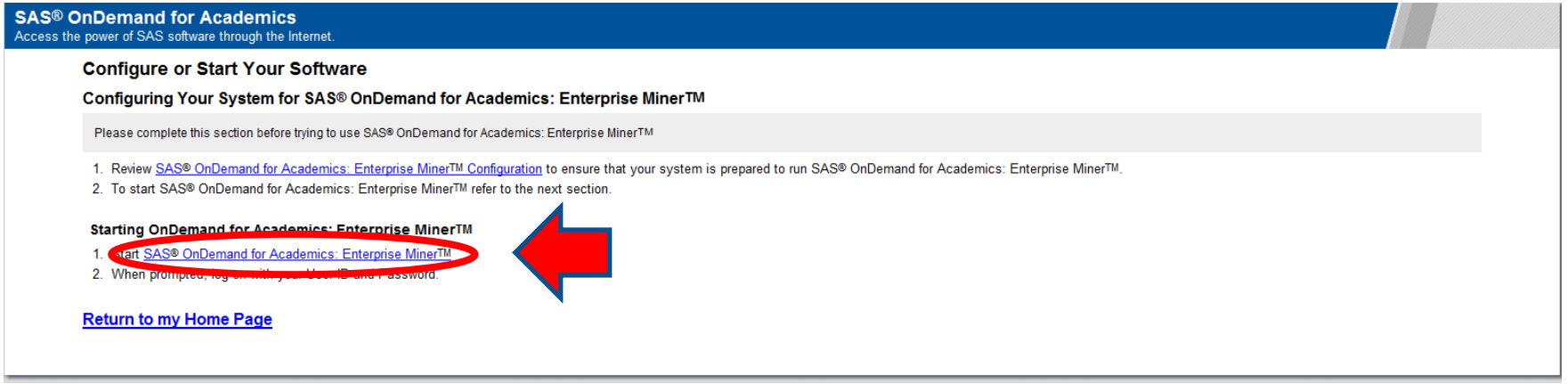
http://support.sas.com/ondemand/account.html

http://support.sas.com/ctx/sodareg/

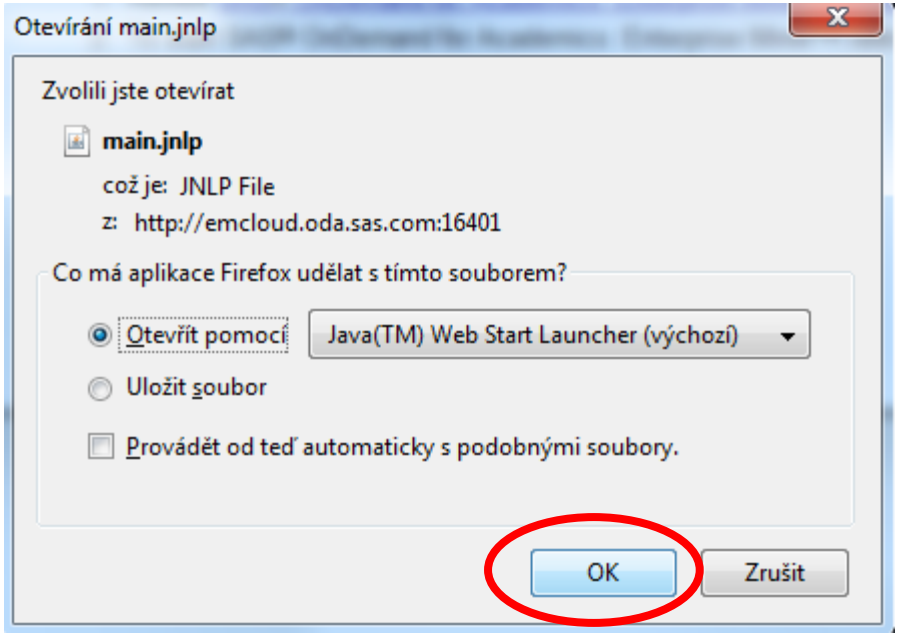


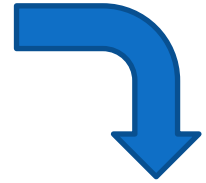
SAS OnDemand for Academics – Enterprise Miner

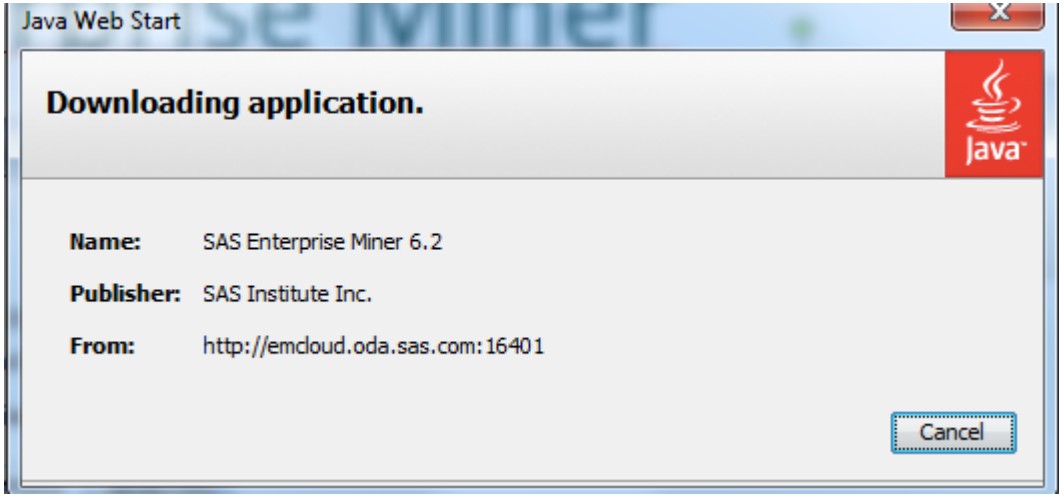




SAS OnDemand for Academics – Enterprise Miner

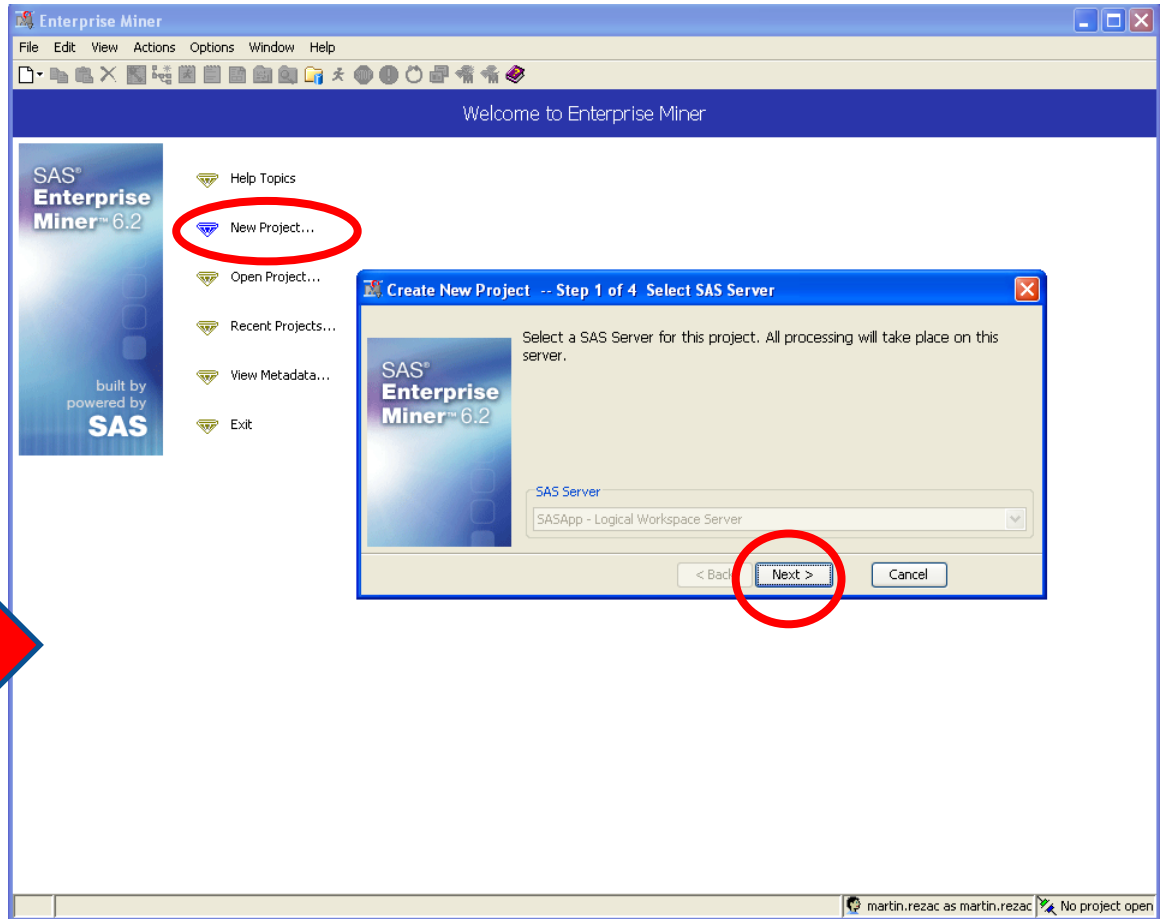




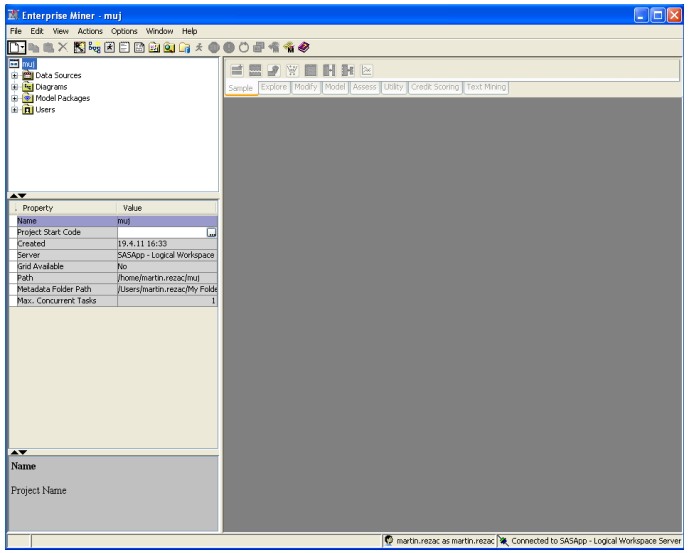


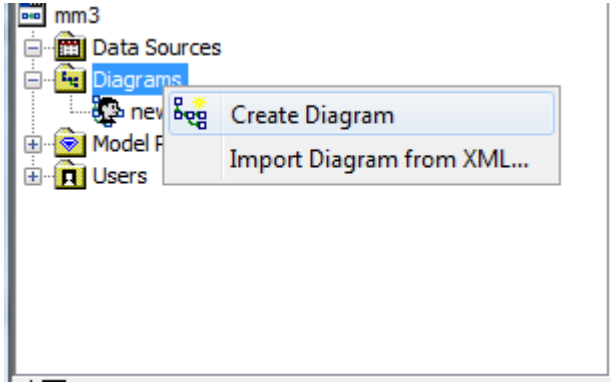
- POZOR!!! Do počítače se stáhne cca 125 MB!

SAS OnDemand for Academics – Enterprise Miner

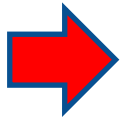


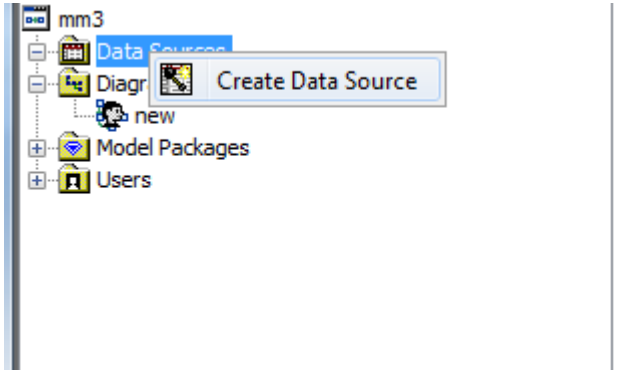
SAS OnDemand for Academics – Enterprise Miner



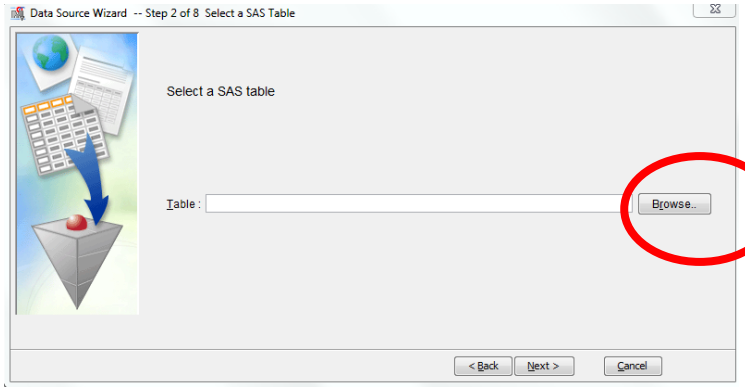


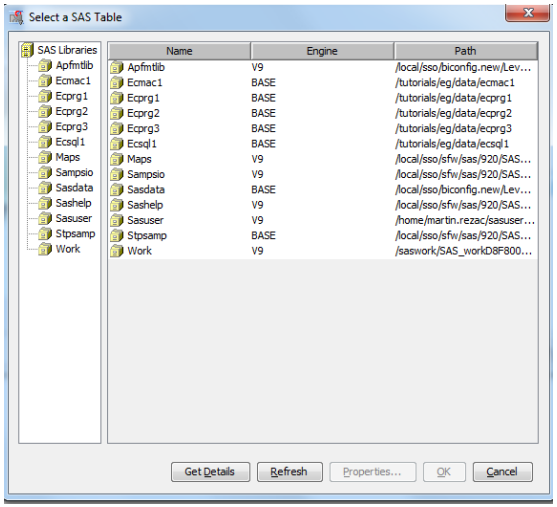
- Je třeba vytvořit diagram a připojit datové zdroje.

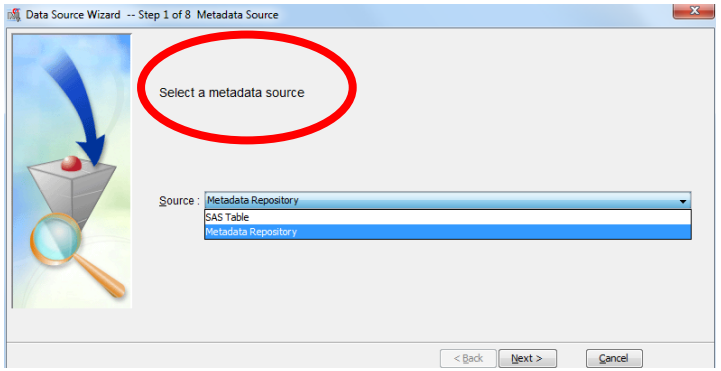


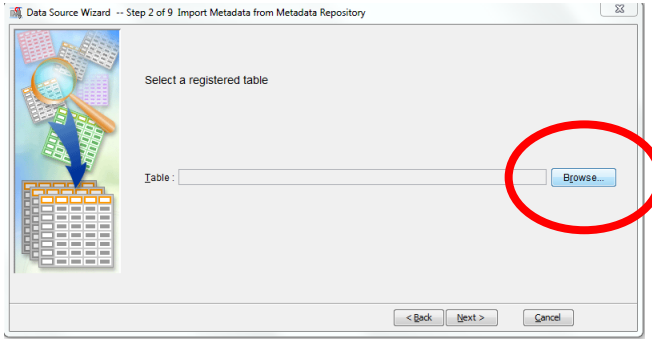


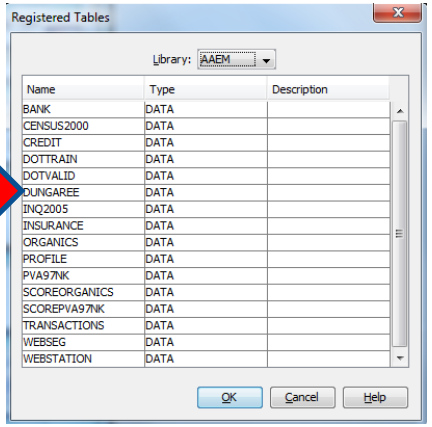
SAS OnDemand for Academics – Enterprise Miner



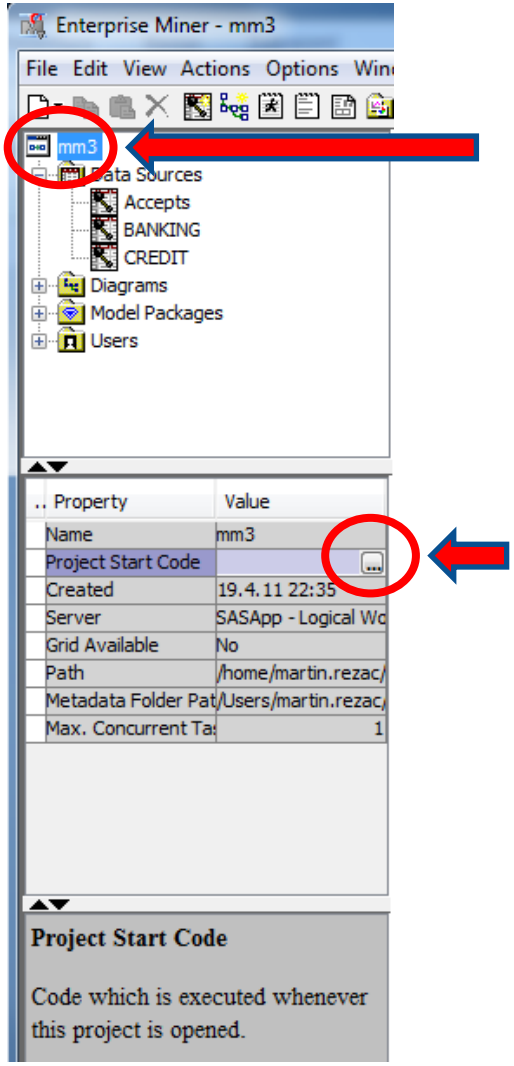






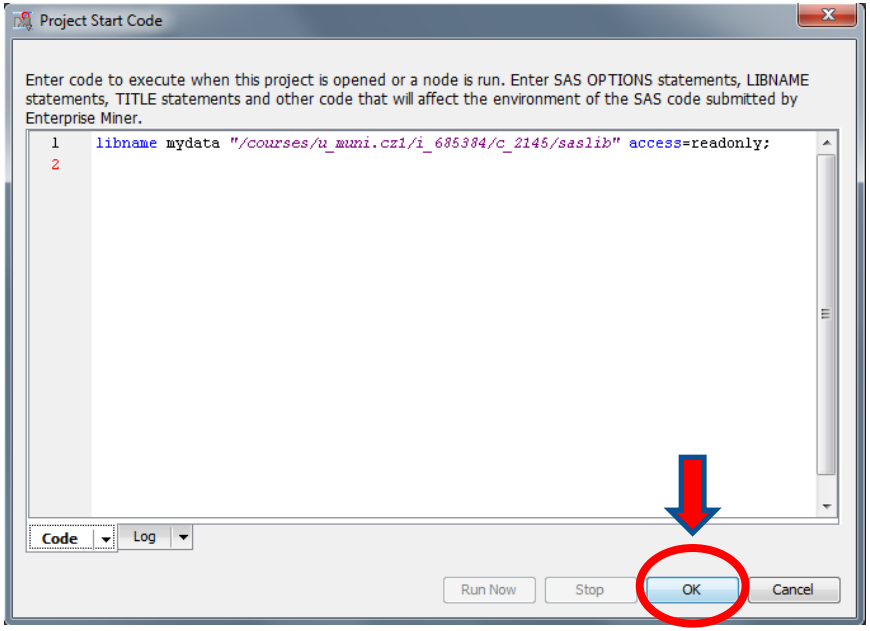


SAS OnDemand for Academics – Enterprise Miner

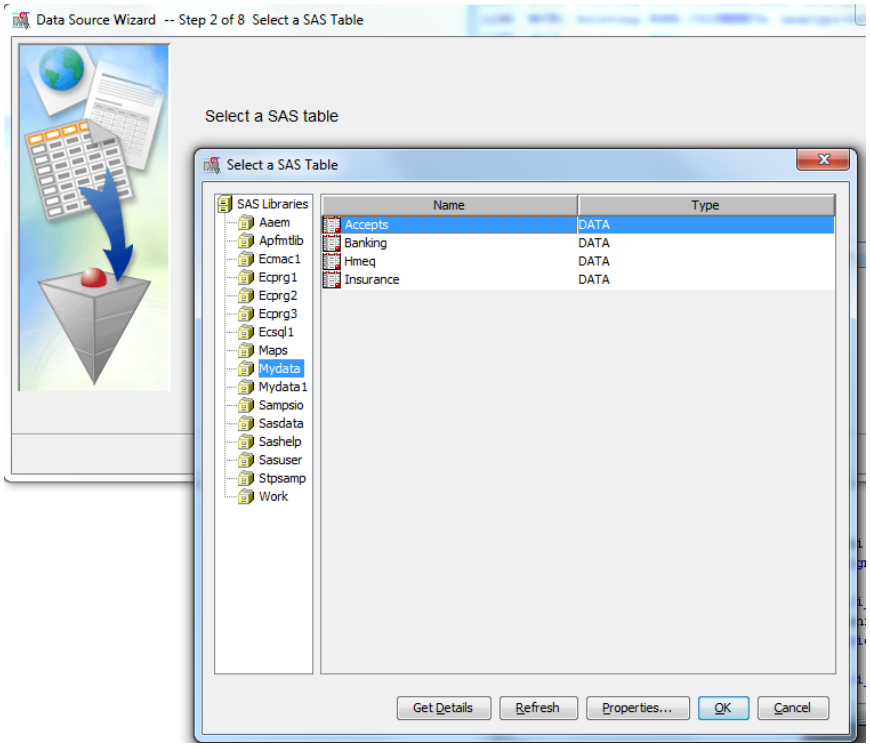


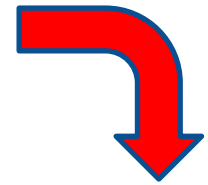
Pro namapování knihovny našeho kurzu je třeba:

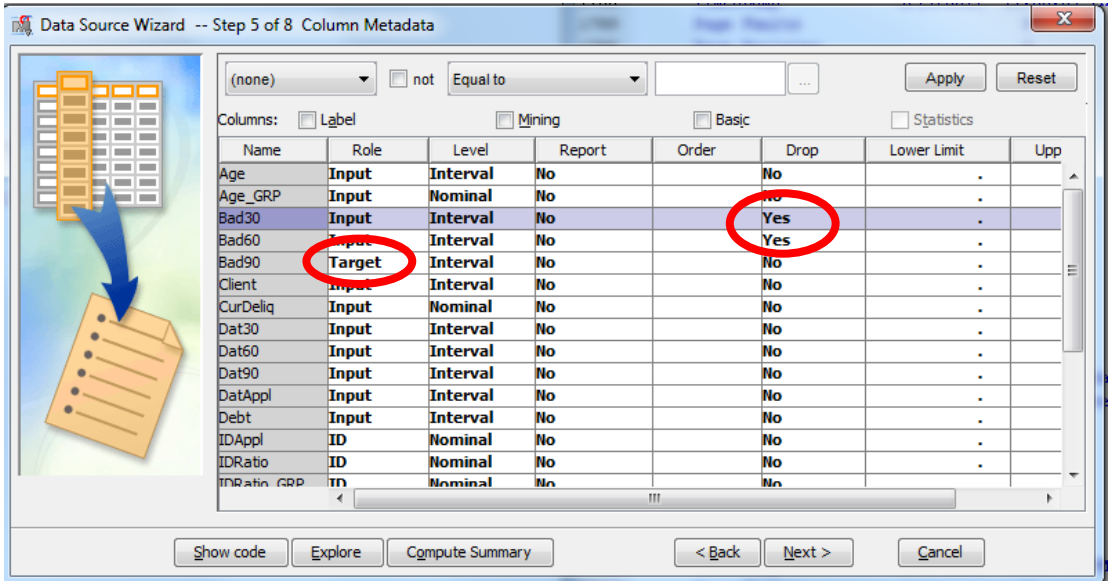
- kliknout na název projektu
- kliknout na „…“ pro editaci „Project Start Code“
- napsat kód pro připojení knihovny
- kliknout na „OK“



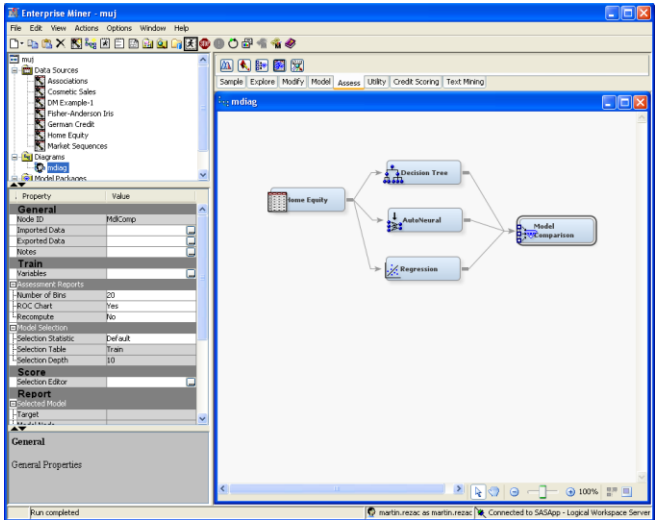
SAS OnDemand for Academics – Enterprise Miner

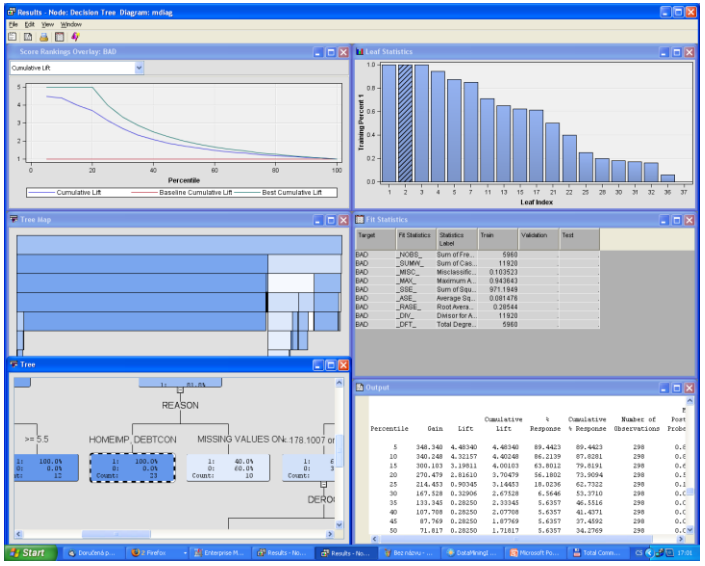


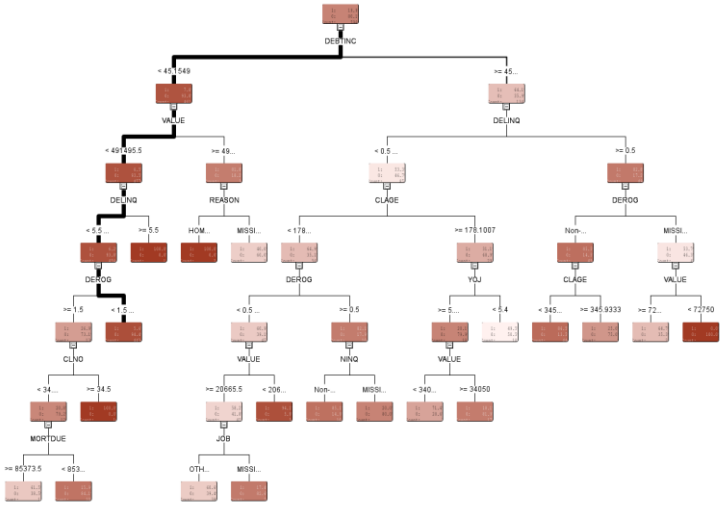


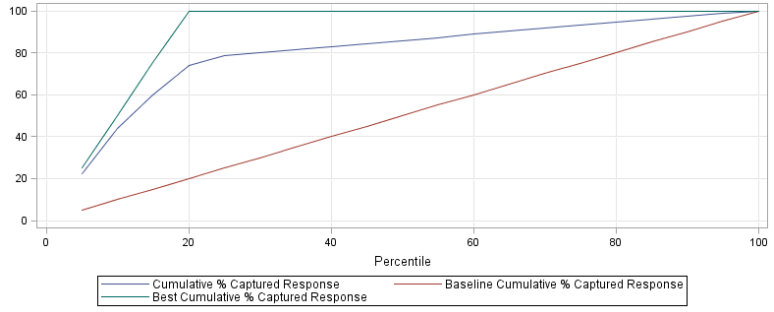


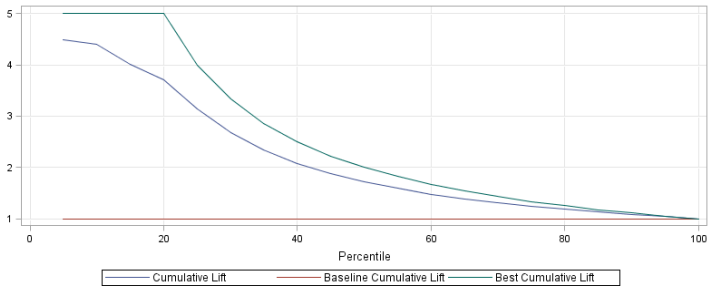
Decision Tree



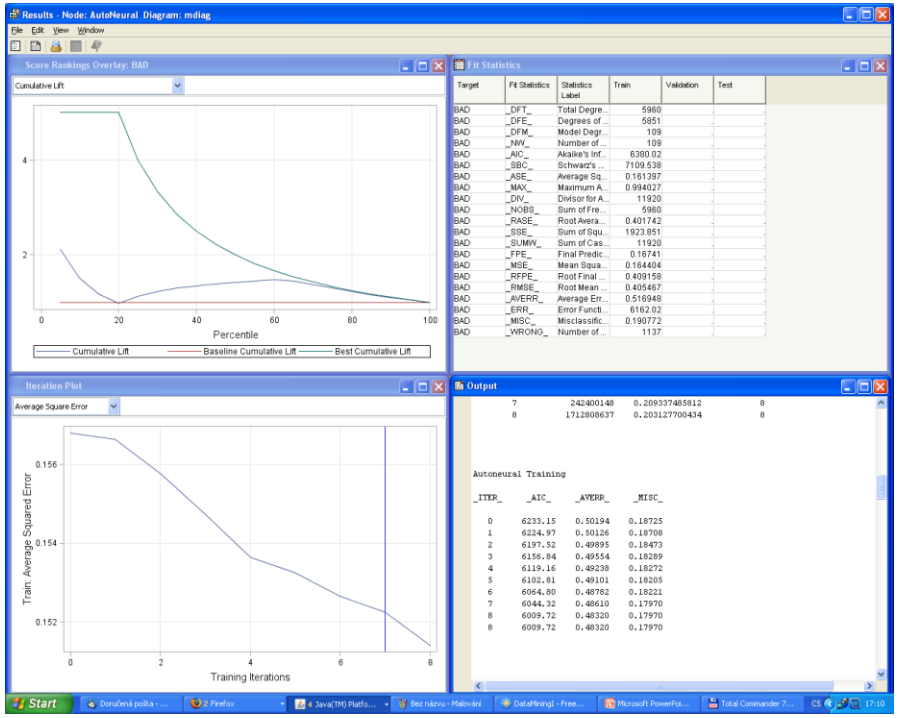


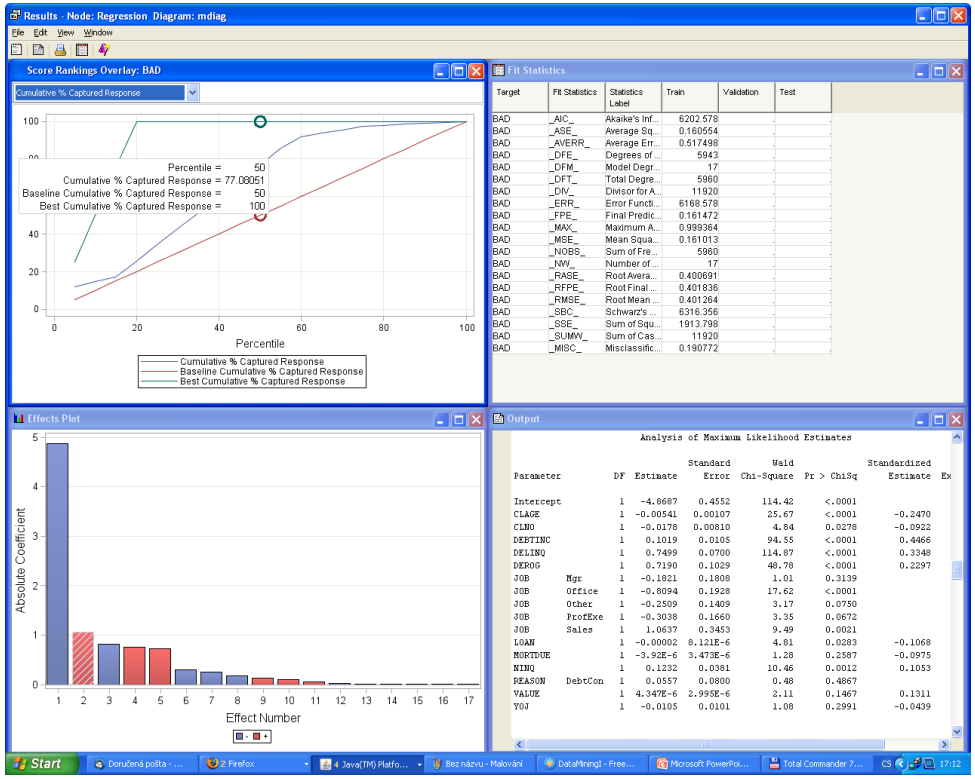




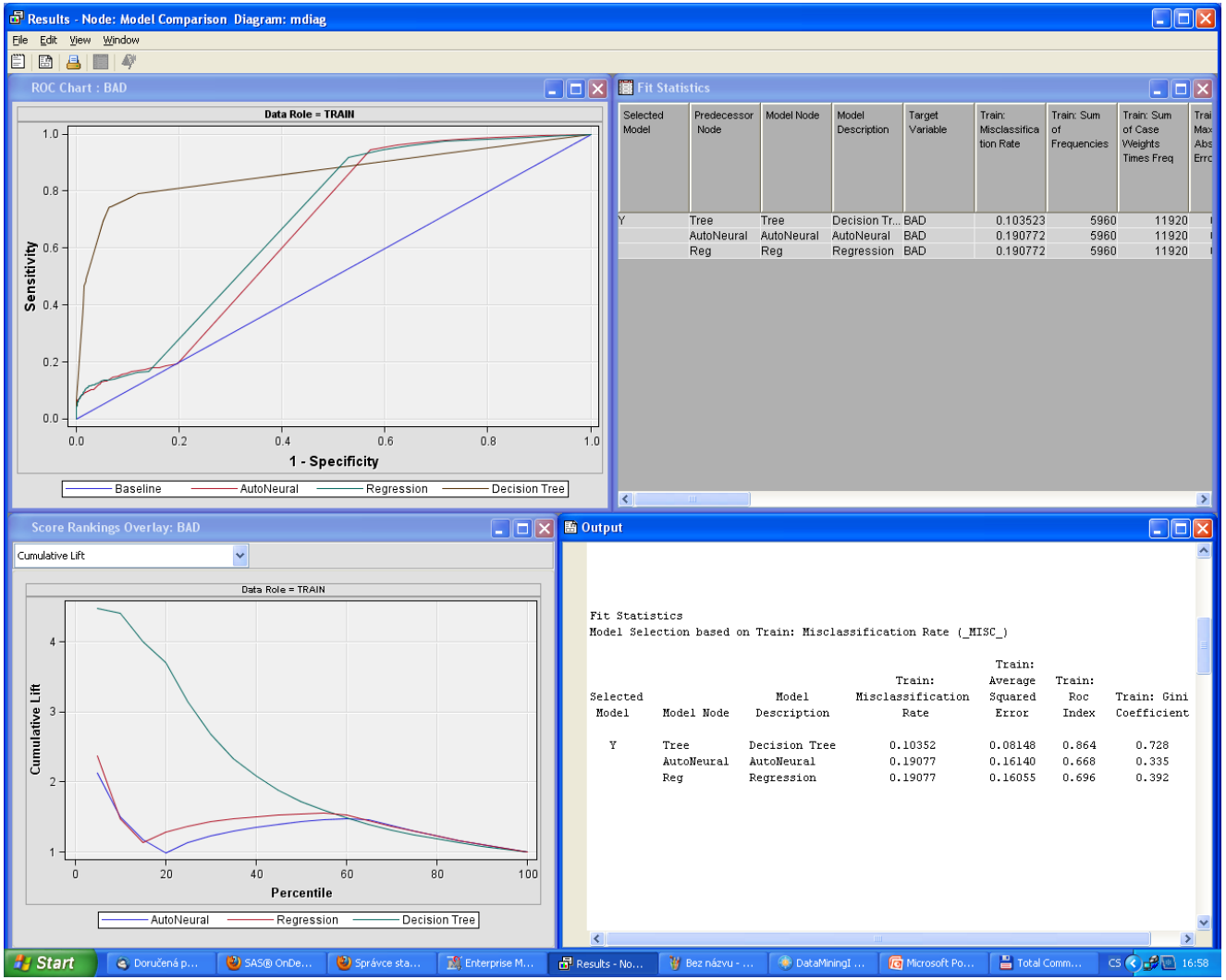


Neural Network

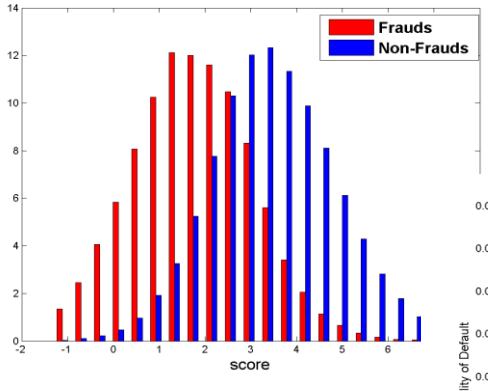


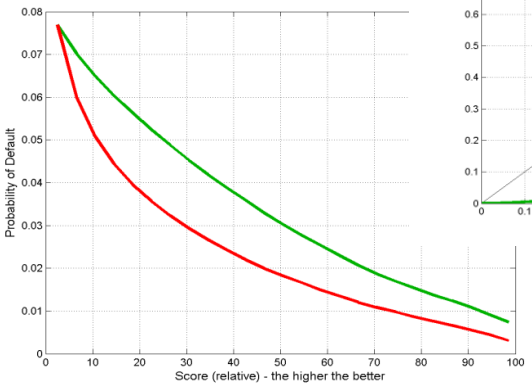


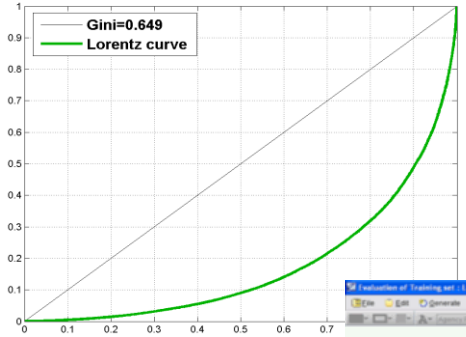
Porovnání modelů (Model Comparison)

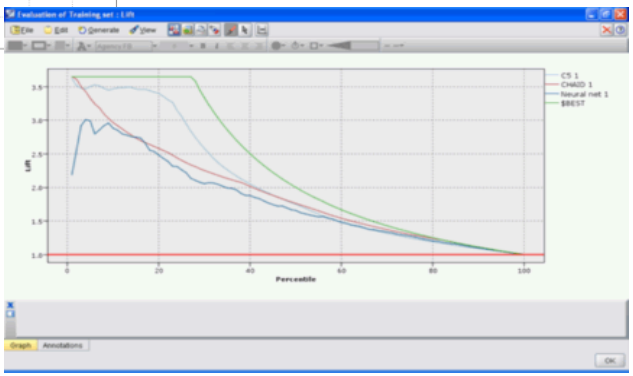


11. Evaluace modelu – LC(ROC),CAP, Gini, KS, Lift, IV









Měření kvality modelu

- Je nemožné využívat predikční modely efektivně bez znalosti jejich kvality/diskriminační síly.
- Většinou je k dispozici celá řada modelů a je třeba vybrat jen jeden – ten nejlepší.

- Uvažujeme dva základní skupiny indexů kvality. První je založena na distribuční funkci. Mezi nejpoužívanější indexy patří

  - Kolmogorovova-Smirnovova statistika (KS)
  - Giniho index (Somersovo D, Kendalovo τ_α , Goodman-Kruskal γ)
  - C-statistika
  - Lift.

- Druhá skupina indexů je založena na pravděpodobnostní hustotě. Mezi nejznámější indexy patří

  - Střední diference (Mahalanobisova vzdálenost)
  - Informační statistika/hodnota (I_{Val}).

Indexy založené na distribuční funkci

$$D_K = \begin{cases} 1, & klient \ \ je\ dobrý \\ 0, & jinak. \end{cases}$$

Počet dobrých klientů: n

Počet špatných klientů: m

Proporce dobrých/špatných klientů: $p_G = \frac{n}{n+m}$, $p_B = \frac{m}{n+m}$

- Empirické distribuční funkce:

$$F_{n.GOOD}(a) = \frac{1}{n}\sum_{i=1}^{n} I(s_i \le a \wedge D_K = 1)$$

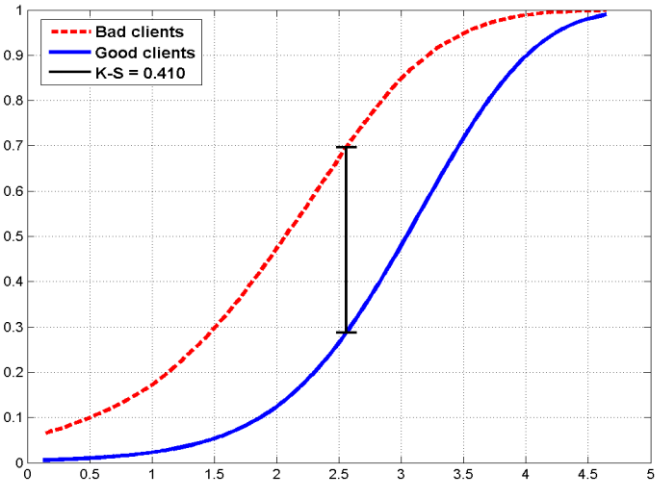
$$F_{m.BAD}(a) = \frac{1}{m}\sum_{i=1}^{m} I(s_i \le a \wedge D_K = 0)$$

$$F_{N.ALL}(a) = \frac{1}{N}\sum_{i=1}^{N} I(s_i \le a) \qquad a \in [L, H]$$

$$I(A) = \begin{cases} 1 & A \ \ platí \\ 0 & jinak \end{cases}$$

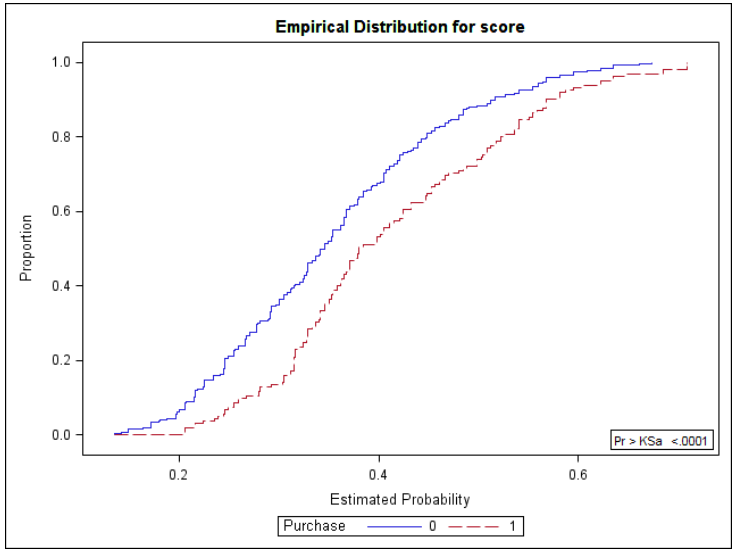
- **Kolmogorovova-Smirnovova statistika (KS)**

$$KS = \max_{a \in [L,H]} \left| F_{m,BAD}(a) - F_{n,GOOD}(a) \right|$$



KS – výpočet v SASu

```
ods graphics on;
proc npar1way edf plots=edfplot
data=st192.sales_score;
      class purchase;
      var score;
run;
ods graphics off;
```



Kolmogorov-Smirnov Test for Variable score
Classified by Variable Purchase

Purchase	N	EDF at Maximum	Deviation from Mean at Maximum
0	269	0,364312	1,408701
1	162	0,135802	-1,815254
Total	431	0,278422	

Maximum Deviation Occurred at Observation 293
Value of score at Maximum = 0,300127

Kolmogorov-Smirnov Two-Sample Test (Asymptotic)
KS 0,110678 D 0,228510
KSa 2,297734 Pr > KSa <,0001

Cramer-von Mises Test for Variable score
Classified by Variable Purchase

Purchase	N	Summed Deviation from Mean
0	269	0,799555
1	162	1,327656

Cramer-von Mises Statistics (Asymptotic)
CM 0,004936 CMa 2,127210

Kuiper Test for Variable score
Classified by Variable Purchase

Purchase	N	Deviation from Mean
0	269	0,228510
1	162	0,000000

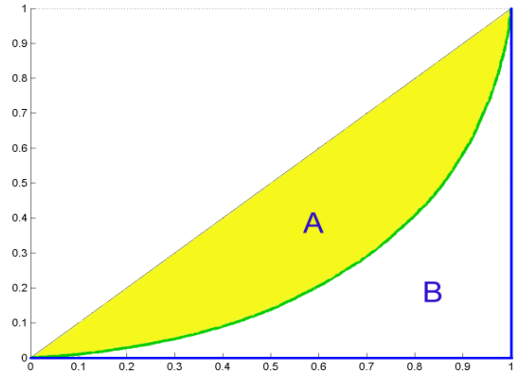
Kuiper Two-Sample Test (Asymptotic)
K 0,228510 Ka 2,297734 Pr > Ka 0,0010

Více viz:
http://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_npar1way_a0000000202.htm

Indexy založené na distribuční funkci

- Lorenzova křivka (LC)

$$x = F_{m.BAD}(a)$$
$$y = F_{n.GOOD}(a), a \in [L, H].$$

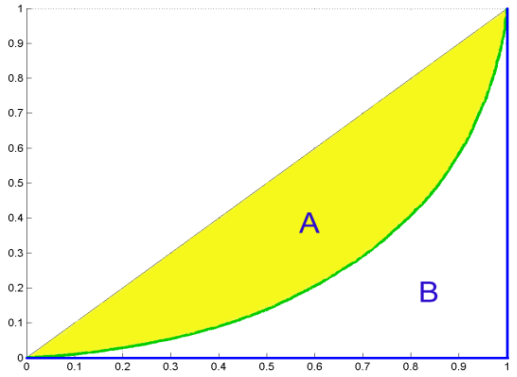


- Tato definice a název (LC) je konzistentní s Müller, M., Rönz, B. (2000). Stejnou definici křivky, ovšem pod názvem ROC lze nalézt v Thomas et al. (2002). Siddiqi (2006) používá název ROC pro křivku s prohozenými osami a LC pro křivku s $F_{m.BAD}(a)$ na svislé ose a $F_{N.ALL}(a)$ na ose horizontální.

Lorenzova křivka, Giniho index

- Lorenzova křivka (LC)

$$x = F_{m.BAD}(a)$$
$$y = F_{n.GOOD}(a),\ a \in [L, H].$$



- Giniho index

$$Gini = \frac{A}{A+B} = 2A$$

$$Gini = 1 - \sum_{k=2}^{n+m} \left(F_{m.BAD\,k} - F_{m.BAD\,k-1}\right) \cdot \left(F_{n.GOOD\,k} + F_{n.GOOD\,k-1}\right)$$

kde $F_{m.BAD\,k}$ ($F_{n.GOOD_k}$) je k-tá hodnota vektoru empirické distribuční funkce špatných (dobrých) klientů

Somersovo D, Kendalovo τ_α

- Giniho index je speciální případ Somersova *D* (Somers (1962)), které je pořadovou asociační mírou definovanou jako

$$D_{YX} = \frac{\tau_{XY}}{\tau_{XX}}$$

kde τ_{XY} je Kendallovo τ_a definované jako $\tau_{XY} = E\left[sign(X_1 - X_2)sign(Y_1 - Y_2)\right]$

kde $(X_1, Y_1), (X_2, Y_2)$ jsou bivariantní, stochasticky nezávislé, náhodné vektory nad touž datovou populací, a $E[\cdot]$ značí střední hodnotu. V našem případě je *Y=1* jestliže je klient dobrý a *Y=0* jestliže je klient špatný. Proměnná *X* reprezentuje skóre.

Thomas (2009) uvádí, že Somersovo *D* hodnotící výkonnost daného credit scoringového modelu lze vypočítat pomocí

$$D_S = \frac{\sum_i g_i \sum_{j<i} b_j - \sum_i g_i \sum_{j>i} b_j}{n \cdot m}$$

kde g_i (b_j) je počet dobrých (špatných) klientů v i-tém intervalu skóre.

Somersovo D, Mann-Whitney U

- Dále platí, že D_S může být vyjádřeno pomocí Mann-Whitneyho U-statistiky.
  - Seřaď datový vzorek ve vzestupném pořadí podle skóre a sečti pořadí dobrých klientů ve vzniklé posloupnosti. Označme tento součet jako R_G. Potom

$$U = R_G - \frac{1}{2} n(n+1)$$

$$D_S = 2\frac{U}{n \cdot m} - 1$$

Konkordantní, diskordantní páry

- Konkordantní pár (X_1,Y_1), (X_2,Y_2):

$$\operatorname{sgn}(X_2 - X_1) = \operatorname{sgn}(Y_2 - Y_1)$$

- Diskordantní pár:

$$\operatorname{sgn}(X_2 - X_1) = -\operatorname{sgn}(Y_2 - Y_1)$$

- V našem případě X představuje skóre a Y ukazatel dobrého klienta (D_K). Protože dobrý klient má hodnotu Y=1 a špatný Y=0, je zřejmé, že u konkordantního páru má dobrý klient vyšší hodnotu skóre než klient špatný.

Somersovo D, Goodman-Kruskal gamma

- Uvažujme tedy dva náhodně vybrané klienty, přičemž jeden je dobrý (Y_1=1) a druhý špatný (Y_2=0), skóre prvního označme s_1, druhého s_2. Pak
  - Konkordantní pár (Concordant): $s_1>s_2$
  - Diskordantní pár (Discordant): $s_1<s_2$
  - Vázaný pár (Tied): $s_1=s_2$

➢ Somersovo D:

$$D_S = \frac{\#Concordant - \#Discodrant}{\#Concordant + \#Discodrant + \#Tied}$$

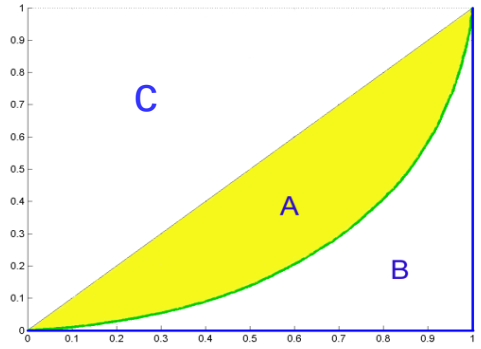
➢ Goodmanovo-Kruskalovo Gamma:

$$\gamma = \frac{\#Concordant - \#Discodrant}{\#Concordant + \#Discodrant}$$

Indexy založené na distribuční funkci

- C-statistika:

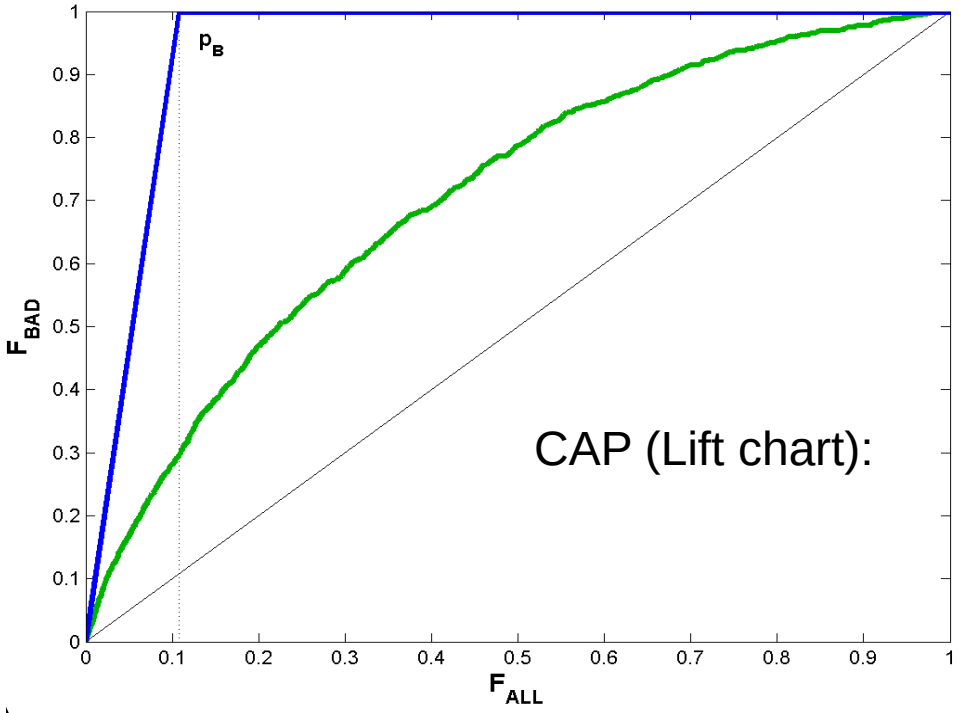
$$c-stat = A + C = \frac{1 + Gini}{2}$$



Tato statistika je rovna pravděpodobnosti, že náhodně vybraný dobrý klient má vyšší skóre než náhodně vybraný špatný klient, tj.

$$c-stat = P(s_1 \geq s_2 \mid D_{K_1} = 1 \wedge D_{K_2} = 0)$$

CAP – index AR



AR (Accuracy Ratio)

$$AR = \frac{\text{Plocha mezi CAP a diagonálou}}{\text{Plocha mezi CAP ideálního modelu a diagonálou}}$$

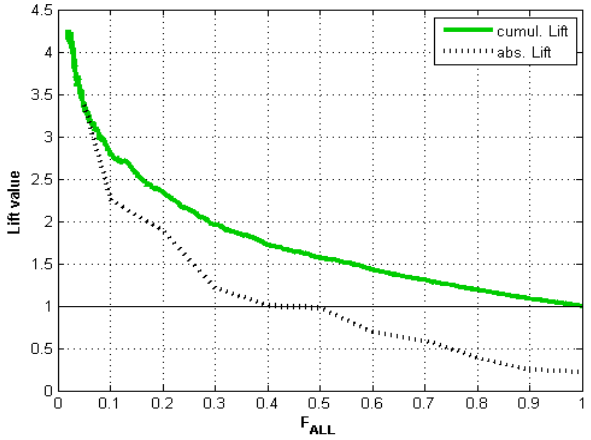
$$= \frac{\text{Plocha mezi CAP a diagonálou}}{0.5(1-p_B)} = Gini$$

V tomto případě máme na x-ové ose proporci všech klientů (F_{ALL}) a na y-vé ose proporci špatných klientů (F_{BAD}). Ideální model je tentokrát reprezentován lomenou čarou z bodu [0, 0] přes [pB, 1] do bodu [1, 1]. Výhoda tohoto obrázku je ta, že je možné odečíst proporci zamítnutých špatných klientů vs. celková proporce zamítnutých klientů. Např. vidíme, že pokud chceme zamítnout 70% špatných klientů, musíme zamítat přibližně 40% všech žadatelů.

Indexy založené na distribuční funkci

- Další možnou mírou kvality scoringového modelu je Lift, který říká kolikrát je daný model, při dané úrovni zamítání, lepší než náhodný model. Přesněji řečeno jde o poměr proporce špatných klientů se skóre menším nebo rovno dané hodnotě skóre a, $a \in [L, H]$, ku proporci špatných klientů v celé populaci. Formálně jej lze zapsat takto:

$$Lift(a) = \frac{CumBadRate(a)}{BadRate} = \frac{\frac{\sum_{i=1}^{n+m} I(s_i \le a \wedge Y = 0)}{\sum_{i=1}^{n+m} I(s_i \le a)}}{\frac{\sum_{i=1}^{n+m} I(Y = 0)}{\sum_{i=1}^{n+m} I(Y = 0 \vee Y = 1)}} = \frac{\frac{\sum_{i=1}^{n+m} I(s_i \le a \wedge Y = 0)}{\sum_{i=1}^{n+m} I(s_i \le a)}}{\frac{n}{N}}$$



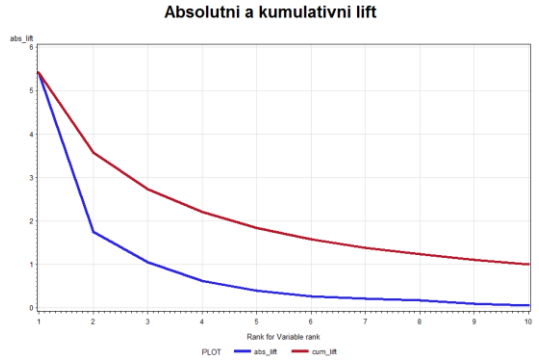
$$absLift(a) = \frac{BadRate(a)}{BadRate}$$

Lift v SASu

```
%macro lift(data=,score=,response=); /*Vypsani tabulky liftu +
vykresleni*/

/* Vytvoreni poradi dle skore*/
  proc sort data=&data;
    by &score;
  run;

  data work.score;
  set &data;
    rank+1;
  run;

/*Rozdeleni na decily dle skore*/
proc rank data=score groups=10 out=score;
  var rank;
  ranks decile;
run;

data score;
  set score;
  decile=decile+1;
run;

/*vytvoreni tabulky pro lift*/
proc sql;
create table work.lift1 as
select decile,count(*) as N,sum(&response) as N_of_bad, (calculated
N_of_bad)/(calculated N)*100 as bad_rate,
      (calculated bad_rate)/(select 100*sum(&response)/count(*) from score) as
abs_lift
  from score
  group by decile;
quit;
```

```
/*Vypocet kumulativniho liftu*/
proc sql;
create table work.lift2 as
select *, (select sum(N_of_bad)/sum(N) from work.lift1 as a where
a.decile<=b.decile)/(select sum(&response)/count(*) from work.score) as cum_lift
from work.lift1 as b;
quit;

/*Vypis tabulky pro lift*/

title „Lift";
proc print data=work.lift2 nonobs;
format bad_rate 4.2
       abs_lift 5.3
       cum_lift 5.3;
run;

/*vykresleni liftu*/
proc gplot data=work.lift2;
title 'Absolutni a kumulativni lift';
plot (abs_lift cum_lift)*decile /overlay legend grid;
symbol interpol=join w=5;
run;
quit;

%mend lift;
```

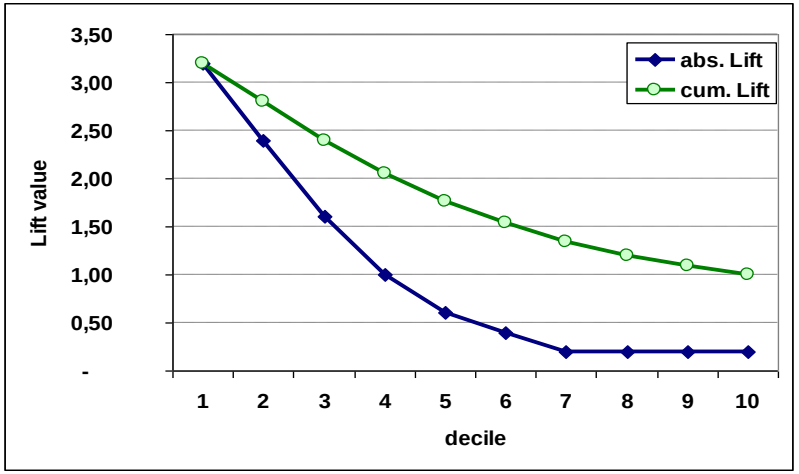
%***lift(data=comp1.score_age,score=score,response=SeriousDlqin2yrs);***

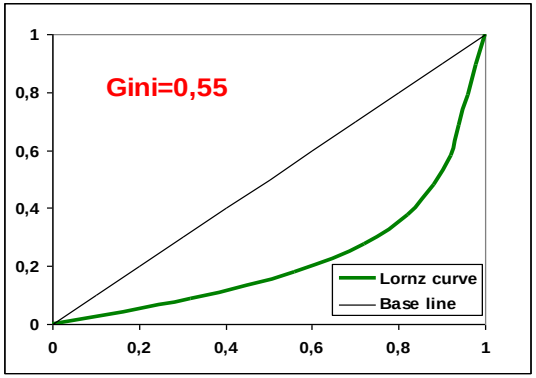


Indexy založené na distribuční funkci

- Pro výpočet lze použít tabulku s počty všech a špatných klientů v daných intervalech skóre (např. decilech).

decile	# cleints	absolutely			cumulatively		
		# bad clients	Bad rate	abs. Lift	# bad clients	Bad rate	cum. Lift
1	100	16	16,0%	3,20	16	16,0%	3,20
2	100	12	12,0%	2,40	28	14,0%	2,80
3	100	8	8,0%	1,60	36	12,0%	2,40
4	100	5	5,0%	1,00	41	10,3%	2,05
5	100	3	3,0%	0,60	44	8,8%	1,76
6	100	2	2,0%	0,40	46	7,7%	1,53
7	100	1	1,0%	0,20	47	6,7%	1,34
8	100	1	1,0%	0,20	48	6,0%	1,20
9	100	1	1,0%	0,20	49	5,4%	1,09
10	100	1	1,0%	0,20	50	5,0%	1,00
All	1000	50	5,0%				

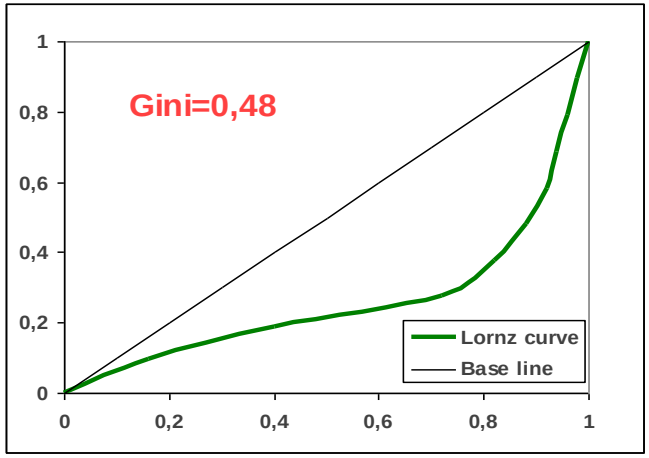


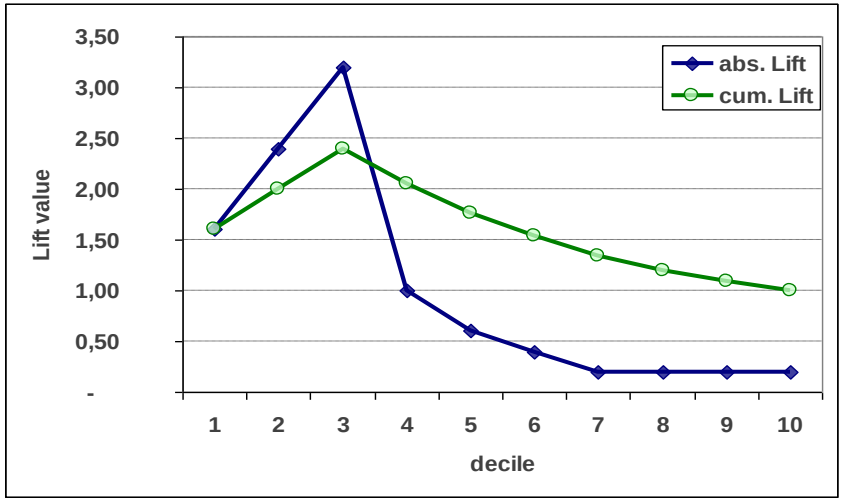


Indexy založené na distribuční funkci

- Pokud bad rate není monotonní:
  - LC vypadá OK
  - Gini se mírně sníží
  - Lift ovšem vypadá podivně

decile	# cleints	absolutely			cumulatively		
		# bad clients	Bad rate	abs. Lift	# bad clients	Bad rate	cum. Lift
1	100	8	8,0%	1,60	8	8,0%	1,60
2	100	12	12,0%	2,40	20	10,0%	2,00
3	100	16	16,0%	3,20	36	12,0%	2,40
4	100	5	5,0%	1,00	41	10,3%	2,05
5	100	3	3,0%	0,60	44	8,8%	1,76
6	100	2	2,0%	0,40	46	7,7%	1,53
7	100	1	1,0%	0,20	47	6,7%	1,34
8	100	1	1,0%	0,20	48	6,0%	1,20
9	100	1	1,0%	0,20	49	5,4%	1,09
10	100	1	1,0%	0,20	50	5,0%	1,00
All	1000	50	5,0%				



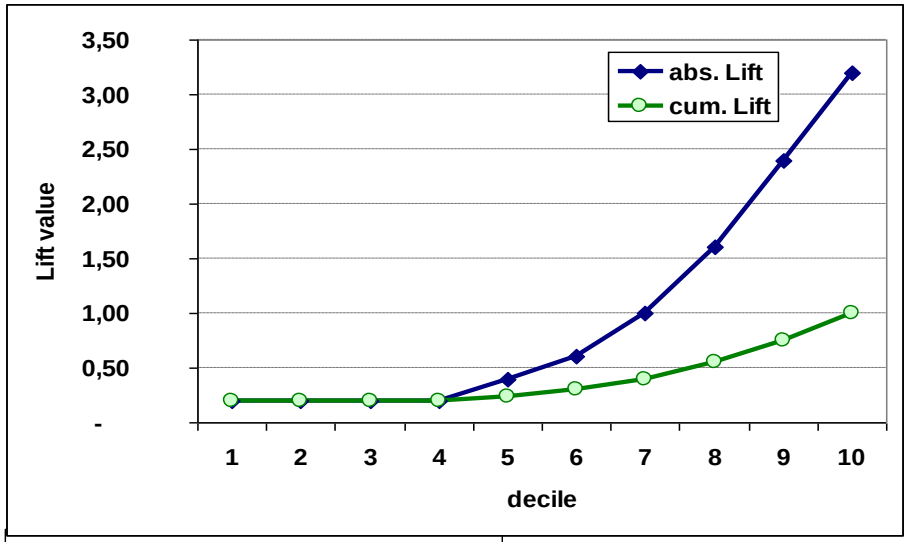


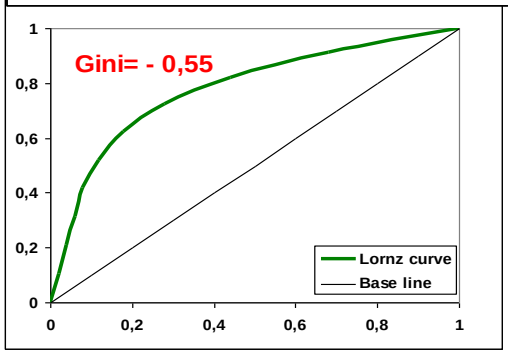
Indexy založené na distribuční funkci

- Pokud má skóre zcela opačný smysl, obdržíme „opačné“ obrázky.

decile	# cleints	absolutely			cumulatively		
		# bad clients	Bad rate	abs. Lift	# bad clients	Bad rate	cum. Lift
1	100	16	16,0%	3,20	16	16,0%	3,20
2	100	12	12,0%	2,40	28	14,0%	2,80
3	100	8	8,0%	1,60	36	12,0%	2,40
4	100	5	5,0%	1,00	41	10,3%	2,05
5	100	3	3,0%	0,60	44	8,8%	1,76
6	100	2	2,0%	0,40	46	7,7%	1,53
7	100	1	1,0%	0,20	47	6,7%	1,34
8	100	1	1,0%	0,20	48	6,0%	1,20
9	100	1	1,0%	0,20	49	5,4%	1,09
10	100	1	1,0%	0,20	50	5,0%	1,00
All	1000	50	5,0%				

decile	# cleints	absolutely			cumulatively		
		# bad clients	Bad rate	abs. Lift	# bad clients	Bad rate	cum. Lift
1	100	1	1,0%	0,20	1	1,0%	0,20
2	100	1	1,0%	0,20	2	1,0%	0,20
3	100	1	1,0%	0,20	3	1,0%	0,20
4	100	1	1,0%	0,20	4	1,0%	0,20
5	100	2	2,0%	0,40	6	1,2%	0,24
6	100	3	3,0%	0,60	9	1,5%	0,30
7	100	5	5,0%	1,00	14	2,0%	0,40
8	100	8	8,0%	1,60	22	2,8%	0,55
9	100	12	12,0%	2,40	34	3,8%	0,76
10	100	16	16,0%	3,20	50	5,0%	1,00
All	1000	50	5,0%				

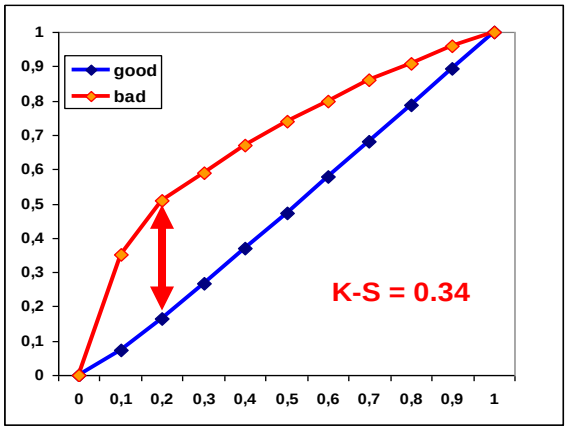


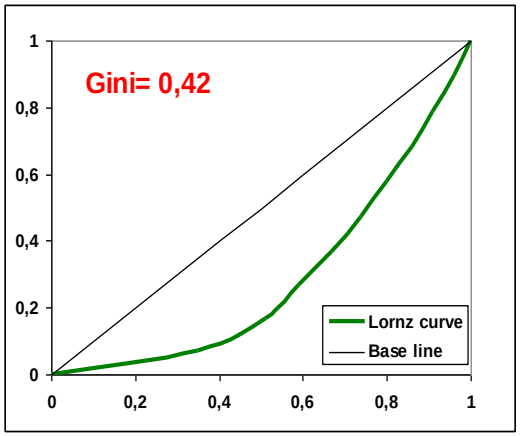


Indexy založené na distribuční funkci

- **SC 1:**

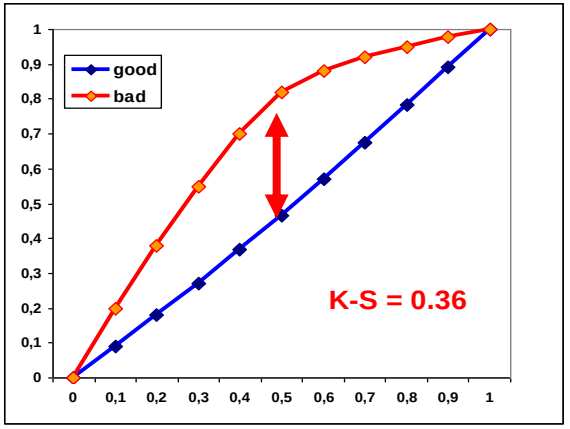
decile	# cleints	# bad clients	Bad rate
1	100	35	35,0%
2	100	16	16,0%
3	100	8	8,0%
4	100	8	8,0%
5	100	7	7,0%
6	100	6	6,0%
7	100	6	6,0%
8	100	5	5,0%
9	100	5	5,0%
10	100	4	4,0%
All	1000	100	10,0%

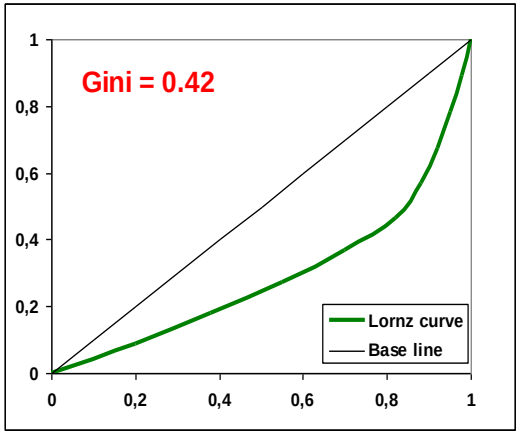




- **SC 2:**

decile	# cleints	# bad clients	Bad rate
1	100	20	20,0%
2	100	18	18,0%
3	100	17	17,0%
4	100	15	15,0%
5	100	12	12,0%
6	100	6	6,0%
7	100	4	4,0%
8	100	3	3,0%
9	100	3	3,0%
10	100	2	2,0%
All	1000	100	10,0%

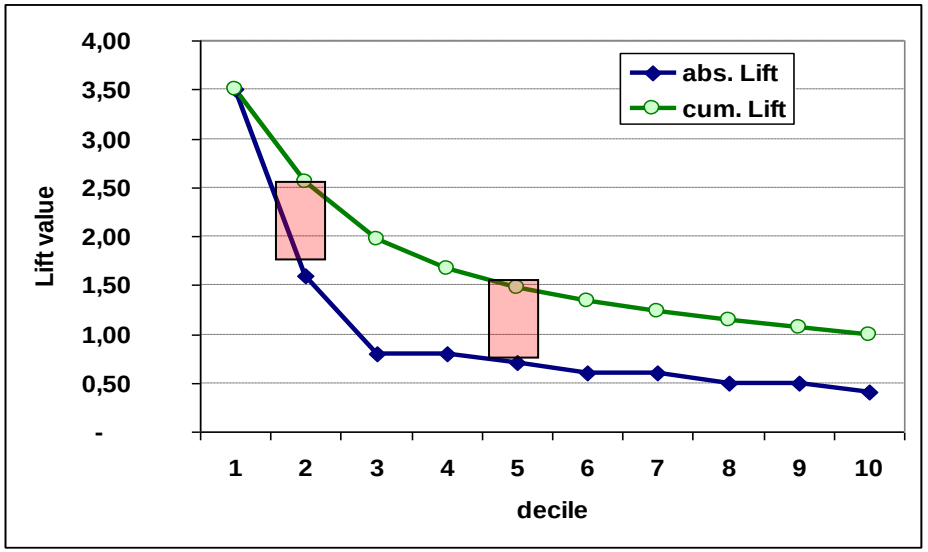




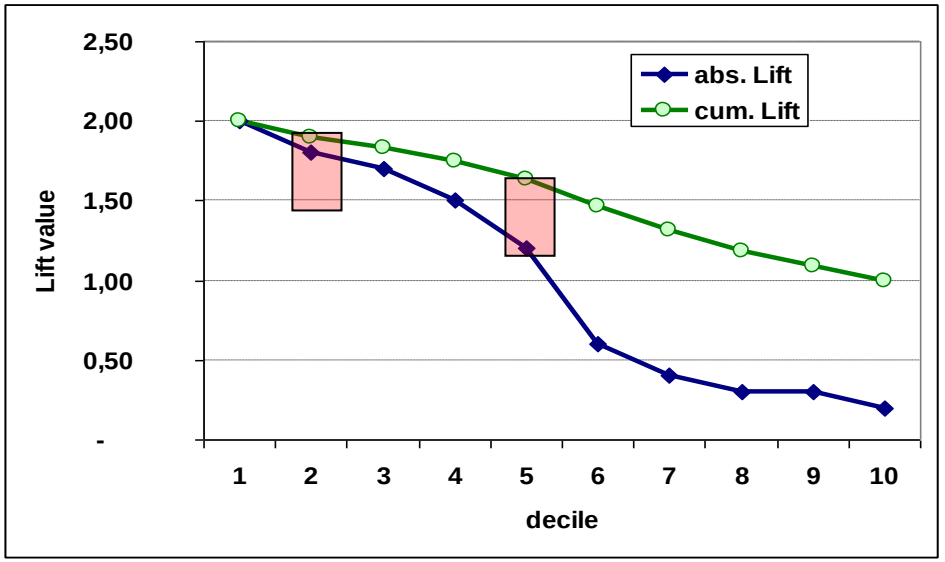
Je evidentní, že pouze Gini a KS nestačí!!!

Indexy založené na distribuční funkci

- **SC 1:**



- **SC 2:**



$Lift_{20\%} = 2.55$ > $Lift_{20\%} = 1.90$

$Lift_{50\%} = 1.48$ < $Lift_{50\%} = 1.64$

SC 2 je lepší, pokud je předpokládaná míra zamítaní (reject rate) přibližně 50%.
SC 1 je významně lepší, pokud je předpokládaný reject rate přibližně 20%.

Lift, QLift

- Lift can be expressed and computed by formula:

$$Lift(a) = \frac{F_{m.BAD}(a)}{F_{N.ALL}(a)}, \quad a \in [L, H]$$

- In practice, Lift is computed corresponding to 10%, 20%, . . . , 100% of clients with the worst score. Hence we define:

$$QLift(q) = \frac{F_{m.BAD}(F^{-1}_{N.ALL}(q))}{F_{N.ALL}(F^{-1}_{N.ALL}(q))} = \frac{1}{q} F_{m.BAD}(F^{-1}_{N.ALL}(q)), \quad q \in (0,1]$$

$$F^{-1}_{N.ALL}(q) = \min\{a \in [L, H], F_{N.ALL}(a) \geq q\}$$

- Typical value of q is 0.1. Then we have

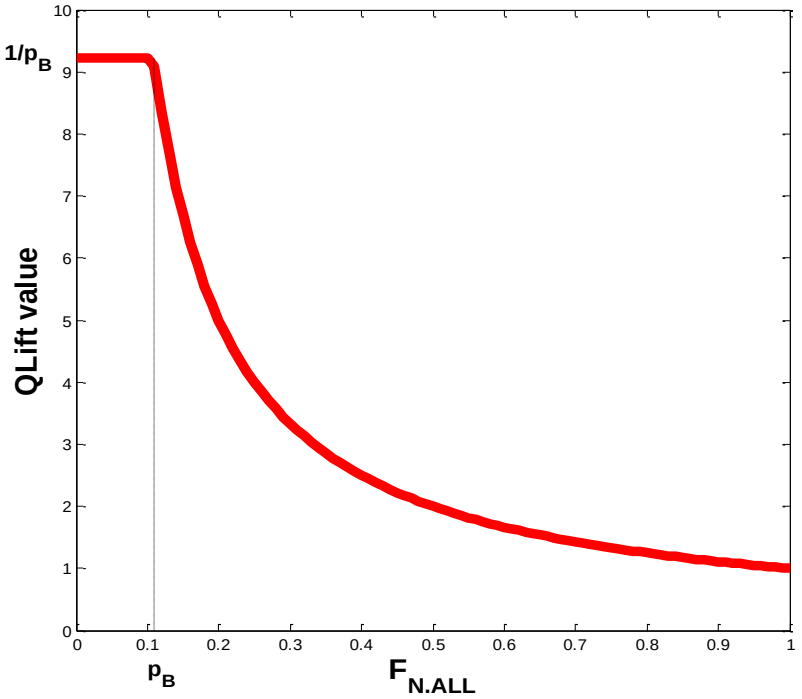
$$QLift_{10\%} = QLift(0.1) = 10 \cdot F_{m.BAD}(F^{-1}_{N.ALL}(0.1))$$

Lift and QLift for ideal model

- It is natural to ask how look Lift and QLift in case of ideal model. Hence we derived following formulas.

- Lift for ideal model:

$$Lift_{ideal}(a) = \begin{cases} \frac{1}{p_B}, & a \leq c \\ \frac{1}{F_{N.ALL}(a)}, & a > c \end{cases}$$



- QLift for ideal model:

$$QLift_{ideal}(q) = \begin{cases} \frac{1}{p_B}, & q \in (0, p_B] \\ \frac{1}{q}, & q \in (p_B, 1] \end{cases}$$

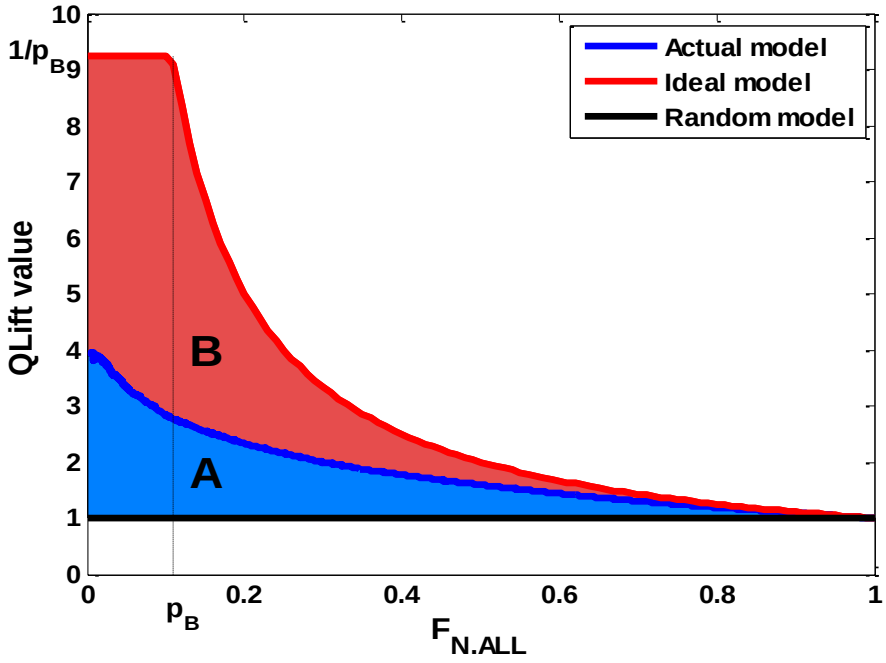
We can see that the upper limit of Lift and QLift is equal to $1/p_B$.

Lift Ratio (LR)

- Once we know form of QLift for ideal model, we can define Lift Ratio as analogy to Gini index.

$$LR = \frac{A}{A+B} = \frac{\int_0^1 QLift(q)\,dq - 1}{\int_0^1 QLift_{ideal}(q)\,dq - 1}$$

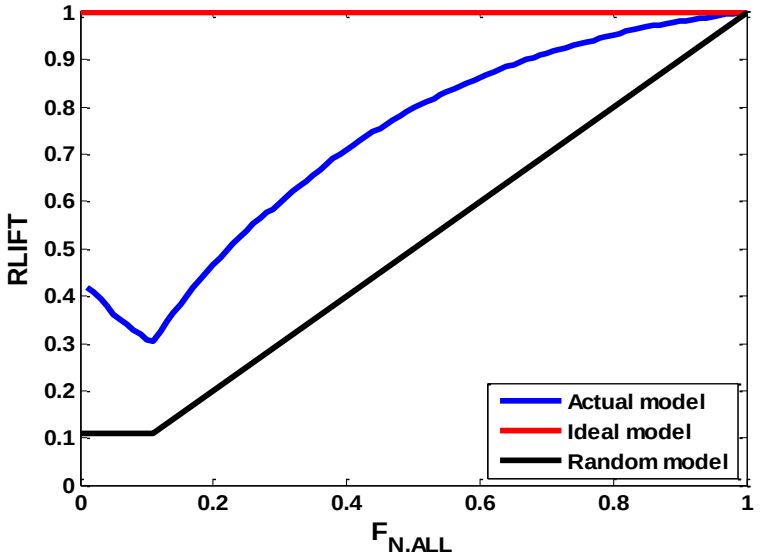
- It is obvious that it is global measure of model's quality and that it takes values from 0 to 1. Value 0 corresponds to random model, value 1 match to ideal model. Meaning of this index is quite simple. The higher, the better. Important feature is that Lift Ratio allows us to fairly compare two models developed on different data samples, which is not possible with Lift.



Rlift, IRL

- Since Lift Ratio compares areas under Lift function for actual and ideal models, next concept is focused on comparison of Lift functions themselves. We define Relative Lift function by

$$RLift(q) = \frac{QLift(q)}{QLift_{ideal}(q)}, \quad q \in (0, 1]$$



- In connection to RLift we define Integrated Relative Lift (IRL):

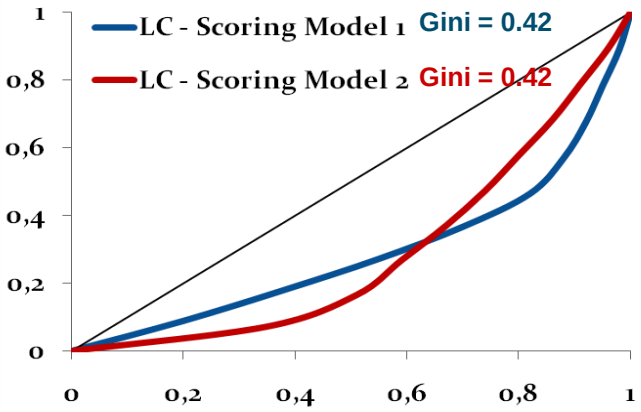
$$IRL = \int_0^1 RLift(q)\, dq$$

- It takes values from $0.5 + \frac{p_B^2}{2}$, for random model, to 1, for ideal model. Following simulation study shows interesting connection to c-statistics.

Příklad

- We consider two scoring models with score distribution given in the table below.
- We consider standard meaning of scores, i.e. higher score band means better clients (the highest probability of default have clients with the lowest scores, i.e. clients in score band 1).
- Gini indexes are equal for both models.
- It is evident from the Lorenz curves, that the first model is stronger for higher score bands and the second one is better for lower score bands.
- The same we can read from values of QLift.

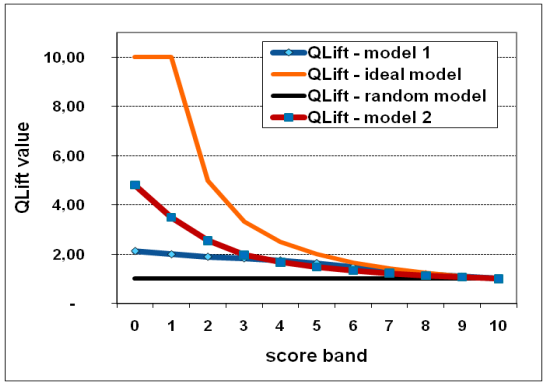
			Scoring Model 1				Scoring Model 2			
score band	# clients	q	# bad clients	# cumul. bad clients	# cumul. bad rate	QLift	# bad clients	# cumul. bad clients	# cumul. bad rate	QLift
1	100	0.1	20	20	20.0%	**2.00**	35	35	35.0%	**3.50**
2	100	0.2	18	38	19.0%	**1.90**	16	51	25.5%	**2.55**
3	100	0.3	17	55	18.3%	**1.83**	8	59	19.7%	**1.97**
4	100	0.4	15	70	17.5%	**1.75**	8	67	16.8%	**1.68**
5	100	0.5	12	82	16.4%	**1.64**	7	74	14.8%	**1.48**
6	100	0.6	6	88	14.7%	**1.47**	6	80	13.3%	**1.33**
7	100	0.7	4	92	13.1%	**1.31**	6	86	12.3%	**1.23**
8	100	0.8	3	95	11.9%	**1.19**	5	91	11.4%	**1.14**
9	100	0.9	3	98	10.9%	**1.09**	5	96	10.7%	**1.07**
10	100	1.0	2	100	10.0%	**1.00**	4	100	10.0%	**1.00**
All	1000		100				100			

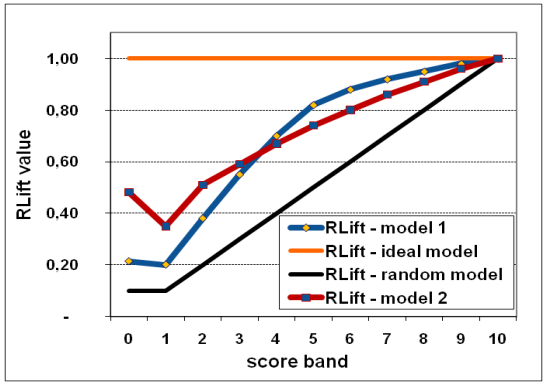


Příklad

- Since Qlift is not defined for q=0, we extrapolated the value by

$$QLift(0) = 3 \cdot QLift(0.1) - 3 \cdot QLift(0.2) + QLift(0.3)$$

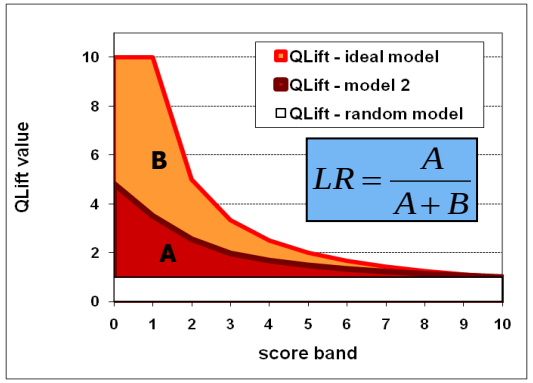


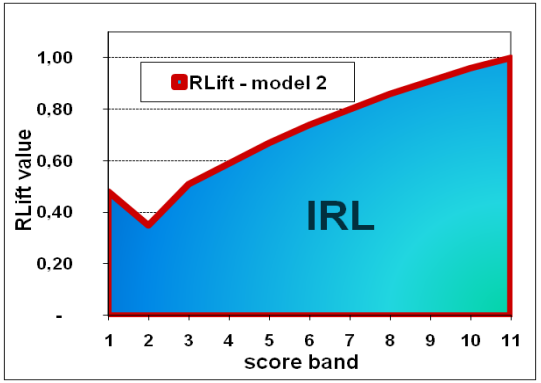


According to both Qlift and Rlift curves we can state that:

- If expected reject rate is up to 40%, then model 2 is better.
- If expected reject rate is more than 40%, then model 1 is better.

- Now, we consider indexes LR and IRL:





	scoring model 1	scoring model 2
GINI	**0.420**	**0.420**
QLift(0.1)	**2.000**	**3.500**
LR	**0.242**	**0.372**
IRL	**0.699**	**0.713**

Using LR and IRL we can state that model 2 is better than model 1 although their Gini coefficients are equal.

Indexy založené na hustotě

- Střední diference (Mahalanobis distance):

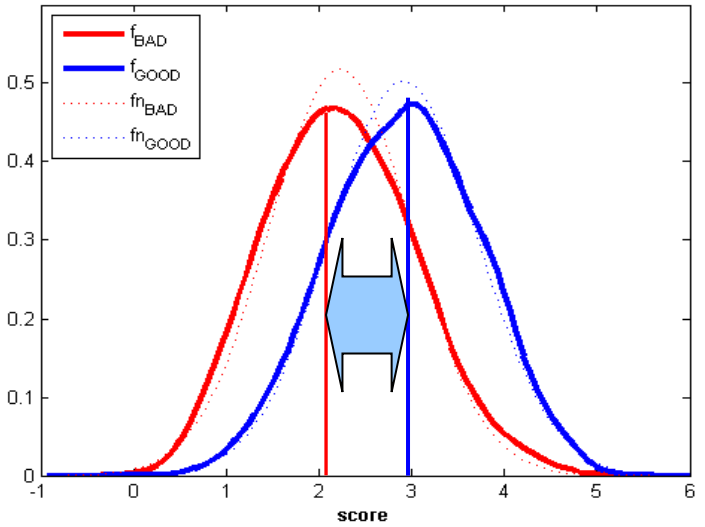
$$D = \frac{M_g - M_b}{S}$$

kde S je společná směrodatná odchylka:

$$S = \left(\frac{nS_g^{\ 2} + mS_b^{\ 2}}{n+m} \right)^{\frac{1}{2}}$$

M_g , M_b jsou střední hodnoty dobrých (špatných) klientů

S_g , S_b jsou příslušné směrodatné odchylky.



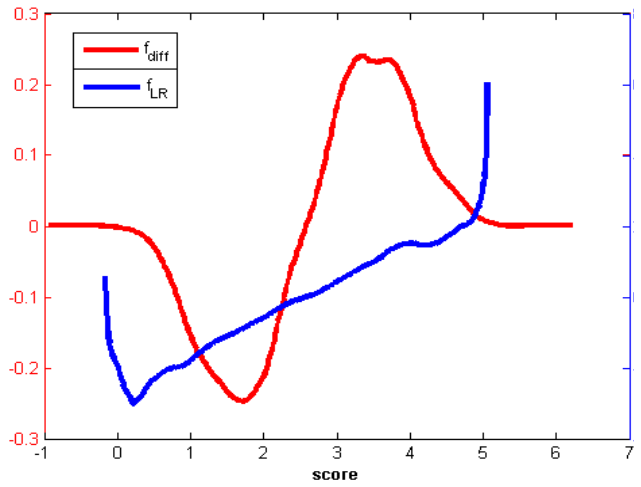
Indexy založené na hustotě

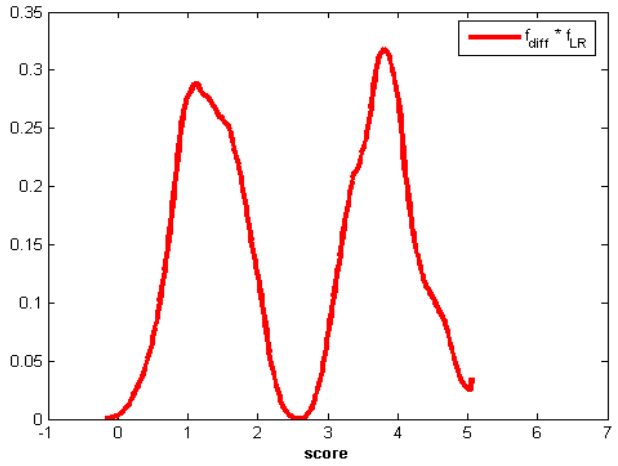
- Informační hodnota (I_{val}) – spojitý případ (Divergence):
  - jde o symetrizovanou Kullback-Leiblerovu divergenci známou také pod názvem J-divergence.

$$I_{val} = \int_{-\infty}^{\infty} (f_{GOOD}(x) - f_{BAD}(x)) \ln\left(\frac{f_{GOOD}(x)}{f_{BAD}(x)}\right) dx$$

$$f_{diff}(x) = f_{GOOD}(x) - f_{BAD}(x)$$

$$f_{LR}(x) = \ln\left(\frac{f_{GOOD}(x)}{f_{BAD}(x)}\right)$$





Indexy založené na hustotě

- Informační statistika/hodnota (I_{val}) – diskrétní případ:

  - Vytvoříme intervaly skóre – typicky decily. Počet dobrých (špatných) klientů v i-tém intervalu označíme g_i (b_i).

  - Musí platit $g_i > 0, b_i > 0 \quad \forall i$

  - Potom dostáváme

$$I_{val} = \sum_i \left(\frac{g_i}{n} - \frac{b_i}{m} \right) \ln \left(\frac{g_i m}{b_i n} \right)$$

score int.	# bad clients	#good clients	% bad [1]	% good [2]	[3] = [2] - [1]	[4] = [2] / [1]	[5] = ln[4]	[6] = [3] * [5]
1	1	10	2,0%	1,1%	-0,01	0,53	-0,64	0,01
2	2	15	4,0%	1,6%	-0,02	0,39	-0,93	0,02
3	8	52	16,0%	5,5%	-0,11	0,34	-1,07	0,11
4	14	93	28,0%	9,8%	-0,18	0,35	-1,05	0,19
5	10	146	20,0%	15,4%	-0,05	0,77	-0,26	0,01
6	6	247	12,0%	26,0%	0,14	2,17	0,77	0,11
7	4	137	8,0%	14,4%	0,06	1,80	0,59	0,04
8	3	105	6,0%	11,1%	0,05	1,84	0,61	0,03
9	1	97	2,0%	10,2%	0,08	5,11	1,63	0,13
10	1	48	2,0%	5,1%	0,03	2,53	0,93	0,03
All	50	950					Info. Value	0,68

Normálně rozložené skóre

- Předpokládejme, že skóre dobrých a špatných klientů je normálně rozloženo, tj. jejich pravděpodobnostní hustoty mají tvar

$$f_{GOOD}(x) = \frac{1}{\sigma_g \sqrt{2\pi}} e^{-\frac{(x-\mu_g)^2}{2\sigma_g^2}} \qquad f_{BAD}(x) = \frac{1}{\sigma_b \sqrt{2\pi}} e^{-\frac{(x-\mu_b)^2}{2\sigma_b^2}}$$

- Odhady parametrů μ_b, μ_b, σ_g a σ_b :

  M_g, M_b jsou aritmetické průměty skóre dobrých (špatných) klientů

  S_g , S_b jsou směrodatné odchylky skóre dobrých (špatných) klientů

- Společná směrodatná odchylka:

$$S = \left(\frac{nS_g^{\,2} + mS_b^{\,2}}{n+m} \right)^{\frac{1}{2}}$$

- Odhady střední hodnoty a směrodatné odchylky skóre všech klientů μ_{ALL}, σ_{ALL}:

$$M = M_{ALL} = \frac{nM_g + mM_b}{n+m} \qquad S_{ALL} = \left(\frac{nS_g^{\,2} + mS_b^{\,2} + n(M_g - M)^2 + m(M_b - M)^2}{(n+m)} \right)^{\frac{1}{2}}$$

Normálně rozložené skóre

- Předpokládejme, že směrodatné odchylky obou skóre jsou rovny hodnotě σ, pak:

$$D = \frac{\mu_g - \mu_b}{\sigma}$$

$$D = \frac{M_g - M_b}{S}$$

$$KS = \Phi\left(\frac{D}{2}\right) - \Phi\left(\frac{-D}{2}\right) = 2 \cdot \Phi\left(\frac{D}{2}\right) - 1$$

$$Gini = 2 \cdot \Phi\left(\frac{D}{\sqrt{2}}\right) - 1$$

$$QLift(q) = \frac{1}{q}\Phi\left(\frac{\sigma_{ALL}}{\sigma} \cdot \Phi^{-1}(q) + p_G \cdot D\right)$$

$$QLift(q) = \frac{1}{q}\Phi\left(\frac{S_{ALL}}{S}\Phi^{-1}(q) + p_G \cdot D\right)$$

$$I_{val} = D^2$$

Kde $\Phi(\cdot)$ je distribuční funkce standardizovaného normálního rozložení, $\Phi_{\mu,\sigma^2}(\cdot)$ je distribuční funkce s parametry μ, σ^2 a $\Phi^{-1}(\cdot)$ je standardizovaná kvantilová funkce.

Normálně rozložené skóre

- Obecně, tj. bez předpokladu rovnosti směrodatných odchylek skóre:

$$D^* = \frac{\mu_g - \mu_b}{\sqrt{\sigma_g^2 + \sigma_b^2}}$$

$$D^* = \frac{M_g - M_b}{\sqrt{S_g^2 + S_b^2}}$$

$$KS = \Phi\left(\frac{a}{b}\sigma_b \cdot D^* - \frac{1}{b}\sigma_g\sqrt{a^2 D^{*2} + 2b \cdot c}\right) - \Phi\left(\frac{a}{b}\sigma_g \cdot D^* - \frac{1}{b}\sigma_b\sqrt{a^2 D^{*2} + 2b \cdot c}\right)$$

kde $a = \sqrt{\sigma_b^2 + \sigma_g^2}, \quad b = \sigma_b^2 - \sigma_g^2, \quad c = \ln\left(\frac{\sigma_g}{\sigma_b}\right)$

$$KS = \Phi\left(\frac{\sqrt{S_b^2 + S_g^2}}{S_b^2 - S_g^2} S_b \cdot D^* - \frac{1}{S_b^2 - S_g^2} S_g \sqrt{(S_b^2 + S_g^2)D^{*2} + 2 \cdot (S_b^2 - S_g^2)\ln\left(\frac{S_g}{S_b}\right)}\right)$$

$$-\Phi\left(\frac{\sqrt{S_b^2 + S_g^2}}{S_b^2 - S_g^2} S_g \cdot D^* - \frac{1}{S_b^2 - S_g^2} S_b \sqrt{(S_b^2 + S_g^2)D^{*2} + 2 \cdot (S_b^2 - S_g^2)\ln\left(\frac{S_g}{S_b}\right)}\right)$$

Normálně rozložené skóre

- Obecně, tj. bez předpokladu rovnosti směrodatných odchylek skóre:

$$Gini = 2 \cdot \Phi(D^*) - 1$$

$$Lift_q = \frac{1}{q} \Phi_{\mu_b, \sigma_b^2}(\mu_{ALL} + \sigma_{ALL} \cdot \Phi^{-1}(q)) = \frac{1}{q} \Phi\left(\frac{\sigma_{ALL} \cdot \Phi^{-1}(q) + \mu_{ALL} - \mu_b}{\sigma_b} \right)$$

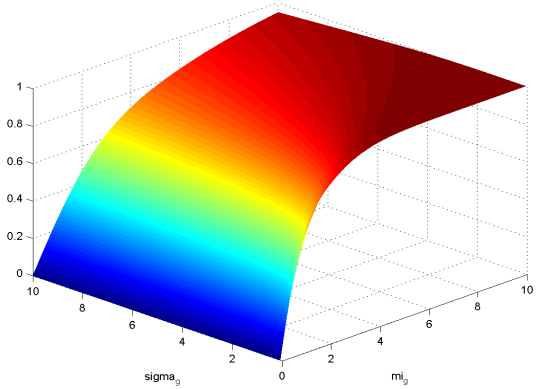
$$QLift(q) = \frac{1}{q} \Phi\left(\frac{S_{ALL} \cdot \Phi^{-1}(q) + M - M_b}{S_b} \right)$$

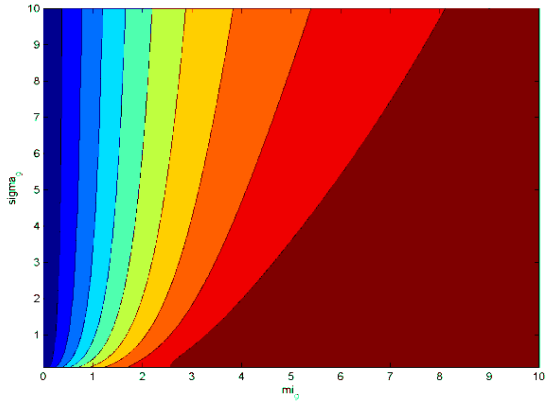
$$I_{val} = (A+1)D^{*2} + A - 1, \quad A = \frac{1}{2}\left(\frac{\sigma_b^2}{\sigma_g^2} + \frac{\sigma_g^2}{\sigma_b^2} \right)$$

$$I_{val} = (A+1)D^{*2} + A - 1, \quad A = \frac{1}{2}\left(\frac{S_b^2}{S_g^2} + \frac{S_g^2}{S_b^2} \right)$$

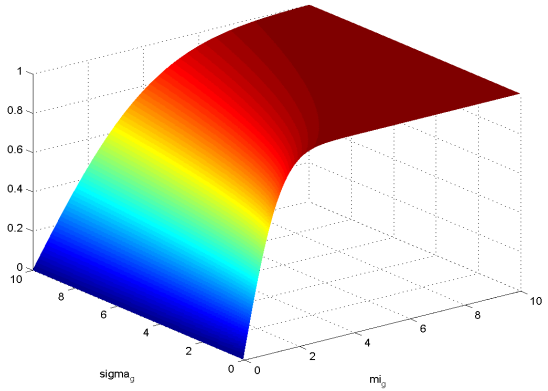
Normálně rozložené skóre

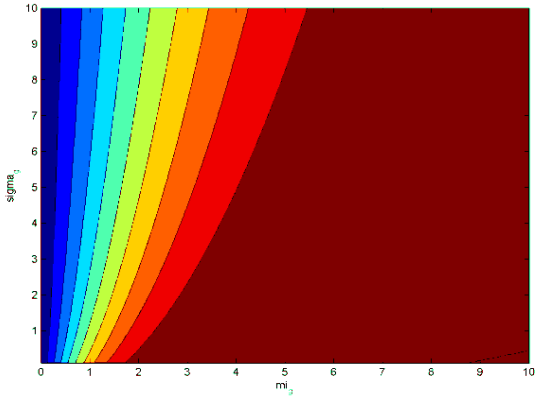
- **KS:** $\mu_b = 0,\ \sigma_b^2 = 1$





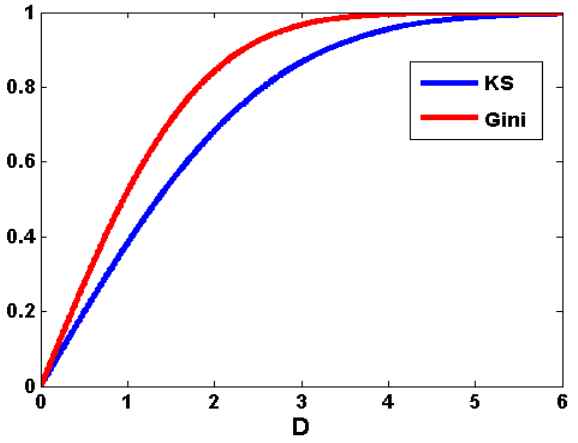
- **Gini** $\mu_b = 0,\ \sigma_b^2 = 1$





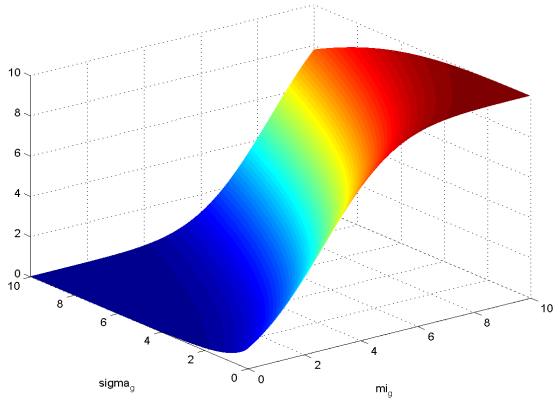
- KS i Gini reagují velmi silně na změnu μ_g, ale zůstávají téměř beze změny ve směru σ_g^2.

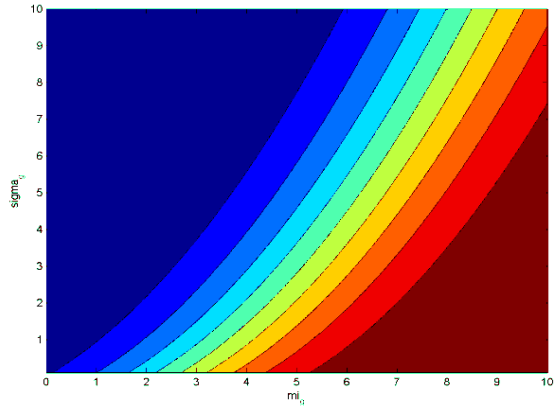
- **Gini > KS**



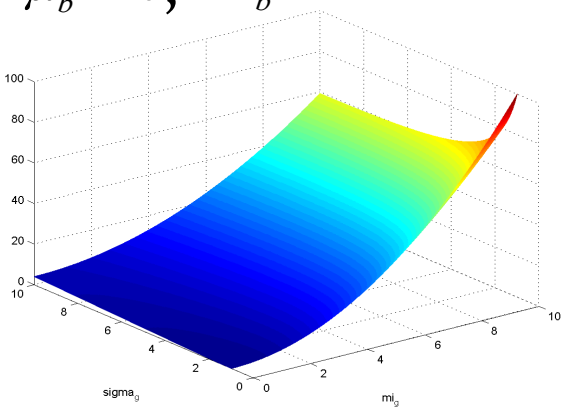
Normálně rozložené skóre

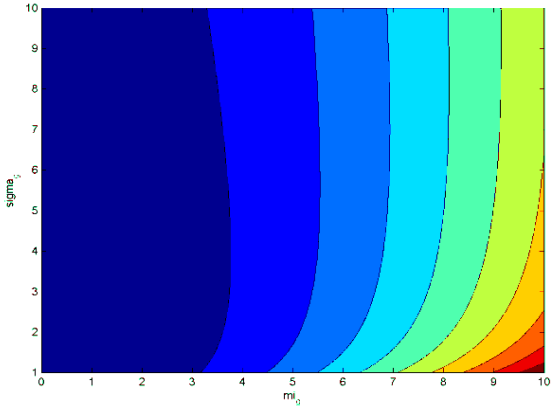
- **$Lift_{10\%}$:** $\mu_b = 0, \quad \sigma_b^2 = 1$





- **I_{val}:** $\mu_b = 0, \quad \sigma_b^2 = 1$



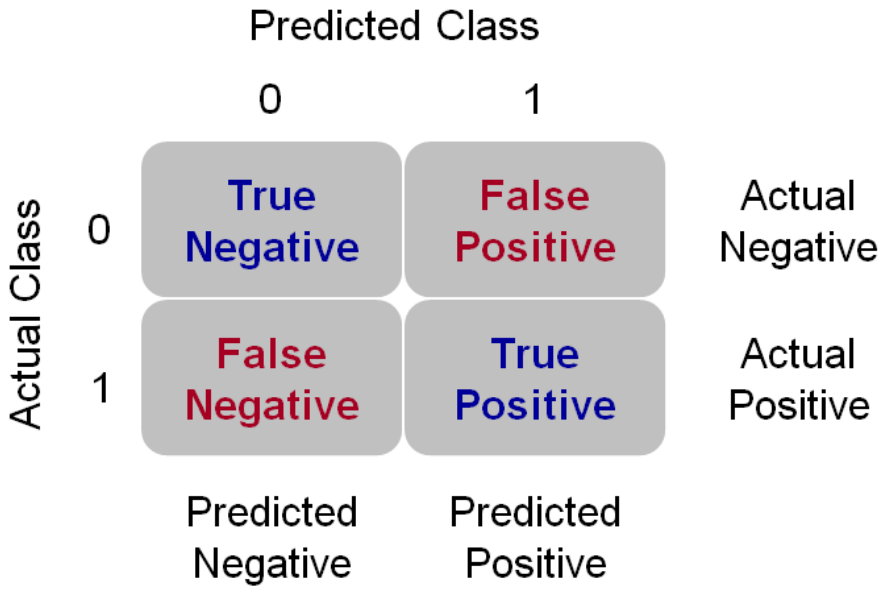


- V případě indexu $Lift_{10\%}$ je evidentní silná závislost na μ_g a významně vyšší závislost na σ_g^2 než v případě KS a Gini.

- Opět velmi silná závislost na μ_g. Navíc hodnota I_{val} míří velmi rychle k nekonečnu pokud se σ_g^2 blíží nule.

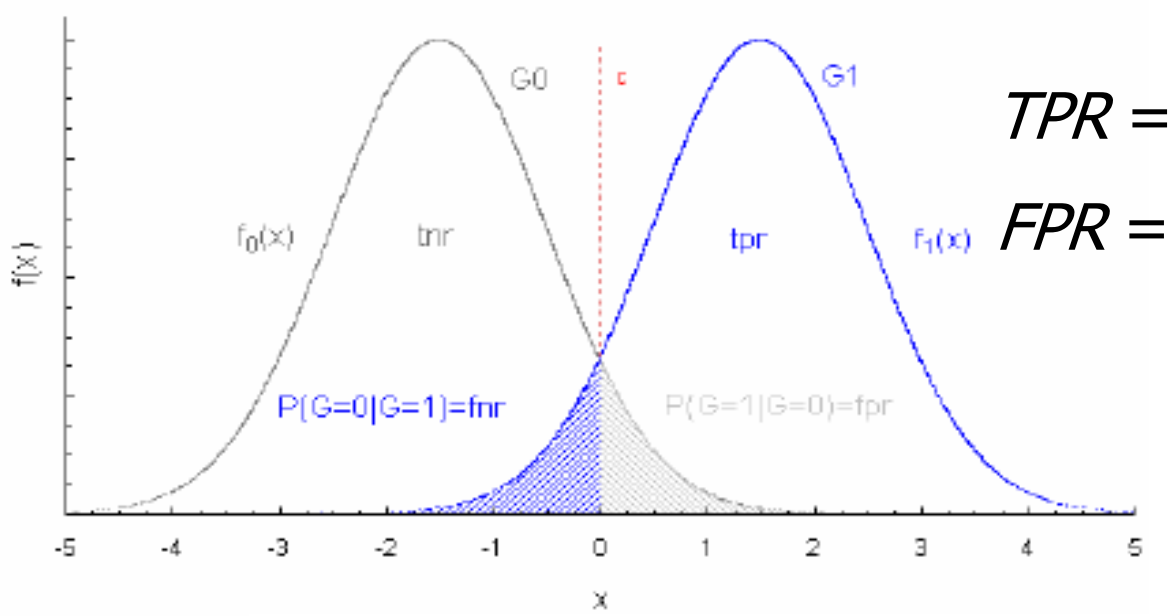
ROC (Receiver operating characteristic)

- *TN (true negative)* - počet správně klasifikovaných negativních případů
- *TP (true positive)* – počet správně klasifikovaných positivních případů
- *FP (false positive)* – počet nesprávně klasifikovaných negativních případů
- *FN (false negative)* – počet nesprávně klasifikovaných positivních případů



	Predikce		
Skuteč.	G0	G1	Celkem
G0	*TN*	*FP*	*N*
G1	*FN*	*TP*	*P*
Celkem	*Pneg*	*PPos*	*n*

ROC –TPR, FPR



$$TPR = TP / P = TP / (TP + FN)$$

$$FPR = FP / N = FP / (FP + TN)$$

$$tpr(c) = P(X > c \mid G_1) = 1 - F_1(c)$$

$$tnr(c) = P(X < c \mid G_0) = F_0(c)$$

$$fpr(c) = P(X > c \mid G_0) = 1 - F_0(c)$$

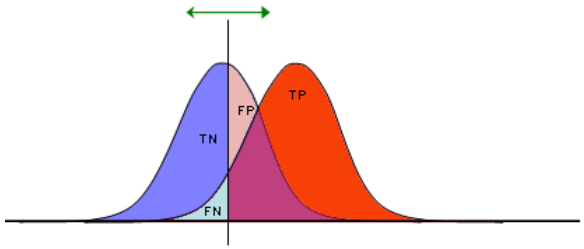
$$fnr(c) = P(X < c \mid G_1) = F_1(c)$$

		Condition (as determined by "Gold standard")		
		Condition Positive	Condition Negative	
Test Outcome	Test Outcome Positive	True Positive	False Positive (Type I error)	Positive predictive value = Σ True Positive / Σ Test Outcome Positive
	Test Outcome Negative	False Negative (Type II error)	True Negative	Negative predictive value = Σ True Negative / Σ Test Outcome Negative
		Sensitivity = Σ True Positive / Σ Condition Positive	Specificity = Σ True Negative / Σ Condition Negative	

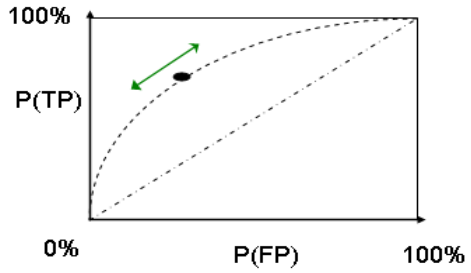
ROC curve

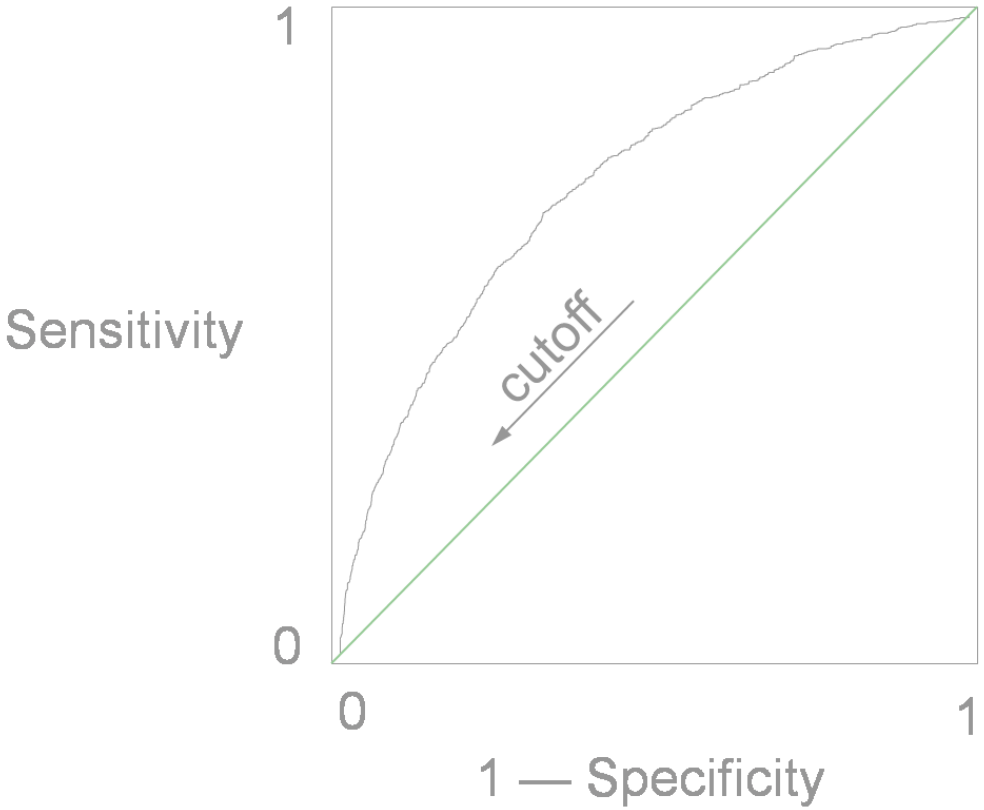
$$x = FPR = 1 - Specificity$$

$$y = TPR = Sensitivity$$



TP	FP
FN	TN
1	1

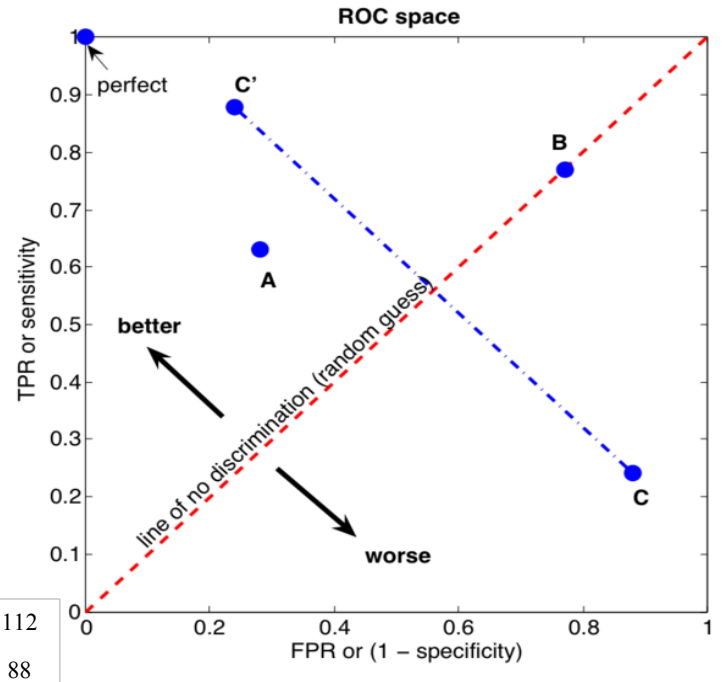




ROC - ACC

Accuracy:

$ACC = (TP + TN) / (P + N)$



A			B			C			C'		
TP=63	FP=28	91	TP=77	FP=77	154	TP=24	FP=88	112	TP=88	FP=24	112
FN=37	TN=72	109	FN=23	TN=23	46	FN=76	TN=12	88	FN=12	TN=76	88
100	100	200	100	100	200	100	100	200	100	100	200
TPR = 0.63			TPR = 0.77			TPR = 0.24			TPR = 0.88		
FPR = 0.28			FPR = 0.77			FPR = 0.88			FPR = 0.24		
ACC = 0.68			ACC = 0.50			ACC = 0.18			ACC = 0.82		

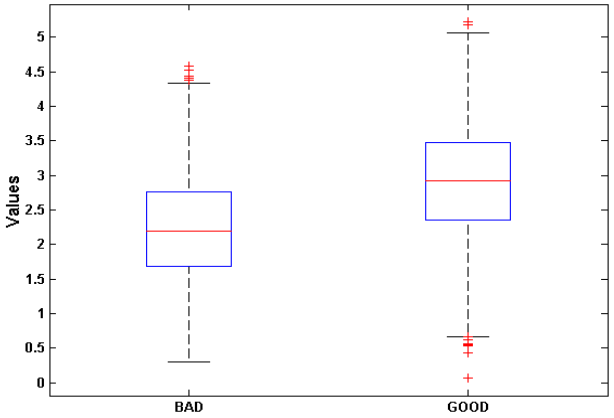
ROC – AUC, Gini

- AUC (area under curve, neboli plocha pod ROC křivkou) je rovna pravděpodobnosti, že daný model ohodnotí náhodně vybraného dobrého klienta vyšším skóre než náhodně vybraného špatného klienta. Dá se ukázat, že plocha pod ROC křivkou se dá vyjádřit pomocí Mann-Whitneymu U, které testuje rozdíl mediánů mezi dvěma skupinami spojitých skóre. AUC se dá vyjádřit i pomocí Giniho koeficientu pomocí vzorce

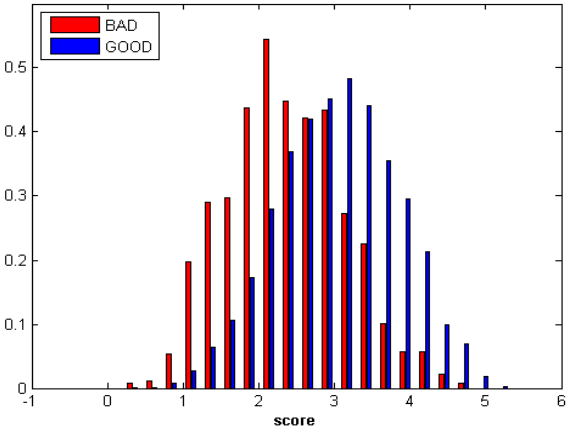
$$Gini + 1 = 2xAUC$$

Další evaluační grafy

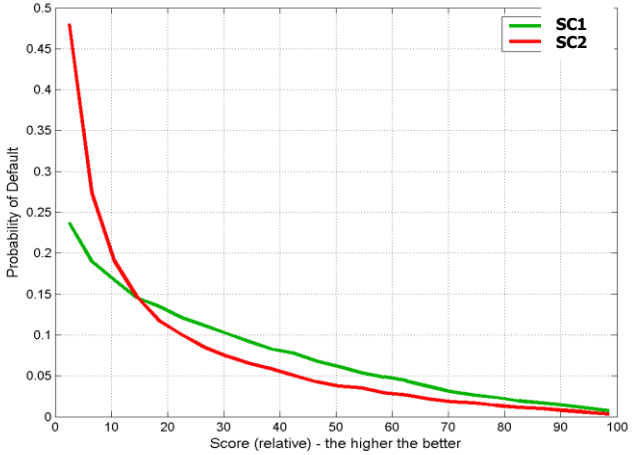
Boxplot



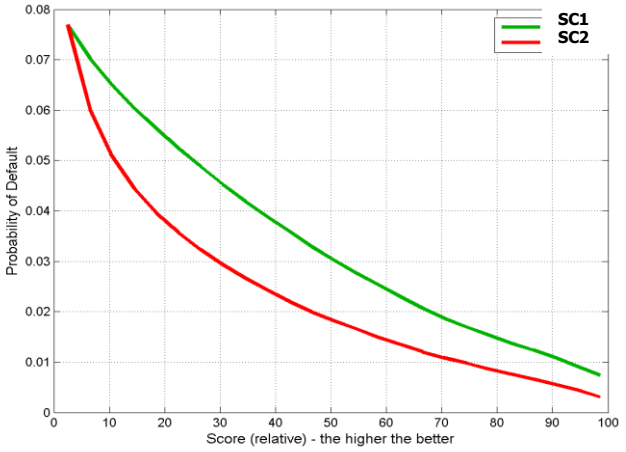
Histogram



PD - absolutně



PD - kumulativně



Postupy evaluace

- **evaluace na učících datech**

Evaluace na učících datech použitých k učícímu procesu není ke zjištění kvality modelu vhodná a má nízkou vypovídací schopnost, protože často může dojít k přeučení modelu. Odhad predikční kvality modelu na učících datech se nazývá resubstituční nebo interní odhad. Odhady ukazatelů kvality modelů provedených na učících datech jsou nadhodnocené, proto se místo nich používají testovací data, která se v rámci přípravy dat pro tyto účely vyčlení.

Postupy evaluace

- **evaluace na testovacích datech**

Evaluace na testovacích datech již má patřičnou vypovídací schopnost, jelikož tato data nebyla použita k sestavení modelu. Na testovací data jsou kladeny určité požadavky. Soubor testovacích dat by měl obsahovat dostatečné množství dat a měl by reprezentovat či vystihovat charakteristiky učících dat. Empiricky doporučený poměr učících a testovacích dat je 75%, resp. 25% případů. Zajištění patřičné reprezentativnosti je realizováno pomocí náhodného stratifikovaného výběru.

Postupy evaluace

➢ **křížové ověřování** *(cross-validation)*

V případě nedostatečného počtu pozorování, kdy rozdělení datového souboru na učící a testovací data za účelem vyhodnocení modelu není možné, je vhodné použít metodu křížového ověřování. Výhodou této metody na rozdíl od dělení datového souboru je, že každý případ z dat je použit k sestavení modelu a každý případ je alespoň jednou použit k testování. Postup je následující:

- Soubor dat je náhodně rozdělen do *n* disjunktních podmnožin tak, že každá podmnožina obsahuje přibližně stejný počet záznamů. Výběry jsou stratifikovány podle tříd (příslušnosti k určité třídě), aby bylo zajištěno, že podíly jednotlivých tříd podmnožin jsou zhruba stejné jako v celém souboru.
- Z těchto *n* disjunktních podmnožin se vyčlení *n-1* podmnožin pro sestavení modelu (konstrukční podmnožina) a zbývající podmnožina (validační podmnožina) je použita k jeho vyhodnocení. Model je tedy evaluován na podmnožině dat, ze kterých nebyl sestaven a na této množině dat je odhadována jeho predikční kvalita.
- Celý postup se zopakuje *n*-krát a dílčí odhady ukazatelů kvality se zprůměrňují. Velikost validační podmnožiny lze přibližně stanovit jako poměr počtu případů ku počtu validačních podmnožin.

Postupy evaluace

- ***bootstrap* metoda**

Metoda *bootstrap* zkoumá charakteristiky jednotlivých resamplovaných vzorků, které byly pořízeny z empirického výběru. Pokud původní výběr osahuje *m* prvků, tak každý má naději objevit se v resamplovaném výběru. Při úplném resamplování o velikosti vzorku *n* jsou uvažovány všechny možné výběry a existuje tedy m^n možných výběrů. Úplné resamplování je teoreticky proveditelné, ale vyžádalo by si mnoho času. Alternativou je simulace *Monte Carlo*, pomocí níž se aproximuje úplné resamplování tak, že se provede *B* náhodných výběrů (obvykle se volí 500 – 10000 výběrů) s tím, že každý prvek je vždy nahrazen (vrácen zpět do osudí). Jsou-li dána data $X=\{X1, \ldots, Xn\}$ a je-li požadován odhad parametru θ, provede se z původních dat *B* výběrů a pro každý výběr je spočítán odhad parametru θ . *Bootstrap* odhad parametru je určen jako průměr dílčích odhadů. V případě evaluace modelů bude parametrem θ zvolený ukazatel predikční kvality.

- ***jackknife***

Tato metoda je založena na sekvenční strategii odebírání a vracení prvků do výběru o velikosti *n*. Pro datový soubor, který obsahuje *n* prvků, procedura generuje *n* vzorků s počtem prvků *n-1*. Pro každý zmenšený výběr o velikosti *n*-1 je odhadnuta hodnota parametru. Dílčí odhady se následně zprůměrují podobně jako u metody *bootstrap*.

Bootstrap

- Základní postup metody (*n* pozorování):
  - Generování *k* (obvykle 20 a více) nezávislých výběrů *n* prvků **s vracením** z původních dat
    - Tyto výběry z výběru *n* pozorování jsou vlastně přibližnou náhradou za nezávislé výběry z celé populace = idea metody
  - Na každém z *k* výběrů odhadneme model stejně jako na původním základním výběru
  - Populace *k* různých výsledků nám umožní odhadovat stabilitu odhadovaných parametrů
  - Zobecnitelnost lze odhadovat s použitím prvků, které v daném kole nebyly vybrány – OOB odhady (=out-of-bag), – tj. užít je jako testovací množiny
    - Těchto OOB prvků je v průměru 36,8 % z počtu pozorování

Portrét: Bradley Efron, který tuto verzi metody Monte Carlo r. 1979 publikoval

12. Úvod do makro jazyka v SAS.

```
%MACRO macro-name(keyword=value, …, keyword=value);
    macro text
%MEND <macro-name>;
```

Purpose of the Macro Facility

- The *macro facility* is a text processing facility for automating and customizing SAS code. The macro facility helps minimize the amount of SAS code you must type to perform common tasks.

- The macro facility supports the following:
  - symbolic substitution within SAS code
  - automated production of SAS code
  - dynamic generation of SAS code
  - conditional construction of SAS code

Purpose of the Macro Facility

- The macro facility enables you to do the following:
  - create and resolve **macro variables** anywhere within a SAS program
  - write and call **macro programs** (**macro definitions** or **macros**) that generate custom SAS code

Substituting System Values

- Example: Include system values within SAS footnotes.

```
proc print data=orion.customer;
title "Customer List";
footnote1 "Created 10:24 Monday, 31MAR2008";
footnote2 "on the WIN System Using SAS 9.2";
run;
```

Automatic macro variables store system values that can be used to avoid hardcoding.

Substituting User-Defined Values

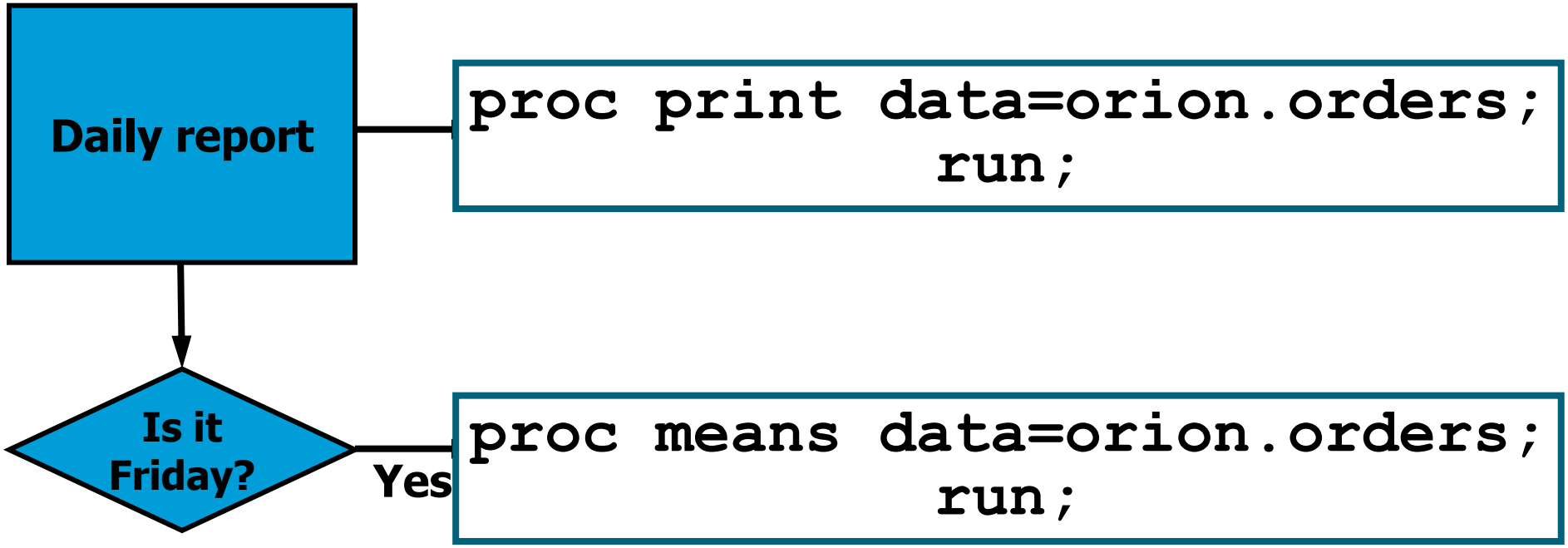
- Example: Reference the same value repeatedly throughout a program.

```
proc freq data=orion.order_fact;
   where year(order_date)=2008;
   table order_type;
   title "Order Types for 2008";
run;
proc means data=orion.order_fact;
   where year(order_date)=2008;
   class order_type;
   var Total_Retail_Price;
   title "Price Statistics for 2008";
run;
```

User-defined macro variables enable you to define a value once and substitute that value repeatedly within a program.

Conditional Processing

- Example: Generate a detailed report on a daily basis. Generate an additional report every Friday, summarizing data on a weekly basis.



A *macro program* can **conditionally** execute selected portions of a SAS program based on user-defined conditions.

Repetitive Processing

- Example: Generate a similar report each year from 2008 to 2010.

```
proc print data=orion.year2008;
            run;
```

```
proc print data=orion.year2009;
            run;
```

```
proc print data=orion.year2010;
            run;
```

A macro program can **generate SAS code repetitively**, substituting different values with each iteration.

Data-Driven Applications

```
data AU CA DE IL TR US ZA;
   set orion.customer;
   select(country);
      when("AU") output AU;
      when("CA") output CA;
      when("DE") output DE;
      when("IL") output IL;
      when("TR") output TR;
      when("US") output US;
      when("ZA") output ZA;
      otherwise;
   end;
run;
```

Example: Create separate subsets of a selected data set for each unique value of a selected variable.

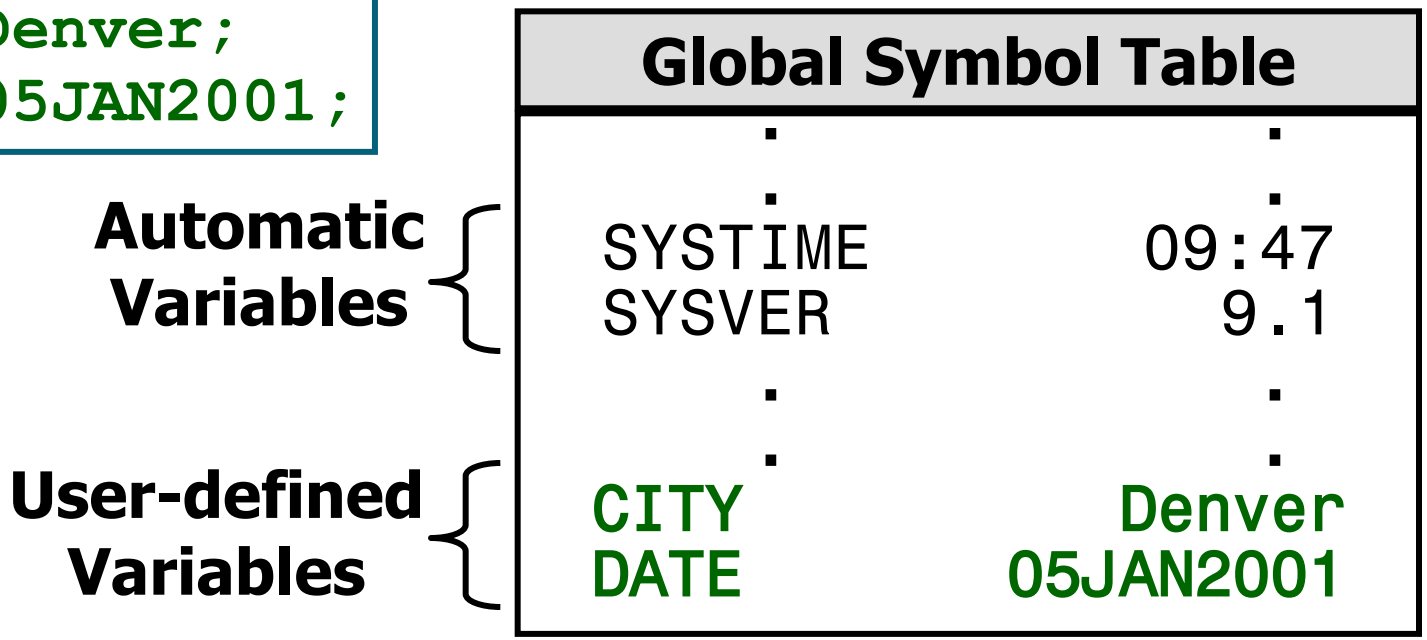
A macro program can **generate data-driven code**.

Global Macro Variables

- When SAS is invoked, a global symbol table is created and initialized with automatic or system-defined macro variables.

You can also create *user-defined* global macro variables with the %LET statement:

```
%let city=Denver;
%let date=05JAN2001;
```



Referencing a Macro Variable

- To substitute the value of a macro variable in your program, you must reference it.

A macro variable reference

- is made by preceding the macro variable name with an ampersand (&)
- causes the macro processor to search for the named macro variable and return its value from the symbol table if the variable exists.

Referencing a Macro Variable

Global Symbol Table	
CITY	Denver
DATE	05JAN2001

Example: Use a macro variable reference to make a substitution in a SAS program statement.

```
where OrderCity="&city";
```

generates

```
WHERE ORDERCITY="Denver";
```

Referencing a Macro Variable

- When the macro processor cannot act upon a macro variable reference, a message is printed in the SAS log.

Global Symbol Table	
CITY	Denver
DATE	05JAN2001

Referencing a nonexistent macro variable results in a warning message.

```
title "Orders from &cityst";
```

generates

```
WARNING: Apparent symbolic reference CITYST not resolved.
```

Referencing a Macro Variable

Global Symbol Table	
CITY	Denver
DATE	05JAN2001

Referencing an invalid macro variable name results in an error message.

```
title "Orders from &THE_CITY_IN_WHICH_THE_ORDER_ORIGINATED";
```

generates

```
ERROR: Symbolic variable name
       THE_CITY_IN_WHICH_THE_ORDER_ORIGINATED
       must be 32 or fewer characters long.
```

Displaying Macro Variable Values

- Use the SYMBOLGEN system option to monitor the value that is substituted for a macro variable referenced.

General form of the SYMBOLGEN system option:

```
OPTIONS SYMBOLGEN;
```

This system option displays the results of resolving macro variable references in the SAS log.

The default option setting is NOSYMBOLGEN.

Displaying Macro Variable Values

Global Symbol Table	
CITY	Denver
DATE	05JAN2001

Partial SAS Log

```
where CustomerAddress2 contains "&city";
SYMBOLGEN: Macro variable CITY resolves to Denver
where CustomerAddress2 contains '&city';
```

Why is no message displayed for the final example?

Displaying Macro Variable Values

- To verify the values of macro variables, you may want to write your own messages to the SAS log. The %PUT statement writes text to the SAS log.

General form of the %PUT statement:

%PUT *text* ;

Displaying Macro Variable Values

- The %PUT statement
  - writes to the SAS log only
  - always writes to a new log line starting in column one
  - writes a blank line if *text* is not specified
  - does not require quotation marks around *text*
  - resolves references to macro variables in *text* before *text* is written
  - removes leading and trailing blanks from *text* unless a macro quoting function is used
  - wraps lines when the length of *text* is greater than the current line size setting
  - can be used inside or outside a macro definition.

Displaying Macro Variable Values

- Example: Write a message to the SAS log to verify the value of the macro variable **CITY**.

Global Symbol Table	
CITY	Denver
DATE	05JAN2001

Partial SAS Log

```
%put The value of the macro variable CITY is: &city;
The value of the macro variable CITY is: Denver
```

System-Defined Automatic Macro Variables

- Some automatic macro variables have fixed values that are set at SAS invocation:

Name	Value
SYSDATE	date of SAS invocation (DATE7.)
SYSDATE9	date of SAS invocation (DATE9.)
SYSDAY	day of the week of SAS invocation
SYSTIME	time of SAS invocation
SYSENV	FORE (interactive execution) BACK (noninteractive or batch execution)
SYSSCP	abbreviation for the operating system used such as OpenVMS, WIN, HP 300
SYSVER	release of SAS software being used
SYSJOBID	identifier of current SAS session or batch job mainframe systems: the userid or job name other systems: the process ID (PID)

System-Defined Automatic Macro Variables

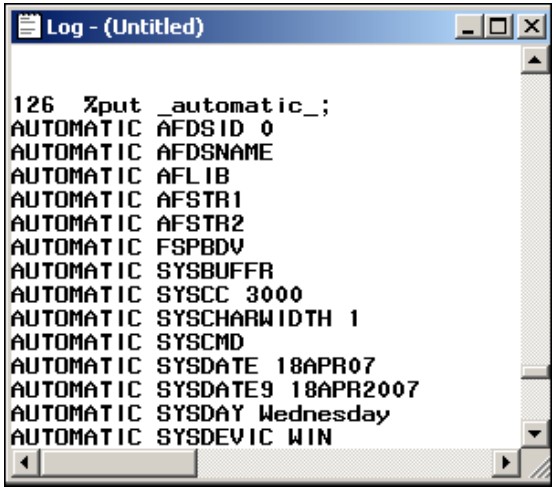
- Some automatic macro variables have values that change automatically, based on submitted SAS statements:

Name	Value
SYSLAST	name of the most recently created SAS data set in the form *libref.name*. If no data set was created, the value is _NULL_.
SYSPARM	text specified at program invocation.

Displaying Automatic Macro Variables

- The values of automatic macro variables can be displayed in the SAS log by specifying the _AUTOMATIC_ argument in the %PUT statement.

```
%PUT _AUTOMATIC_;
```



The values of the macro variables **SYSDATE**, **SYSDATE9**, and **SYSTIME** are character strings, **not** SAS date or time values.

The %LET Statement

- The %LET statement enables you to define a macro variable and assign it a value.

General form of the %LET statement:

%LET *variable*=*value*;

The %LET Statement

- Rules for the %LET statement:
  - *Variable* can be any name following the SAS naming convention.
  - *Value* can be any string.
  - If *variable* already exists in the symbol table, *value* **replaces** the current value.
  - If either *variable* or *value* contains a macro statement or macro variable reference, the trigger is evaluated **before** the assignment is made.

continued...

The %LET Statement

- Rules for the %LET statement:
  - The maximum length is 64K characters.
  - The minimum length is 0 characters (*null value*).
  - Numeric tokens are stored as character strings.
  - Mathematical expressions are **not** evaluated.
  - The case of *value* is preserved.
  - Quotes bounding literals are stored as part of *value*.
  - Leading and trailing blanks **are removed** from *value* before the assignment is made.

%LET Statement Examples

- Use the rules on the previous page to determine the values assigned to macro variables by these %LET statements:

```
%let name=Ed Norton;
%let name2=' Ed Norton ';
%let title="Joan's Report";
%let start=;
%let total=0;
%let sum=3+4;
%let total=&total+&sum;
%let x=varlist;
```

Value
Ed Norton
' Ed Norton '
"Joan's Report"
0
3+4
0+3+4
varlist

Deleting User-Defined Macro Variables

- The %SYMDEL statement deletes one or more user-defined macro variables from the global symbol table.

To release memory, delete macro variables from the global symbol table when they are no longer needed.

General form of the %SYMDEL statement:

%SYMDEL *macro-variables*;

Example: Delete the macro variables OFFICE and UNITS.

```
%symdel office units;
```

Referencing Macro Variables

- You can reference macro variables anywhere in your program, including these special situations:

Macro variable references adjacent to leading and/or trailing text:

text&*variable*
&*variable***text**
text&*variable***text**

Adjacent macro variable references:

&*variable*&*variable*

Combining Macro Variables with Text

```
%let month=jan;
proc chart data=orion.y2000&month;
   hbar week / sumvar=sale;
run;
proc plot data=orion.y2000&month;
   plot sale*day;
run;
```

generates

```
proc chart data=orion.y2000jan;
   hbar week / sumvar=sale;
run;
proc plot data=orion.y2000jan;
   plot sale*day;
run;
```

Combining Macro Variables with Text

- This example illustrates adjacent macro variable references.

Example: Modify the previous program to allow both the **month** and the **year** to be substituted.

```
%let year=2000;
%let month=jan;
proc chart data=orion.y&year&month;
   hbar week / sumvar=sale;
run;
proc plot data=orion.y&year&month;
   plot sale*day;
run;
```

Combining Macro Variables with Text

- You can place text immediately after a macro variable reference if it does not change the reference.

Example: Modify the previous program to substitute the name of an analysis variable.

```
%let year=2000;
%let month=jan;
%let var=sale;
proc chart data=orion.y&year&month;
   hbar week / sumvar=&var;
run;
proc plot data=orion.y&year&month;
   plot &var*day;
run;
```

Macro Variable Name Delimiter

- The word scanner recognizes the end of a macro variable reference when it encounters a character that cannot be part of the reference.
- A *period* (.) is a special delimiter that ends a macro variable reference. The period does not appear as text when the macro variable is resolved.

Macro Variable Name Delimiter

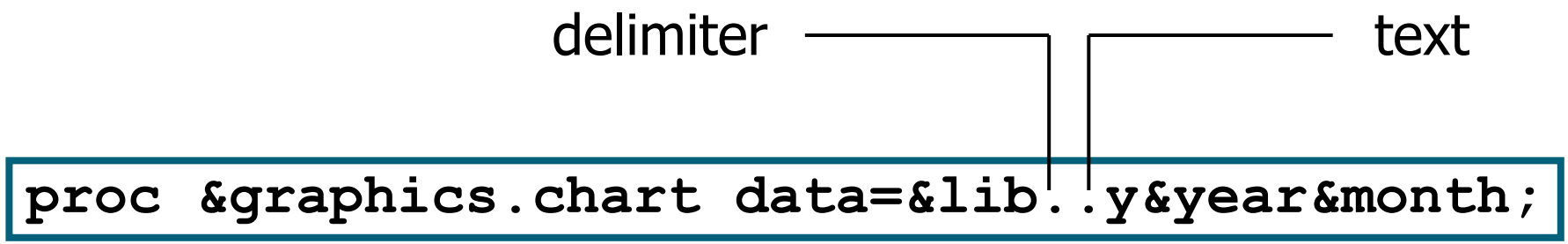
```
%let graphics=g;
%let year=2000;
%let month=jan;
%let var=sale;
proc &graphics.chart data=orion.y&year&month;
   hbar week / sumvar=&var;
run;
proc &graphics.plot data=orion.y&year&month;
   plot &var*day;
run;
```

Macro Variable Name Delimiter

- Use another period after the delimiter period to supply the needed token.

```
%let lib=orion;
...
libname &lib 'SAS-data-library';
proc &graphics.chart data=&lib..y&year&month;
...
proc &graphics.plot data=&lib..y&year&month;
```

Macro Variable Name Delimiter



- The first period is treated as a delimiter, the second as text.

- The compiler receives this:

```
...
proc gchart data=orion.y2000jan;
...
```

Macro Functions

Selected character string manipulation functions:

%UPCASE	translates letters from lowercase to uppercase.
%SUBSTR	extracts a substring from a character string.
%SCAN	extracts a word from a character string.
%INDEX	searches a character string for specified text.

Other functions:

%EVAL	performs arithmetic and logical operations.
%SYSFUNC	executes SAS functions.
%STR	quotes special characters.
%NRSTR	quotes special characters, including macro triggers.

The %SUBSTR Function

- General form of the %SUBSTR function:

%SUBSTR(*argument*, *position* <,*n*>)

- The %SUBSTR function returns the portion of *argument* beginning at *position* for a length of *n* characters.
- When *n* is not supplied, the %SUBSTR function returns the portion of *argument* beginning at *position* to the end of *argument*.

The %SCAN Function

- General form of the %SCAN function:

%SCAN(*argument*, *n* <,*delimiters*>)

- The %SCAN function does the following:
  - returns the *n*th word of *argument*, where words are strings of characters separated by delimiters
  - uses a default set of delimiters if none are specified
  - returns a null string if there are fewer than *n* words in *argument*

The %EVAL Function

- General form of the %EVAL function:

%EVAL(*expression*)

- The %EVAL function does the following:
  - performs arithmetic and logical operations
  - truncates non-integer results
  - returns a text result
  - returns 1 (true) or 0 (false) for logical operations
  - returns a null value and issues an error message when non-integer values are used in arithmetic operations

The %EVAL Function

- Example: Compute the first year of a range based on the current date.

```
%let thisyr=%substr(&sysdate9,6);
%let lastyr=%eval(&thisyr-1);
proc means data=orion.order_fact maxdec=2 min max mean;
   class order_type;
   var total_retail_price;
   where year(order_date) between &lastyr and &thisyr;
   title1 "Orders for &lastyr and &thisyr";
   title2 "(as of &sysdate9)";
run;
```

The %SYSFUNC Function

- The %SYSFUNC macro function executes SAS functions.

- General form of the %SYSFUNC function:

%SYSFUNC(*SAS function(argument(s)) <,format>*)

  - *SAS function(argument(s))* is the name of a SAS function and its corresponding arguments.
  - The second argument is an optional format for the value returned by the first argument.

Defining a Macro

- A *macro* or *macro definition* enables you to write macro programs.
- General form of a macro definition:

```
%MACRO macro-name;
        macro-text
%MEND <macro-name>;
```

- *macro-name* follows SAS naming conventions.
- *macro-text* can include the following:
  - any text
  - SAS statements or steps
  - macro variable references
  - macro statements, expressions, or calls
  - any combination of the above

Calling a Macro

- A *macro call*
  - causes the macro to execute
  - is specified by placing a percent sign before the name of the macro
  - can be made anywhere in a program (similar to a macro variable reference)
  - represents a macro trigger
  - is **not** a statement (no semicolon required).

- General form of a macro call:

```
%macro-name
```

Simple Macro

- A macro can generate SAS code.
- Example: Write a macro that generates a PROC MEANS step. Reference macro variables within the macro.

```
%macro calc;
   proc means data=orion.order_item &stats;
      var &vars;
   run;
%mend calc;
```

  - This macro contains no macro language statements.

Simple Macro

- Example: Call the CALC macro. Precede the call with %LET statements that populate macro variables referenced within the macro.

```
%let stats=min max;
%let vars=quantity;
%calc
```

Macro Parameters

- Example: Define a macro with a *parameter list* of macro variables referenced within the macro.

```
%macro calc(stats,vars);
   proc means data=orion.order_item &stats;
      var &vars;
   run;
%mend calc;
```

Positional Parameters

- *Positional parameters* use a one-to-one correspondence between the following:
  - parameter **names** supplied on the macro definition
  - parameter **values** supplied on the macro call

```
%macro calc(stats,vars);
   proc means data=orion.order_item &stats;
      var &vars;
   run;
%mend calc;

%calc(min max,quantity)
```

Keyword Parameters

- A parameter list can include keyword parameters.
- General form of a macro definition with keyword parameters:

```
%MACRO macro-name(keyword=value, …, keyword=value);
      macro text
%MEND <macro-name>;
```

- Keyword parameters are assigned a default value after an equal (=) sign.

Keyword Parameters

- General form of a macro call with keyword parameters:

%*macro-name***(***keyword=value*, …, *keyword=value***)**

- *keyword=value* combinations can be
  - specified in any order
  - omitted from the call without placeholders.

- If omitted from the call, a keyword parameter receives its default value.

Keyword Parameters

- Example: Assign default parameter values by defining the macro with keyword parameters.

```
%macro count(opts=,start=01jan04,stop=31dec04);
   proc freq data=orion.orders;
      where order_date between
            "&start"d and "&stop"d;
      table order_type / &opts;
      title1 "Orders from &start to &stop";
   run;
%mend count;
options mprint;
%count(opts=nocum)
%count(stop=01jul04,opts=nocum nopercent)
%count()
```

Více viz např.

- http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#mcrolrefwhatsnew902.htm

- http://support.sas.com/documentation/cdl/en/mcrolref/61885/HTML/default/viewer.htm#macro-stmt.htm

- http://www.nesug.org/proceedings/nesug03/bt/bt009.pdf

- http://www2.sas.com/proceedings/sugi24/Handson/p149-24.pdf

Roland's SAS Macros:
http://www.datasavantconsulting.com/roland/

Macros by SAS users worlwide:
http://schick.tripod.com/p-index.html

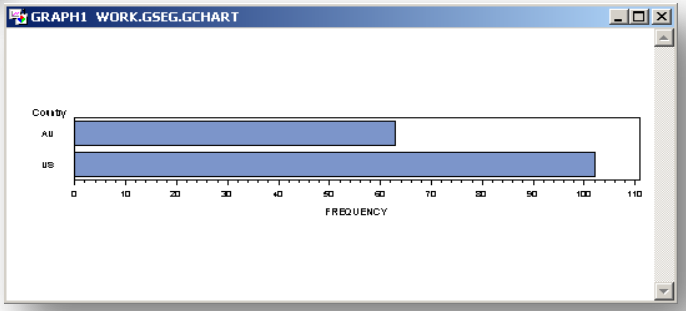
13. Úprava výstupů/reportů SASu, export ze SASu.

Personnel Data

IdNumber	LastName	FirstName	City	State	Gender	JobCode	Salary	Birth	Hired	HomePhone
1919	Adams	Gerald	Stamford	CT	M	TA2	34376	15SEP70	07JUN05	203/781-1255
1653	Alexander	Susan	Bridgeport	CT	F	ME2	35108	18OCT72	12AUG98	203/675-7715
1400	Apple	Troy	New York	NY	M	ME1	29769	08NOV85	19OCT06	212/586-0808
1350	Arthur	Barbara	New York	NY	F	FA3	32886	03SEP63	01AUG00	718/383-1549
1401	Avery	Jerry	Paterson	NJ	M	TA3	38822	16DEC68	20NOV93	201/732-8787
1499	Barefoot	Joseph	Princeton	NJ	M	ME3	43025	29APR62	10JUN95	201/812-5665
1101	Baucom	Walter	New York	NY	M	SCP	18723	09JUN80	04OCT98	212/586-8060
1333	Blair	Justin	Stamford	CT	M	PT2	88606	02APR79	13FEB03	203/781-1777
1402	Blalock	Ralph	New York	NY	M	TA2	32615	20JAN71	05DEC98	718/384-2849
1479	Bostic	Marie	New York	NY	F	TA3	38785	25DEC66	08OCT03	718/384-8816
1403	Bowden	Earl	Bridgeport	CT	M	ME1	28072	31JAN79	24DEC99	203/675-3434
1739	Boyce	Jonathan	New York	NY	M	PT1	66517	28DEC82	30JAN00	212/587-1247

The FREQ Procedure

Country	Frequency	Percent	Cumulative Frequency	Cumulative Percent
AU	63	38.18	63	38.18
US	102	61.82	165	100.00



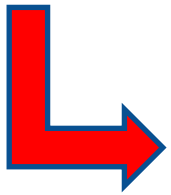
PROC Print

- The PRINT procedure prints the observations in a SAS data set, using all or some of the variables. You can create a variety of reports ranging from a simple listing to a highly customized report that groups the data and calculates totals and subtotals for numeric variables.

```
PROC PRINT <option(s)>;
 BY <DESCENDING> variable-1 <...<DESCENDING> variable-n><NOTSORTED>;
            PAGEBY BY-variable;
            SUMBY BY-variable;
 ID variable(s) <option>;
 SUM variable(s) <option>;
 VAR variable(s) <option>;
```

PROC Print

```
ods rtf file='your_file.rtf';
proc print data=empdata(obs=12);
id idnumber / style(DATA) = {background = red foreground = white}
                style(HEADER) = {background = blue foreground = white};
        title 'Personnel Data'; run;
ods rtf close;
```



Personnel Data

IdNumber	LastName	FirstName	City	State	Gender	JobCode	Salary	Birth	Hired	HomePhone
1919	Adams	Gerald	Stamford	CT	M	TA2	34376	15SEP70	07JUN05	203/781-1255
1653	Alexander	Susan	Bridgeport	CT	F	ME2	35108	18OCT72	12AUG98	203/675-7715
1400	Apple	Troy	New York	NY	M	ME1	29769	08NOV85	19OCT06	212/586-0808
1350	Arthur	Barbara	New York	NY	F	FA3	32886	03SEP63	01AUG00	718/383-1549
1401	Avery	Jerry	Paterson	NJ	M	TA3	38822	16DEC68	20NOV93	201/732-8787
1499	Barefoot	Joseph	Princeton	NJ	M	ME3	43025	29APR62	10JUN95	201/812-5665
1101	Baucom	Walter	New York	NY	M	SCP	18723	09JUN80	04OCT98	212/586-8060
1333	Blair	Justin	Stamford	CT	M	PT2	88606	02APR79	13FEB03	203/781-1777
1402	Blalock	Ralph	New York	NY	M	TA2	32615	20JAN71	05DEC98	718/384-2849
1479	Bostic	Marie	New York	NY	F	TA3	38785	25DEC66	08OCT03	718/384-8816
1403	Bowden	Earl	Bridgeport	CT	M	ME1	28072	31JAN79	24DEC99	203/675-3434
1739	Boyce	Jonathan	New York	NY	M	PT1	66517	28DEC82	30JAN00	212/587-1247

Více na:

http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a000057825.htm

PROC Report

- PROC REPORT combines features of PROC PRINT, PROC SUMMARY, PROC SORT, and the data step – it can sort and summarize data and perform calculations.

```
proc sql;
create table smallprod as
select country,
region,
prodtype,
product,
actual label="format=comma10.2,
predict label="format=comma10.2,
month
from sashelp.prdsale
where mod(monotonic(), 75) = 0
order by ranuni(94612);
quit;
```

COUNTRY	REGION	PRODTYPE	PRODUCT	ACTUAL	PREDICT	MONTH
CANADA	EAST	FURNITURE	BED	983.00	851.00	Jun
GERMANY	EAST	OFFICE	CHAIR	197.00	747.00	Mar
U.S.A.	EAST	OFFICE	DESK	415.00	763.00	Dec
CANADA	EAST	OFFICE	DESK	234.00	452.00	Sep
CANADA	WEST	OFFICE	TABLE	778.00	231.00	Dec
CANADA	WEST	OFFICE	CHAIR	838.00	98.00	Jun
U.S.A.	EAST	FURNITURE	SOFA	662.00	566.00	Mar
GERMANY	EAST	FURNITURE	BED	770.00	110.00	Sep
U.S.A.	WEST	OFFICE	CHAIR	425.00	296.00	Mar
GERMANY	WEST	OFFICE	DESK	875.00	890.00	Sep
GERMANY	EAST	OFFICE	DESK	625.00	953.00	Dec
GERMANY	WEST	OFFICE	TABLE	881.00	817.00	Dec
U.S.A.	EAST	OFFICE	CHAIR	862.00	21.00	Jun
U.S.A.	WEST	FURNITURE	BED	550.00	369.00	Jun
CANADA	WEST	FURNITURE	SOFA	142.00	583.00	Mar
U.S.A.	WEST	OFFICE	DESK	655.00	912.00	Sep
CANADA	EAST	OFFICE	CHAIR	670.00	679.00	Mar
U.S.A.	EAST	FURNITURE	BED	153.00	37.00	Sep
GERMANY	WEST	FURNITURE	BED	468.00	576.00	Jun

PROC Report

- If we run PROC REPORT with virtually no options, we'll get output similar (but not identical) to that of PROC PRINT:

```
title 'Plain PROC REPORT';
proc report data=smallprod
            nowindows missing;
run;
```

Plain PROC REPORT

Country	Region	Product type	Product	ACTUAL	PREDICT	Month
CANADA	EAST	FURNITURE	BED	983.00	851.00	Jun
GERMANY	EAST	OFFICE	CHAIR	197.00	747.00	Mar
U.S.A.	EAST	OFFICE	DESK	415.00	763.00	Dec
<LINES DELETED>						
GERMANY	WEST	FURNITURE	BED	468.00	576.00	Jun

```
proc report data=smallprod
            nowindows
            missing
            headline headskip;
run;
```

Country	Region	Product type	Product	ACTUAL	PREDICT	Month
CANADA	EAST	FURNITURE	BED	983.00	851.00	Jun
GERMANY	EAST	OFFICE	CHAIR	197.00	747.00	Mar
U.S.A.	EAST	OFFICE	DESK	415.00	763.00	Dec
<LINES DELETED>						
GERMANY	WEST	FURNITURE	BED	468.00	576.00	Jun

PROC Report

```
proc report data=smallprod
        nowindows missing
        headline headskip;
   column country region product
          month predict actual;
run;
```

Country	Region	Product	Month	PREDICT	ACTUAL
CANADA	EAST	BED	Jun	851.00	983.00
GERMANY	EAST	CHAIR	Mar	747.00	197.00
U.S.A.	EAST	DESK	Dec	763.00	415.00
<LINES DELETED>					
GERMANY	WEST	BED	Jun	576.00	468.00

- In some ways, the DEFINE statement is the real workhorse of PROC REPORT.
- It defines how variables are grouped, sorted, summarized, and displayed.

```
proc report data=smallprod
        nowindows missing
        headline headskip;
   column country region product
          month predict actual;
   define country / order;
   define region / order;
run;
```

Country	Region	Product	Month	PREDICT	ACTUAL
CANADA	EAST	BED	Jun	851.00	983.00
		DESK	Sep	452.00	234.00
		CHAIR	Mar	679.00	670.00
	WEST	TABLE	Dec	231.00	778.00
		CHAIR	Jun	98.00	838.00
		SOFA	Mar	583.00	142.00
GERMANY	EAST	CHAIR	Mar	747.00	197.00
<LINES DELETED>					

PROC Report

The GROUP option on the DEFINE statement tells PROC REPORT that you want to create a SUMMARY report rather than a detail report, summarizing by the variables with DEFINE / ORDER.

All columns should be either GROUP variables or numeric variables to be summarized. Numeric variables are summed by default.

```
proc report data=smallprod
        nowindows missing
        headline headskip;
   column country region
          predict actual;
   define country / group;
   define region / group;
run;
```

Country	Region	PREDICT	ACTUAL
CANADA	EAST	1,982.00	1,887.00
	WEST	912.00	1,758.00
GERMANY	EAST	1,810.00	1,592.00
	WEST	2,283.00	2,224.00
U.S.A.	EAST	1,387.00	2,092.00
	WEST	1,577.00	1,630.00

The most common statistic is probably SUM, but other statistics (MEAN, STD, MIN, MAX, and so forth) are also available and can be specified in the DEFINE statement (only one at a time). N and PCTN can also be specified (either in the COLUMN statement, or for a particular variable).

```
proc report data=smallprod
        nowindows missing
        headline headskip;
   column country region
          predict actual;
   define country / group;
   define region / group;
   define predict / mean;
   define actual / mean;
run;
```

Country	Region	PREDICT	ACTUAL
CANADA	EAST	660.67	629.00
	WEST	304.00	586.00
GERMANY	EAST	603.33	530.67
	WEST	761.00	741.33
U.S.A.	EAST	346.75	523.00
	WEST	525.67	543.33

Více na: www.excursive.com/sas/ProcReportSlides.pdf

http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a002473620.htm

http://www.sas.com/offices/NA/canada/downloads/presentations/Vancouver_Spring_2007/Craig.pdf

Global Statements

- The following are global statements that enhance reports:
  - OPTIONS
  - TITLE
  - FOOTNOTE
  - ODS
- Global statements are specified anywhere in your SAS program and they remain in effect until canceled, changed, or your SAS session ends.

The OPTIONS Statement

- The *OPTIONS statement* changes the value of one or more SAS system options.
- General form of the OPTIONS statement:

OPTIONS *option(s)*;

  - Some SAS system options change the appearance of a report.
  - The OPTIONS statement is **not** usually included in a PROC or DATA step.

SAS System Options for Reporting

- Selected SAS System Options:

DATE (default)	displays the date and time that the SAS session began at the top of each page of SAS output.
NODATE	does not display the date and time that the SAS session began at the top of each page of SAS output.
NUMBER (default)	prints page numbers on the first line of each page of SAS output.
NONUMBER	does not print page numbers on the first line of each page of SAS output.
PAGENO=*n*	defines a beginning page number (*n*) for the next page of SAS output.

continued...

SAS System Options for Reporting

- Selected SAS System Options:

CENTER (default)	centers SAS output.
NOCENTER	left-aligns SAS output.
PAGESIZE=*n* PS=*n*	defines the number of lines (*n*) that can be printed per page of SAS output.
LINESIZE=*width* LS=*width*	defines the line size (*width*) for the SAS log and SAS output.

SAS System Options for Reporting

```
options ls=80 date number;

proc means data=orion.sales;
   var Salary;
run;
```

```
                                                09:11 Monday, January 14, 2008   35

                              The MEANS Procedure

                          Analysis Variable : Salary

      N            Mean         Std Dev         Minimum         Maximum
-------------------------------------------------------------------------
    165        31160.12        20082.67        22710.00       243190.00
-------------------------------------------------------------------------
```

80 characters wide

SAS System Options for Reporting

```
options nodate pageno=1;

proc freq data=orion.sales;
   tables Country;
run;
```

```
                                                                   1

                         The FREQ Procedure

                                              Cumulative    Cumulative
  Country     Frequency     Percent     Frequency       Percent
  ---------------------------------------------------------------
  AU                63        38.18            63         38.18
  US               102        61.82           165        100.00
```

80 characters wide

The TITLE Statement

- The *TITLE statement* specifies title lines for SAS output.
- General form of the TITLE statement:

TITLE*n* '*text* ';

- Titles appear at the top of the page.
- The default title is **The SAS System**.
- The value of ***n*** can be from 1 to 10.
- An unnumbered **TITLE** is equivalent to **TITLE1**.
- Titles remain in effect until they are changed, canceled, or you end your SAS session.

The FOOTNOTE Statement

- The *FOOTNOTE statement* specifies footnote lines for SAS output.
- General form of the FOOTNOTE statement:

FOOTNOTE*n* *'text '*;

- Footnotes appear at the bottom of the page.
- No footnote is printed unless one is specified.
- The value of ***n*** can be from 1 to 10.
- An unnumbered **FOOTNOTE** is equivalent to FOOTNOTE1.
- Footnotes remain in effect until they are changed, canceled, or you end your SAS session.

The TITLE and FOOTNOTE Statements

```
footnote1 'By Human Resource Department';
footnote3 'Confidential';

proc means data=orion.sales;
   var Salary;
   title 'Orion Star Sales Employees';
run;
```

Orion Star Sales Employees

The MEANS Procedure

Analysis Variable : Salary

N	Mean	Std Dev	Minimum	Maximum
165	31160.12	20082.67	22710.00	243190.00

By Human Resource Department

Confidential

Changing Titles and Footnotes

- **TITLE*n*** or **FOOTNOTE*n***
  - replaces a previous title or footnote with the same number
  - cancels all titles or footnotes with higher numbers.

Canceling All Titles and Footnotes

- The null TITLE statement cancels all titles.

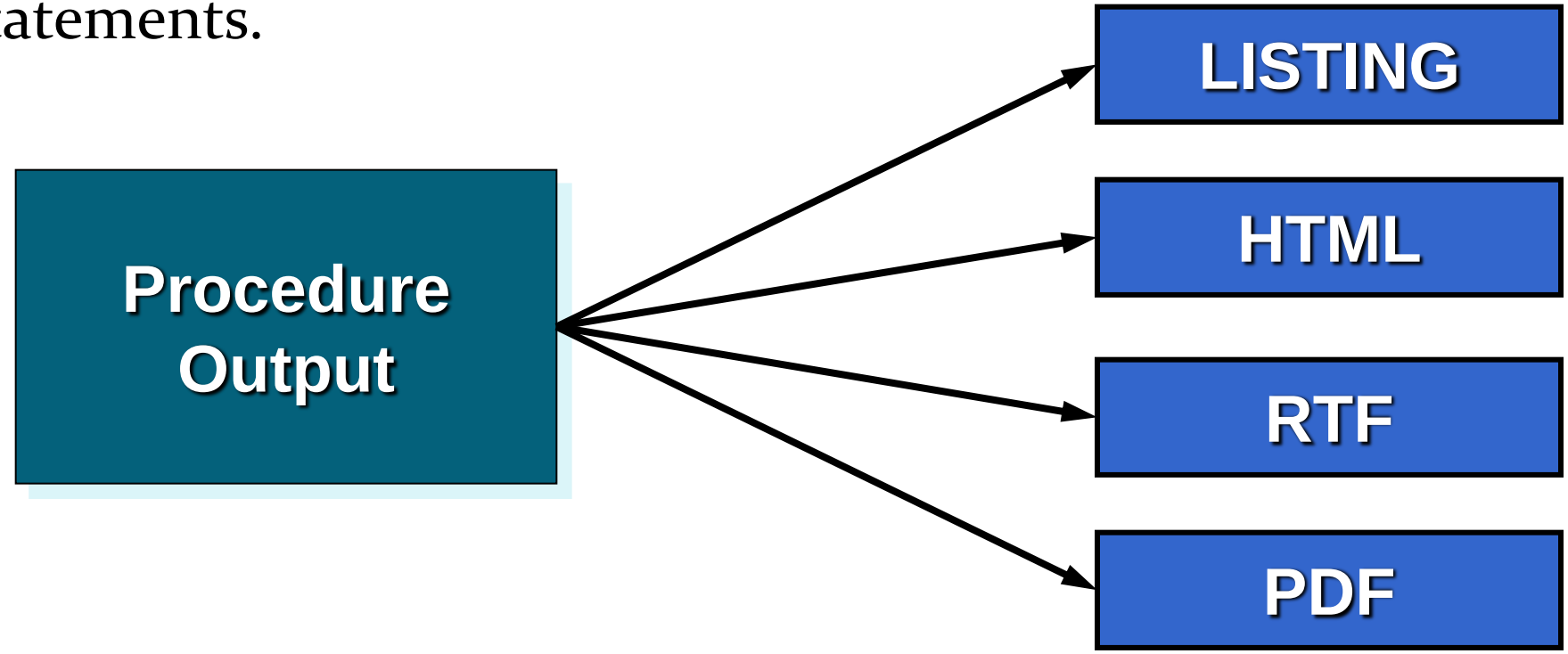
```
title;
```

- The null FOOTNOTE statement cancels all

```
footnote;
```

Output Delivery System

- Output can be sent to a variety of destinations by using ODS statements.

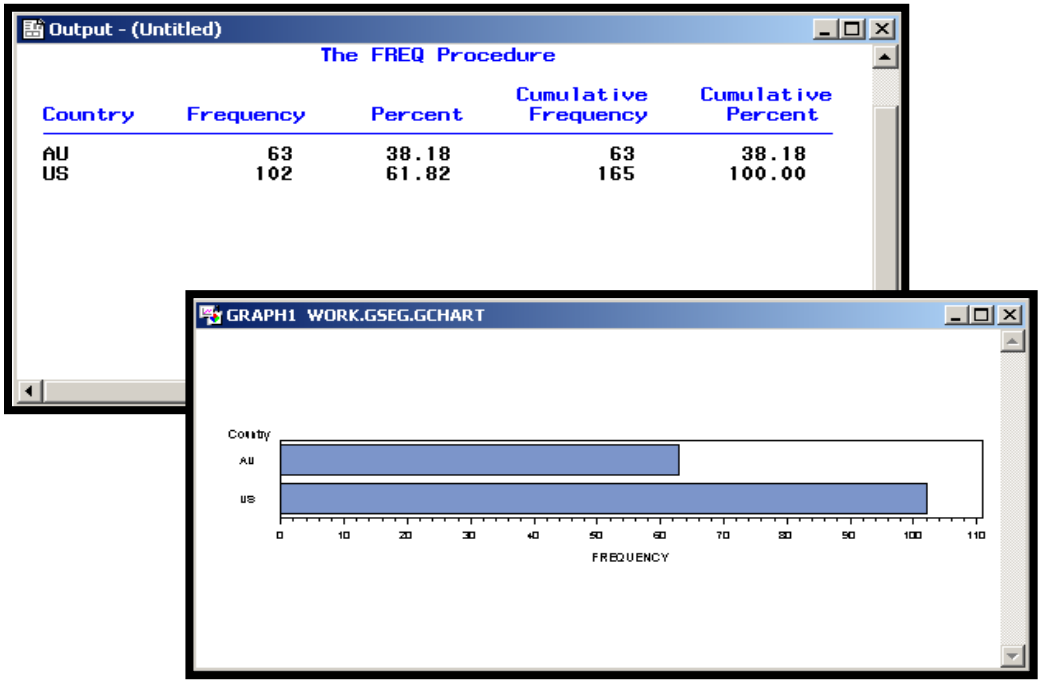


Default ODS Destination

- The LISTING destination is the default ODS destination.

```
ods listing;

proc freq data=orion.sales;
   tables Country;
run;

proc gchart
data=orion.sales;
   hbar Country / nostats;
run;
```

- The LISTING destination directs output to the OUTPUT window and the GRAPH window.



Default ODS Destination

- The ODS LISTING CLOSE statement stops sending output to the OUTPUT and GRAPH windows.

```
ods listing close;

proc freq data=orion.sales;
   tables Country;
run;

proc gchart
data=orion.sales;
   hbar Country / nostats;
run;
```

- A warning will appear in the SAS log if the LISTING destination is closed and no other destinations are active.

```
23   ods listing close;
24
25   proc freq data=orion.sales;
26      tables Country;
27   run;

WARNING: No output destinations active.
NOTE: There were 165 observations read from the data set ORION.SALES.
```

HTML, PDF, and RTF Destinations

- ODS destinations such as HTML, PDF, and RTF are opened and closed in the following manner:

ODS *destination* FILE = ' *filename.ext* **'**
<options>;

SAS code to generate a report(s)

ODS *destination* CLOSE;

Single Destination

- Output can be sent to only one destination.

```
ods listing close;

ods html file='example.html';

proc freq data=orion.sales;
   tables Country;
run;

ods html close;

ods listing;
```

It is a good habit to open the LISTING destination at the end of a program to guarantee an open destination for the next submission.

Multiple Destinations

- Output can be sent to many destinations.

```
ods listing;
ods pdf file='example.pdf';
ods rtf file='example.rtf';

proc freq data=orion.sales;
   tables Country;
run;

ods pdf close;
ods rtf close;
```

- To view the results, all destinations except the LISTING destination must be closed.

Multiple Destinations

- Use _ALL_ in the ODS CLOSE statement to close all open destinations including the LISTING destination.

```
ods listing;
ods pdf file='example.pdf';
ods rtf file='example.rtf';

proc freq data=orion.sales;
   tables Country;
run;

ods _all_ close;
ods listing;
```

Multiple Procedures

- Output from many procedures can be sent to ODS destinations.

```
ods listing;
ods pdf file='example.pdf';
ods rtf file='example.rtf';

proc freq data=orion.sales;
   tables Country;
run;

proc means data=orion.sales;
   var Salary;
run;

ods _all_ close;
ods listing;
```

File Location

- A path can be specified to control the location of where the file is stored.

```
ods html file='s:\workshop\example.html';

proc freq data=orion.sales;
   tables Country;
run;

proc means data=orion.sales;
   var Salary;
run;

ods html close;
```

- If no path is specified, the file is saved in the current default directory.

STYLE= Option

- Use a STYLE= option in the ODS destination statement to specify a style definition.

```
ODS destination FILE = 'filename.ext'
                       STYLE = style-
definition;
```

- A *style definition* describes how to display the presentation aspects such as colors and fonts of SAS output.
- STYLE= cannot be used with the LISTING destination.

SAS Supplied Style Definitions

Analysis	Astronomy	Banker	BarrettsBlue
Beige	blockPrint	Brick	Brown
Curve	D3d	Default	Education
EGDefault	Electronics	fancyPrinter	Festival
FestivalPrinter	Gears	Journal	Magnify
Meadow	MeadowPrinter	Minimal	Money
NoFontDefault	Normal	NormalPrinter	Printer
Rsvp	Rtf	sansPrinter	sasdocPrinter
Sasweb	Science	Seaside	SeasidePrinter
serifPrinter	Sketch	Statdoc	Statistical
Theme	Torn	Watercolor	

SAS Supplied Style Definitions

- The following style definitions are new to SAS 9.2:

grayscalePrinter	Harvest	HighContrast
Journal2	Journal3	Listing
monochromePrinter	Ocean	Solutions

Examples

STYLE=DEFAULT

The FREQ Procedure

Country	Frequency	Percent	Cumulative Frequency	Cumulative Percent
AU	63	38.18	63	38.18
US	102	61.82	165	100.00

STYLE=SASWEB

The FREQ Procedure

Country	Frequency	Percent	Cumulative Frequency	Cumulative Percent
AU	63	38.18	63	38.18
US	102	61.82	165	100.00

Examples

STYLE=PRINTER

The FREQ Procedure

Country	Frequency	Percent	Cumulative Frequency	Cumulative Percent
AU	63	38.18	63	38.18
US	102	61.82	165	100.00

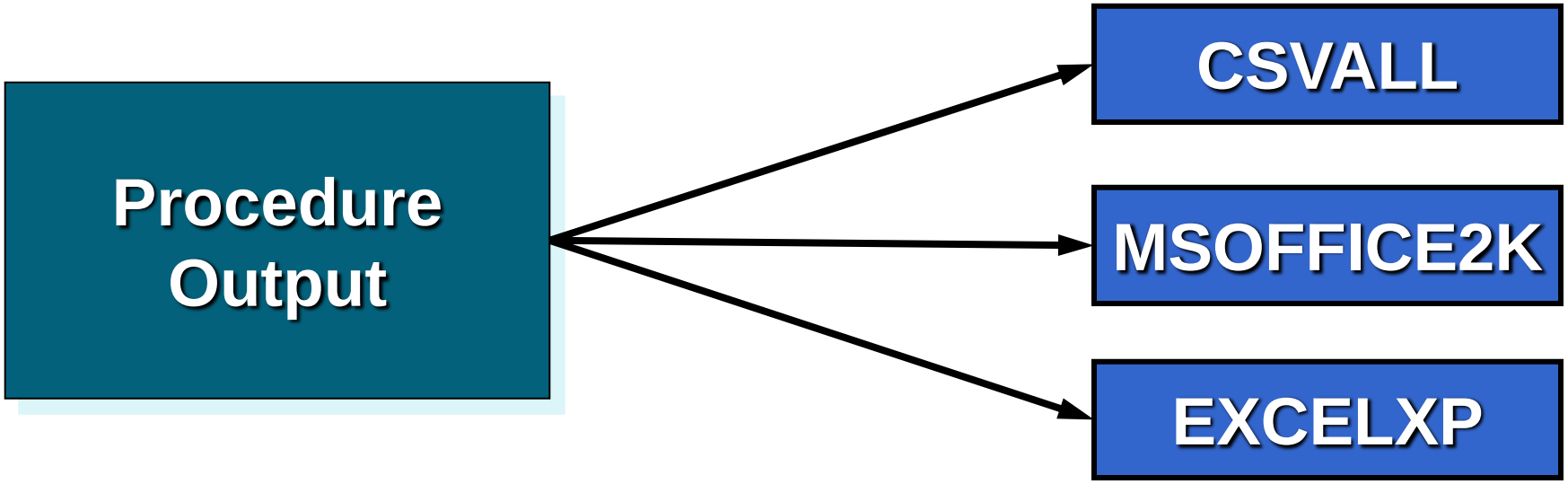
STYLE=JOURNAL

The FREQ Procedure

Country	*Frequency*	*Percent*	*Cumulative Frequency*	*Cumulative Percent*
AU	63	38.18	63	38.18
US	102	61.82	165	100.00

Destinations Used with Excel

- The following destinations create files that can be opened in Excel.



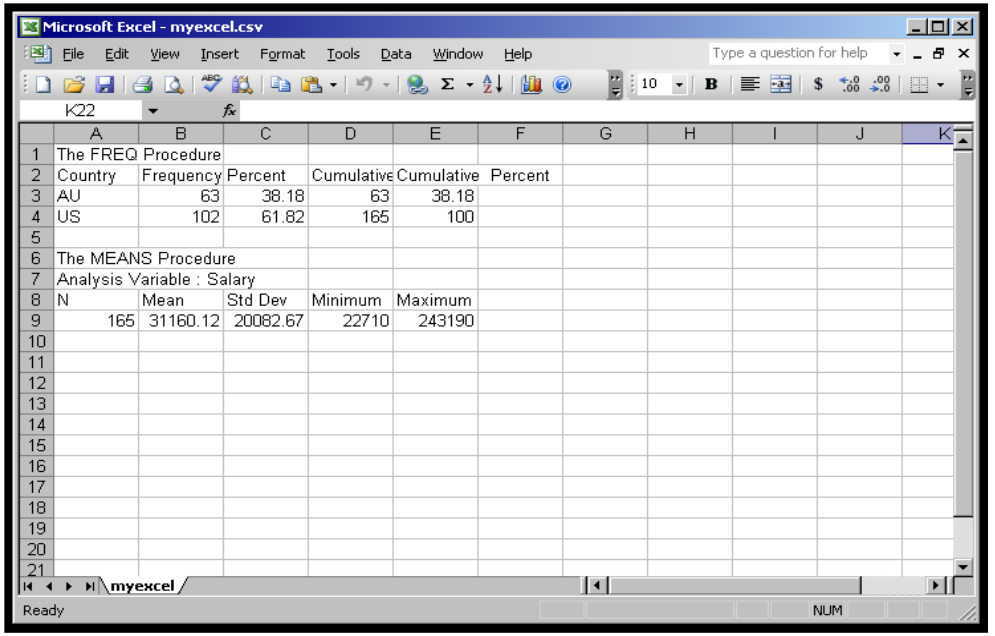
Destinations Used with Excel

Destination	Type of File	Viewed In
CSVALL	Comma-Separated Value	Editor or Microsoft Excel
MSOFFICE2K	Hypertext Markup Language	Web Browser or Microsoft Word or Microsoft Excel
EXCELXP	Extensible Markup Language	Microsoft Excel

CSVALL Destination

- CSVALL does not include any style information.

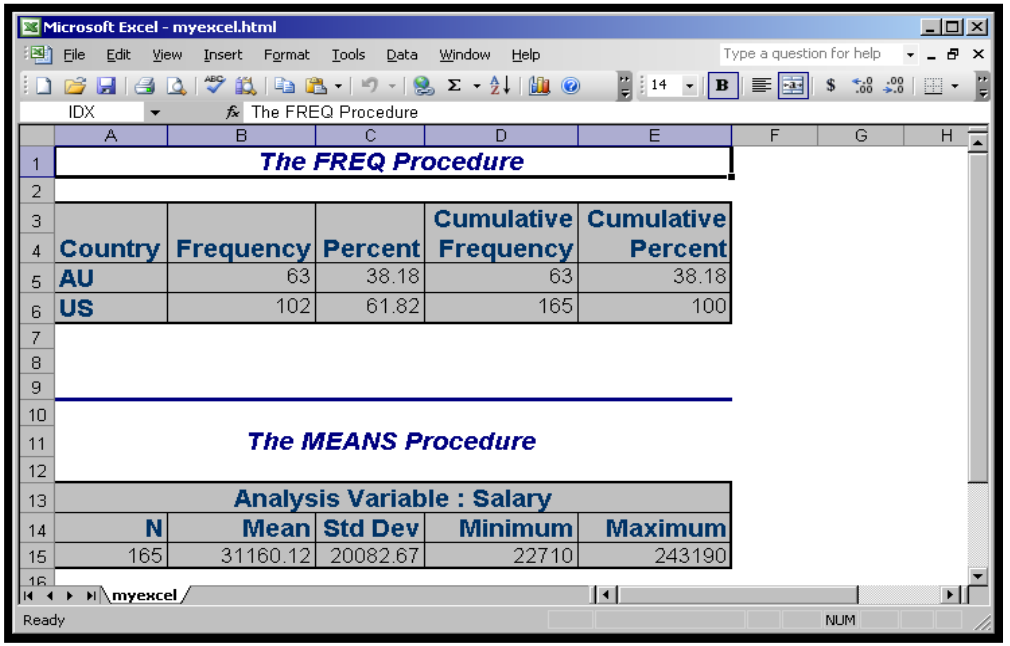
```
ods csvall file='myexcel.csv';

proc freq data=orion.sales;
   tables Country;
run;

proc means data=orion.sales;
   var Salary;
run;

ods csvall close;
```



MSOFFICE2K Destination

- MSOFFICE2K keeps the style information including spanning headers.

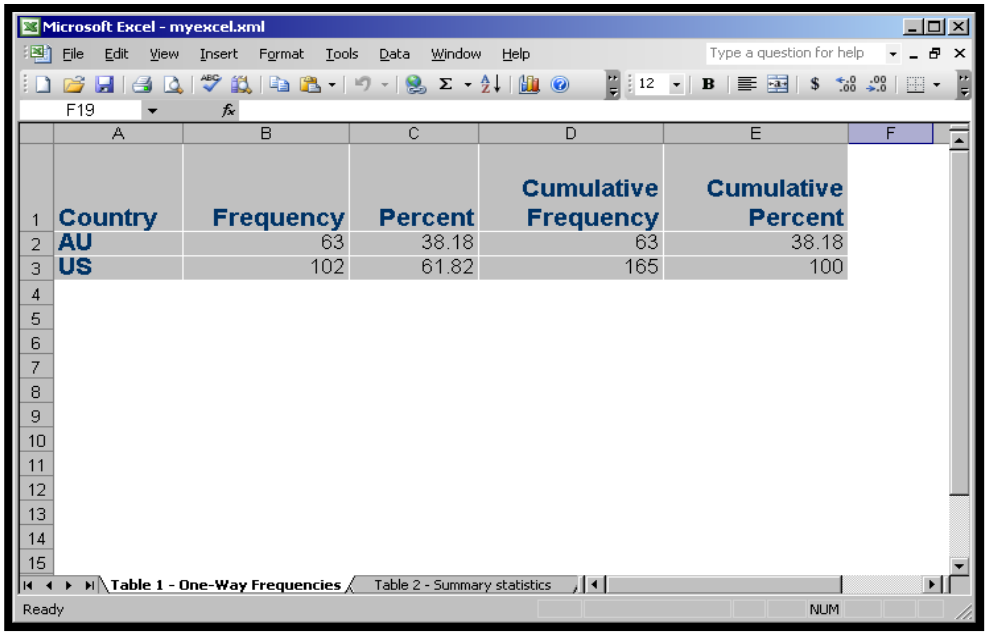
```
ods msoffice2k file='myexcel.html';

proc freq data=orion.sales;
   tables Country;
run;

proc means data=orion.sales;
   var Salary;
run;

ods msoffice2k close;
```



EXCELXP Destination

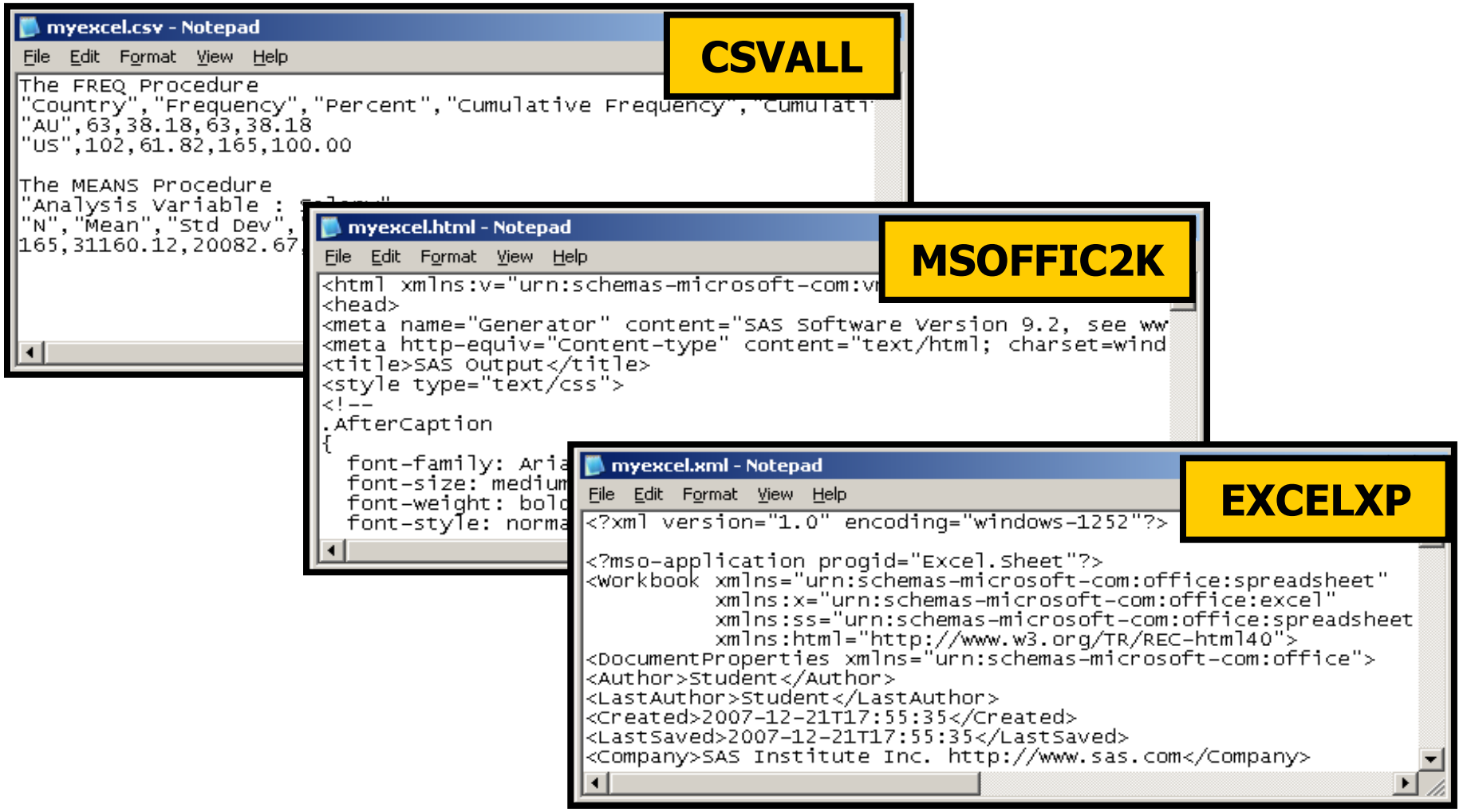
- EXCELXP keeps the style information and each procedure is a separate sheet.

```
ods tagsets.excelxp file='myexcel.xml';

proc freq data=orion.sales;
   tables Country;
run;

proc means data=orion.sales;
   var Salary;
run;

ods tagsets.excelxp close;
```



	A	B	C	D	E
1	Country	Frequency	Percent	Cumulative Frequency	Cumulative Percent
2	AU	63	38.18	63	38.18
3	US	102	61.82	165	100

The file you are creating is not an Excel file!!!



Data _null_

```
/*WRITE RAW DATA SEPARATED BY BLANKS*/
data _null_;
  set iris;
  file "c:\temp\iris.dat";
  put  sepallen
 sepalwid
 petallen
 petalwid
 species;
run;
```

```
/*WRITE RAW DATA SEPARATED BY TABS*/
data _null_;
  set iris;
  file "c:\temp\iris.txt"
dlm="09"X;
  put  sepallen
 sepalwid
 petallen
 petalwid
 species;
run;
```

```
/*WRITE RAW DATA SEPARATED BY COMMAS*/
data _null_;
  set iris;
  file "c:\temp\iris.csv" dlm=",";
  put  sepallen
 sepalwid
 petallen
 petalwid
 species;
run;
```

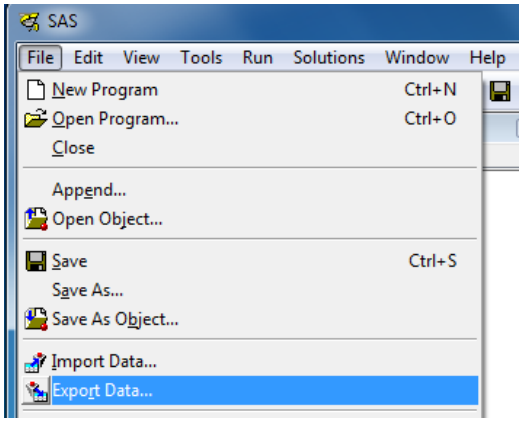
```
/*WRITE RAW DATA INTO SPECIFIED COLUMNS*/
data _null_;
  set iris;
  file " c:\temp\iris_column.dat";
  put species  1-10
      sepallen 12-15
      sepalwid 17-20
      petallen 22-25
      petalwid 27-30;
run;
```

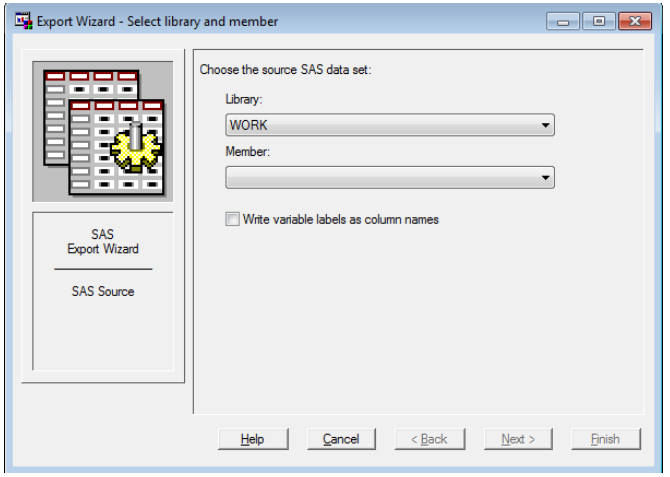
Proc Export

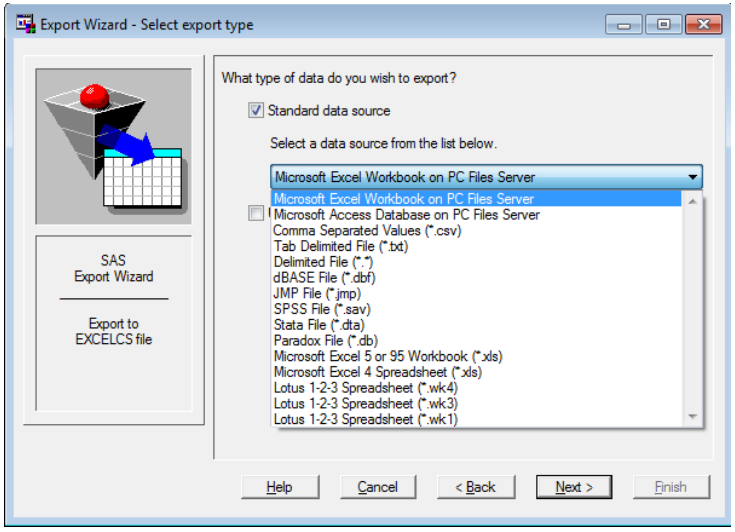
```
PROC EXPORT DATA= WORK.IRIS
            OUTFILE= "C:\temp\iris.xls"
            DBMS=EXCEL REPLACE;
     SHEET="sheet1";
RUN;
```

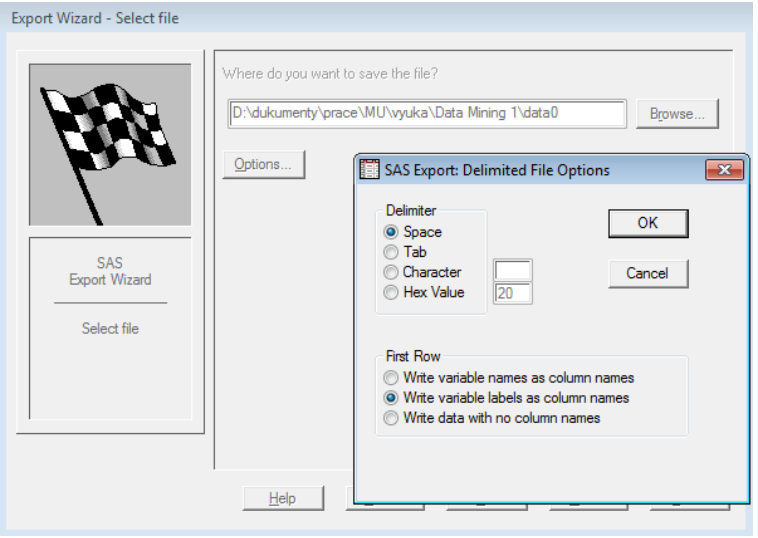
Obecně:
http://support.sas.com/documentation/cdl/en/proc/61895/HTML/default/viewer.htm#a000316288.htm

Export Wizard









14. Reference

Literatura - knihy

- Allison, P.D. (2009). *Logistic Regression using SAS: Theory and Application*, 8th printing, SAS Institute Inc.

- Anderson, R. (2007). *The Credit Scoring Toolkit: Theory and Practice for Retail Credit Risk Management and Decision Automation*, Oxford: Oxford University Press.

- Giudici, P. (2003). *Applied Data Mining: statistical methods for business and industry*, Chichester : Wiley.

- Han, J., Kamber, M. (2006). *Data mining: Concepts and Techniques*, 2nd ed. San Francisco: Morgan Kaufmann.

Literatura - knihy

- Hastie, T., Tibshirani, R., Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, New York: Springer-Verlag.

- Hosmer, D. W., Lemeshow S. (2000). *Applied Logistic Regression, Textbook and Solutions Manual* , 2nd ed., New York: John Wiley and Sons.

- Siddiqi, N. (2006). *Credit Risk Scorecards: developing and implementing intelligent credit scoring*, New Jersey: Wiley.

- Thomas, L.C. (2009). *Consumer Credit Models: Pricing, Profit, and Portfolio*, Oxford: Oxford University Press.

Literatura - knihy

- Thomas, L.C., Edelman, D.B., Crook, J.N. (2002). *Credit Scoring and Its Applications*, Philadelphia: SIAM Monographs on Mathematical Modeling and Computation.

- Wilkie, A.D. (2004). Measures for comparing scoring systems, In: Thomas, L.C., Edelman, D.B., Crook, J.N. (Eds.), *Readings in Credit Scoring*. Oxford: Oxford University Press, pp. 51-62.

- Witten, I.H., Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*, San Francisco: Morgen Kaufmann.

Literatura - časopisy

- Crook, J.N., Edelman, D.B., Thomas, L.C. (2007). Recent developments in consumer credit risk assessment. *European Journal of Operational Research,* 183 (3), 1447-1465

- Hand, D.J. and Henley, W.E. (1997). Statistical Classification Methods in Consumer Credit Scoring: a review. *Journal. of the Royal Statistical Society*, Series A., 160,No.3, 523-541.

- Harrell, F.E., Lee, K.L. and Mark, D.B. (1996). Multivariate prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in Medicine*, 15, 361-387.

- Lilliefors, H.W. (1967). On the Komogorov-Smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 62, 399-402.

Literatura - časopisy

- Nelsen, R. B. (1998). Concordance and Gini's measure of association. *Journal of Nonparametric Statistics*, 9, Isssue 3, 227–238.

- Newson R. (2006). Confidence intervals for rank statistics: Somers' *D* and extensions. *The Stata Journal*, 6(3), 309-334.

- Řezáč M. & Řezáč F. (2011). How to Measure the Quality of Credit Scoring Models. Finance a úvěr - Czech Journal of Economics and Finance, 61(5), 486-507.

- Somers R. H. (1962). A new asymmetric measure of association for ordinal variables. *American Sociological Review,* 27, 799-811.

- Thomas, L.C. (2000). A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. *International Journal of Forecasting*, 16(2), 149-172 .

Literatura - web

- Coppock, D.S. (2002). *Why Lift?, DM Review Online*, www.dmreview.com/news/53291.html

- Xu, K. (2003). *How has the literature on Gini's index evolved in past 80 years?*, www.economics.dal.ca/RePEc/dal/wparch/howgini.pdf

- Xin Ming Tu, Wan Tang (2006). *Categorical Data Analysis*. http://www.urmc.rochester.edu/smd/biostat/people/faculty/TuSite/bst466/handouts.htm

- Jiawei Han and Micheline Kamber (2006). *Data Mining: Concepts and Techniques*. http://www.cs.illinois.edu/~hanj/bk2/

- Jens Peter Dittrich (2007). *Data warehousing*. http://www.dbis.ethz.ch/education/ss2007/07_dbs_datawh/Data_Mining.pdf

- Joe Carthy (2006). Data Warehousing. http://www.csi.ucd.ie/staff/jcarthy/home/DataMining/DM-Lecture02-01.ppt

- Jan Spousta. Přednášky k data miningu. [cit. 19.03.2009] http://samba.fsv.cuni.cz/~soukup

Další zajímavé zdroje informací

- http://www.cs.uiuc.edu/homes/hanj/
- http://www-users.cs.umn.edu/~kumar/
- http://www.kdnuggets.com/
- http://www.kdnuggets.com/datasets/competitions.html
- http://www.crc.man.ed.ac.uk/conference/
- http://www.crc.man.ed.ac.uk/conference/archive/
- http://www.kmining.com/info_conferences.html
- http://en.wikipedia.org/wiki/Data_mining
- http://cs.wikipedia.org/wiki/Data_mining
- http://en.wikipedia.org/wiki/Credit_scorecards

Užitečné zdroje dat

- http://archive.ics.uci.edu/ml/
- http://kdd.ics.uci.edu/

- http://sede.neurotech.com.br:443/PAKDD2009/

- http://www.dataminingbook.com/

- http://www.stat.uni-muenchen.de/service/datenarchiv/welcome_e.html