evropský sociální fond v ČR

EVROPSKÁ UNIE

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

OP Vzdělávání pro konkurenceschopnost

UNIVERSITAS MASARYKIANA BRUNENSIS

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

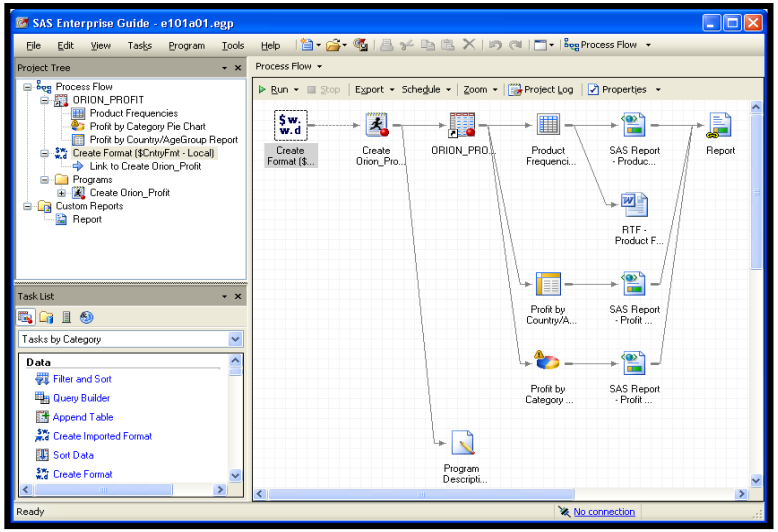
Data Mining II

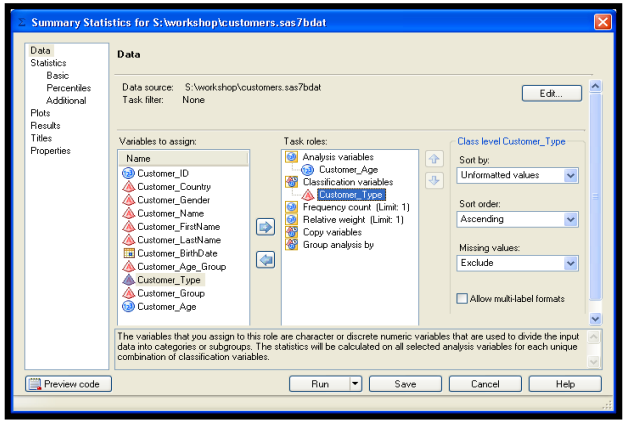
Martin Řezáč

2013

Obsah:

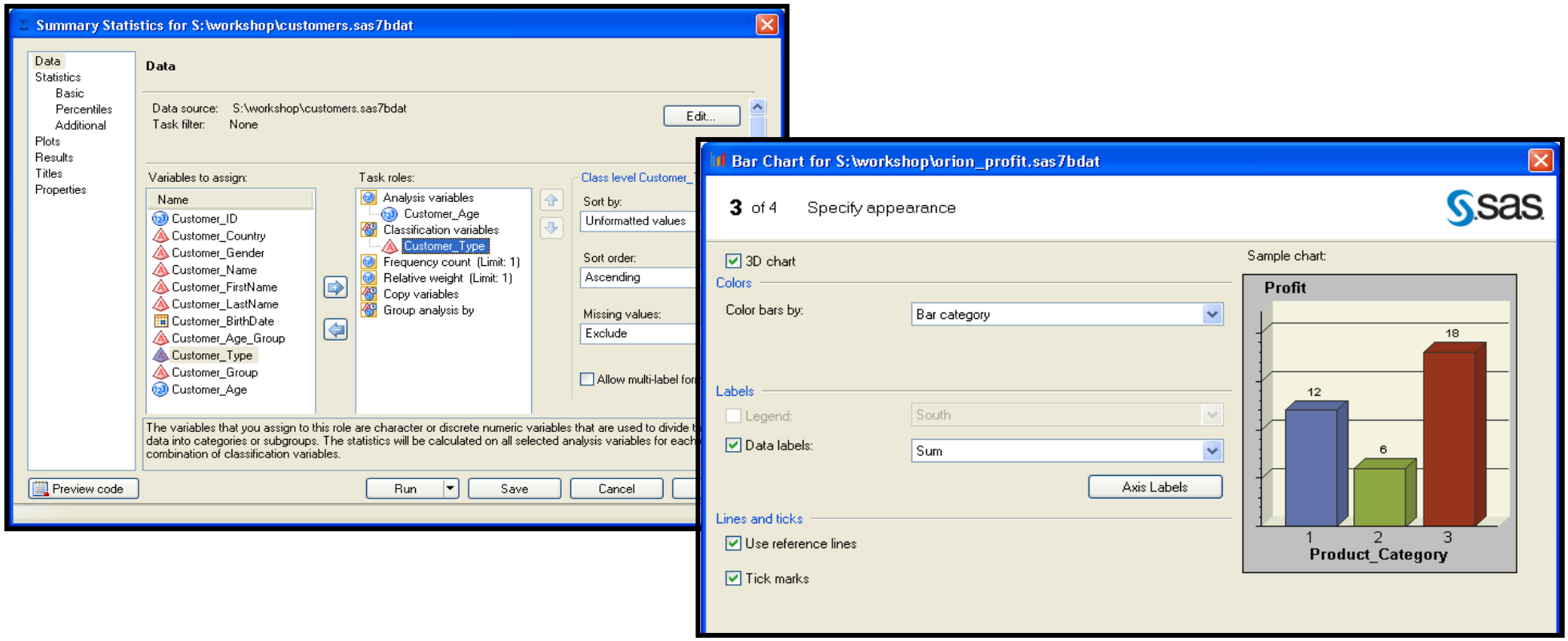
1. Úvod do SAS EG





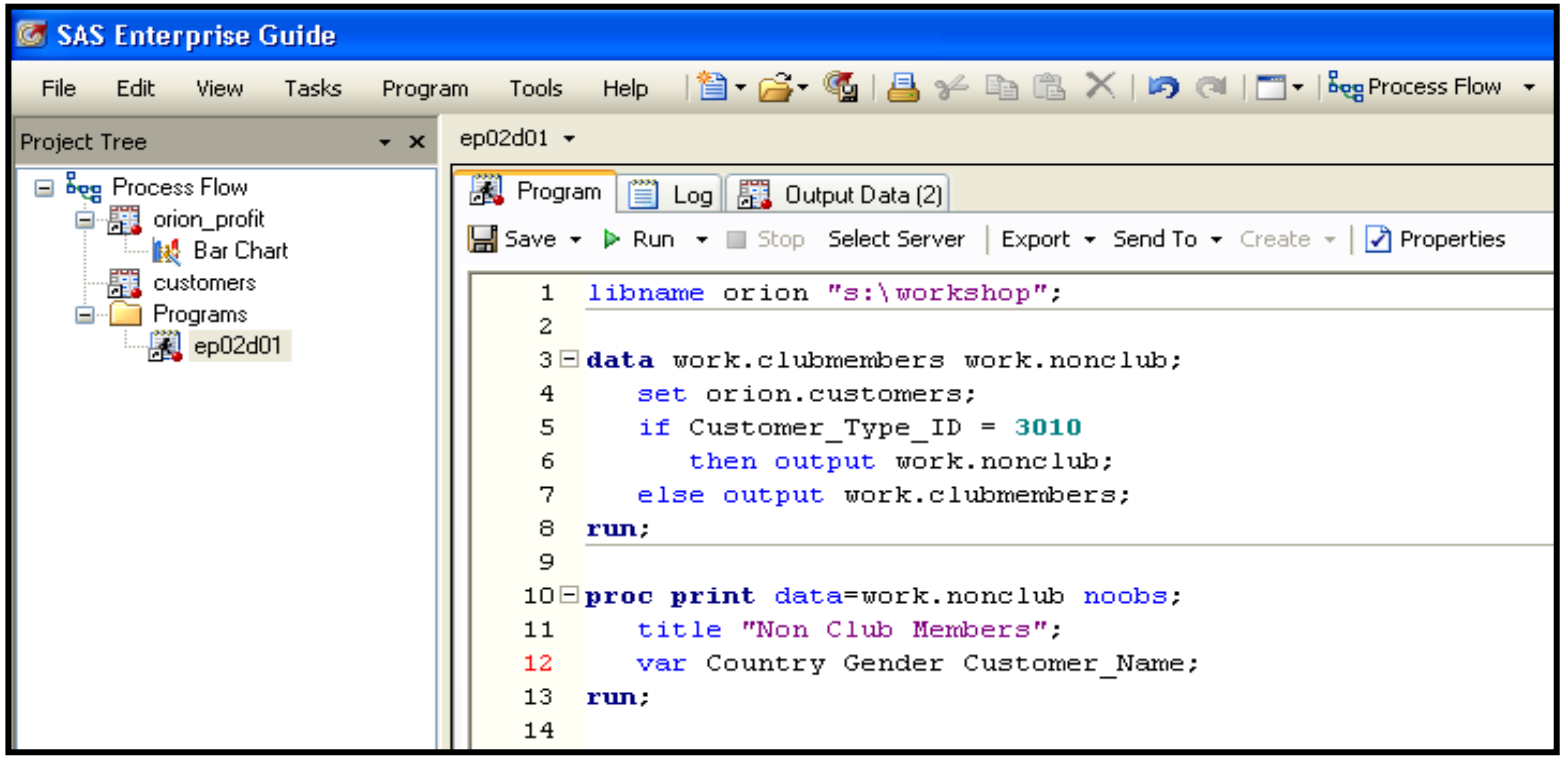
Introduction to SAS Enterprise Guide

- SAS Enterprise Guide provides a point-and-click interface for managing data and generating reports.



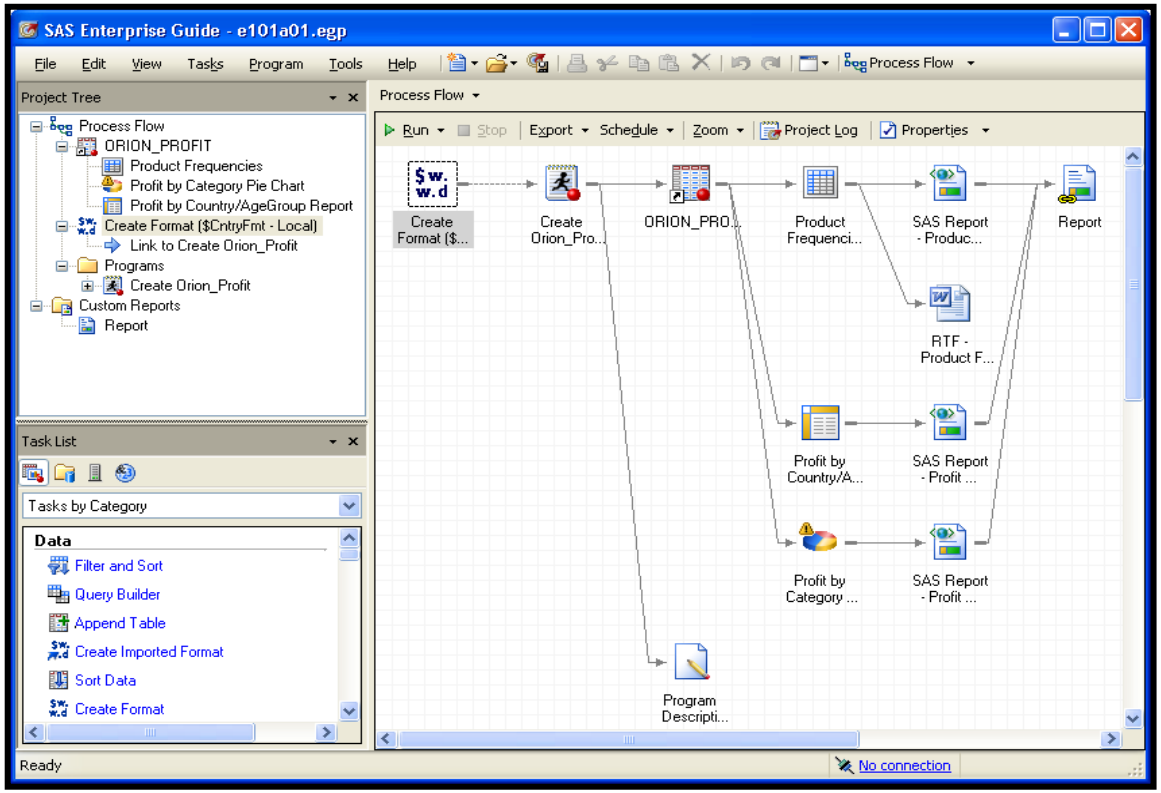
SAS Enterprise Guide Interface

- SAS Enterprise Guide also includes a full programming interface that can be used to write, edit, and submit SAS code.



SAS Enterprise Guide Interface: The Project

- A project serves as a collection of
  - data sources
  - SAS programs and logs
  - tasks and queries
  - results
  - informational notes for documentation.



You can control the contents, sequencing, and updating of a project.

SAS Programs

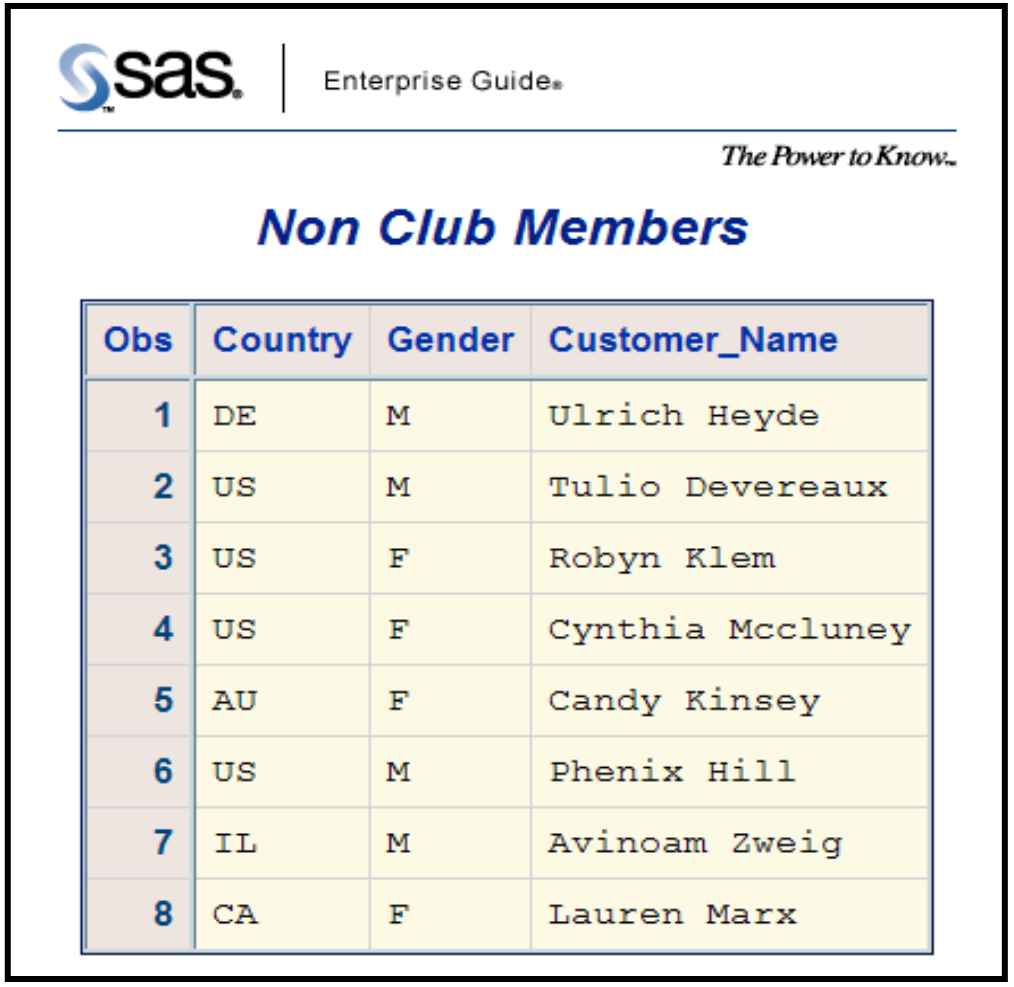
```
data work.clubmembers work.nonclub;
   set orion.customer;
   if Customer_Type_ID = 3010
      then output work.nonclub;
   else output work.clubmembers;
run;

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

DATA Step

PROC Step

ep02d01.sas

PROC PRINT Output

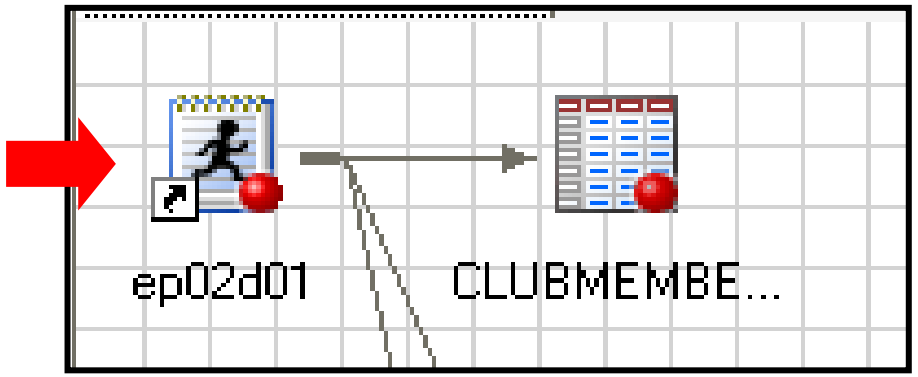


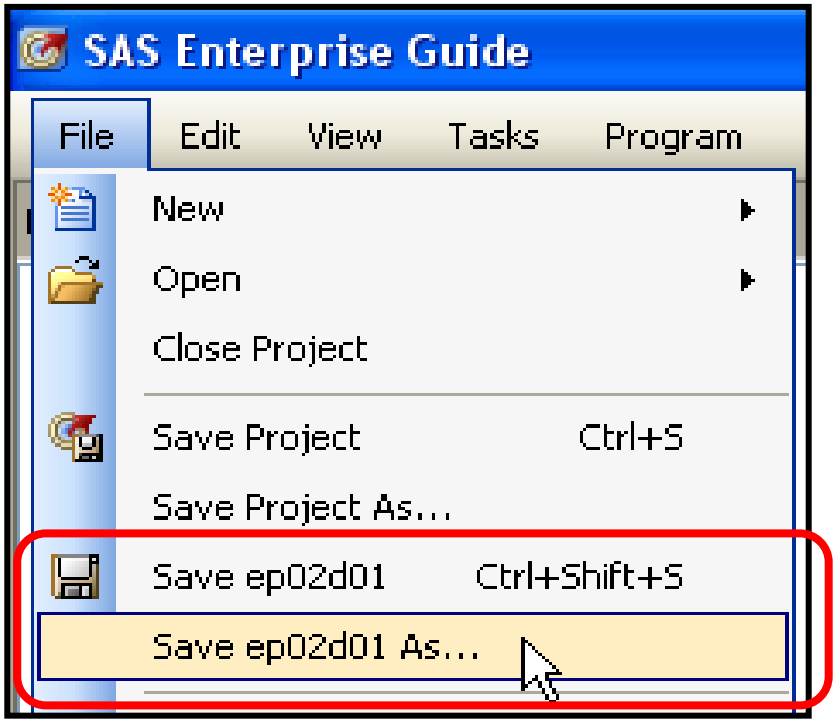
Non Club Members

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

Saving SAS Programs

- The SAS program in the project is a shortcut to the physical storage location of the .sas file. Select the program icon and then select **File** ⇨ **Save *program name*** to save the program as the same name, or **Save *program name* As...** to choose a different name or storage location.

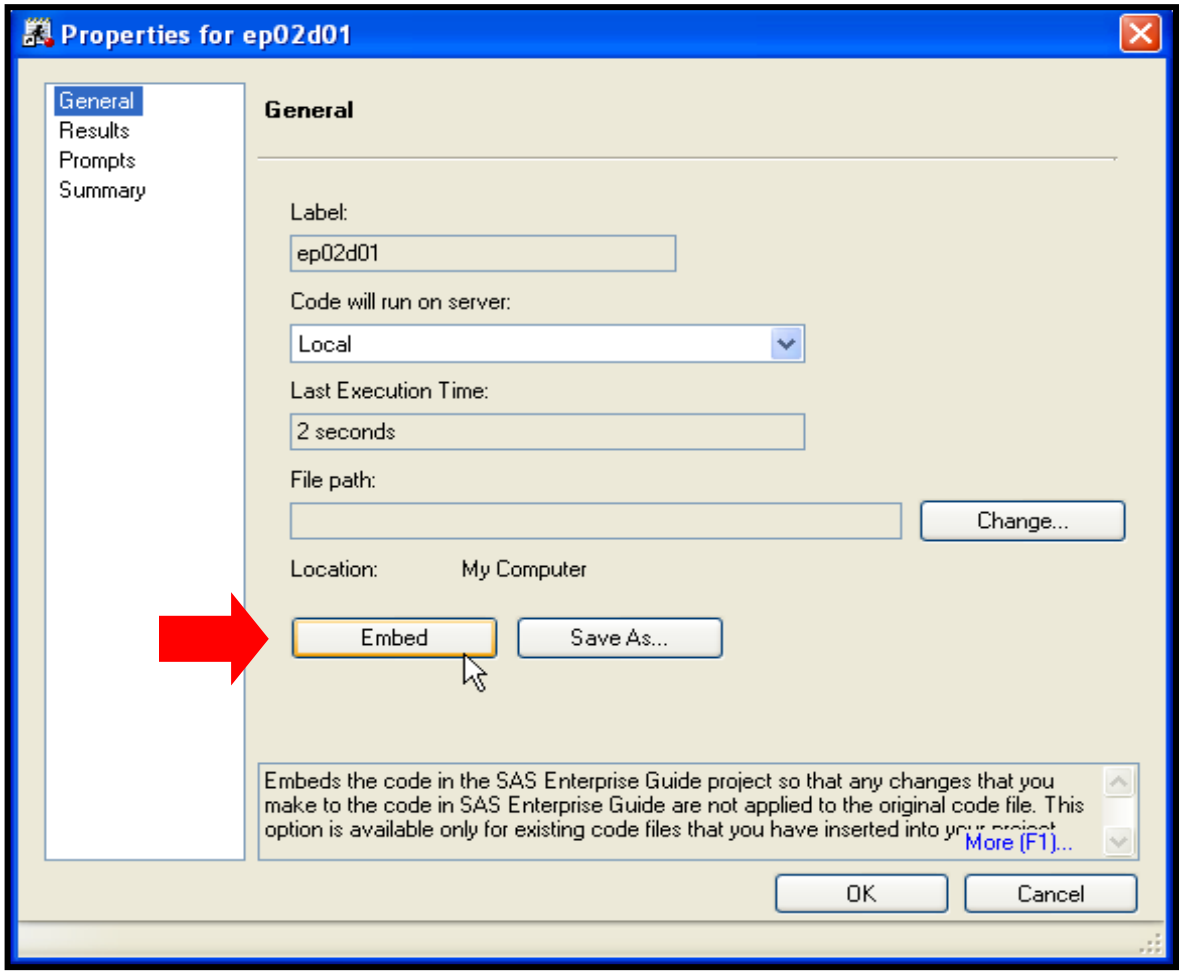




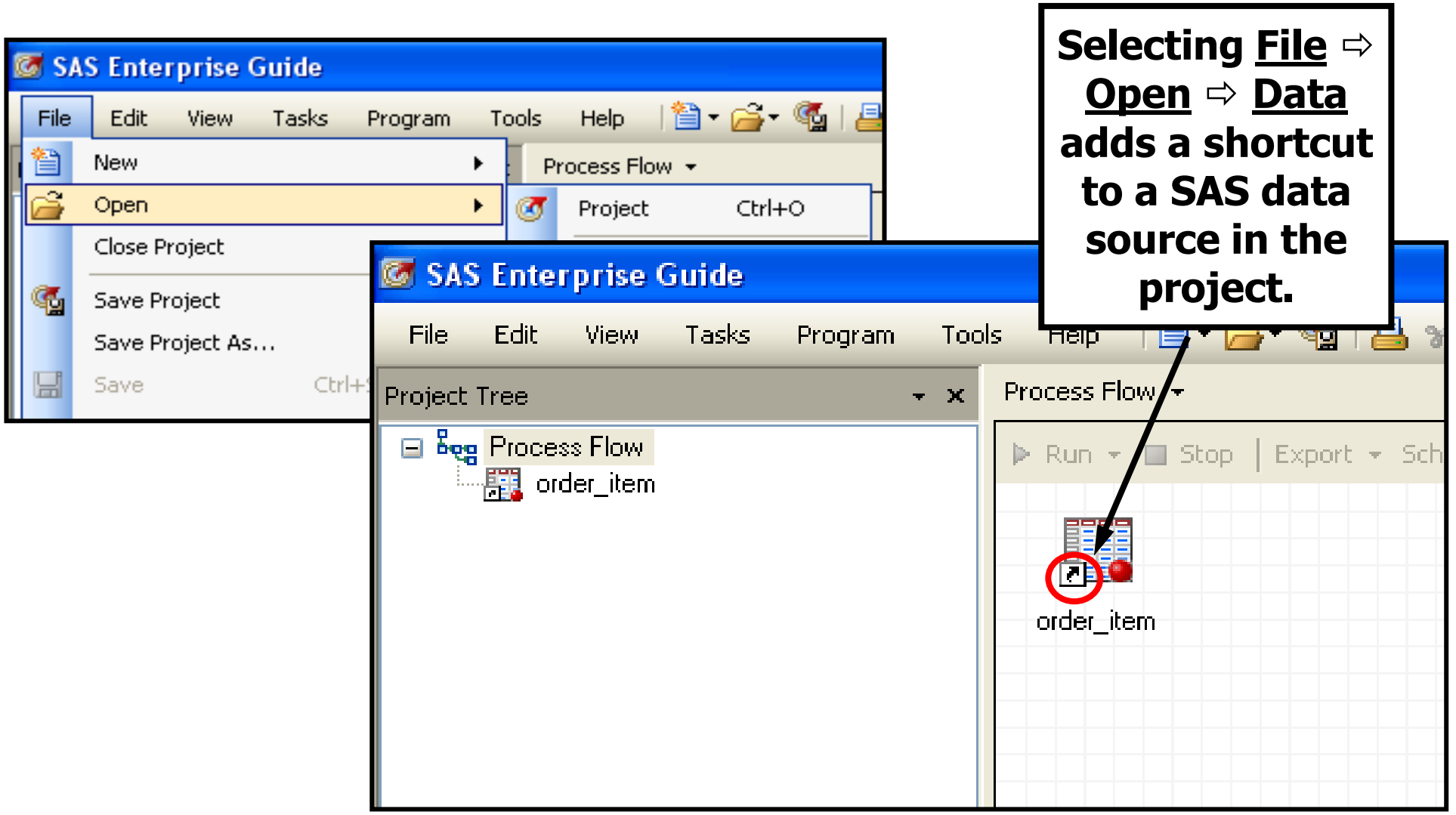
Embedding Programs in a Project

- A SAS program can also be embedded in a project so that the code is stored as part of the project .epg file.

- Right-click on the **Code** icon in a project and select **Properties** ⇨ **Embed**.

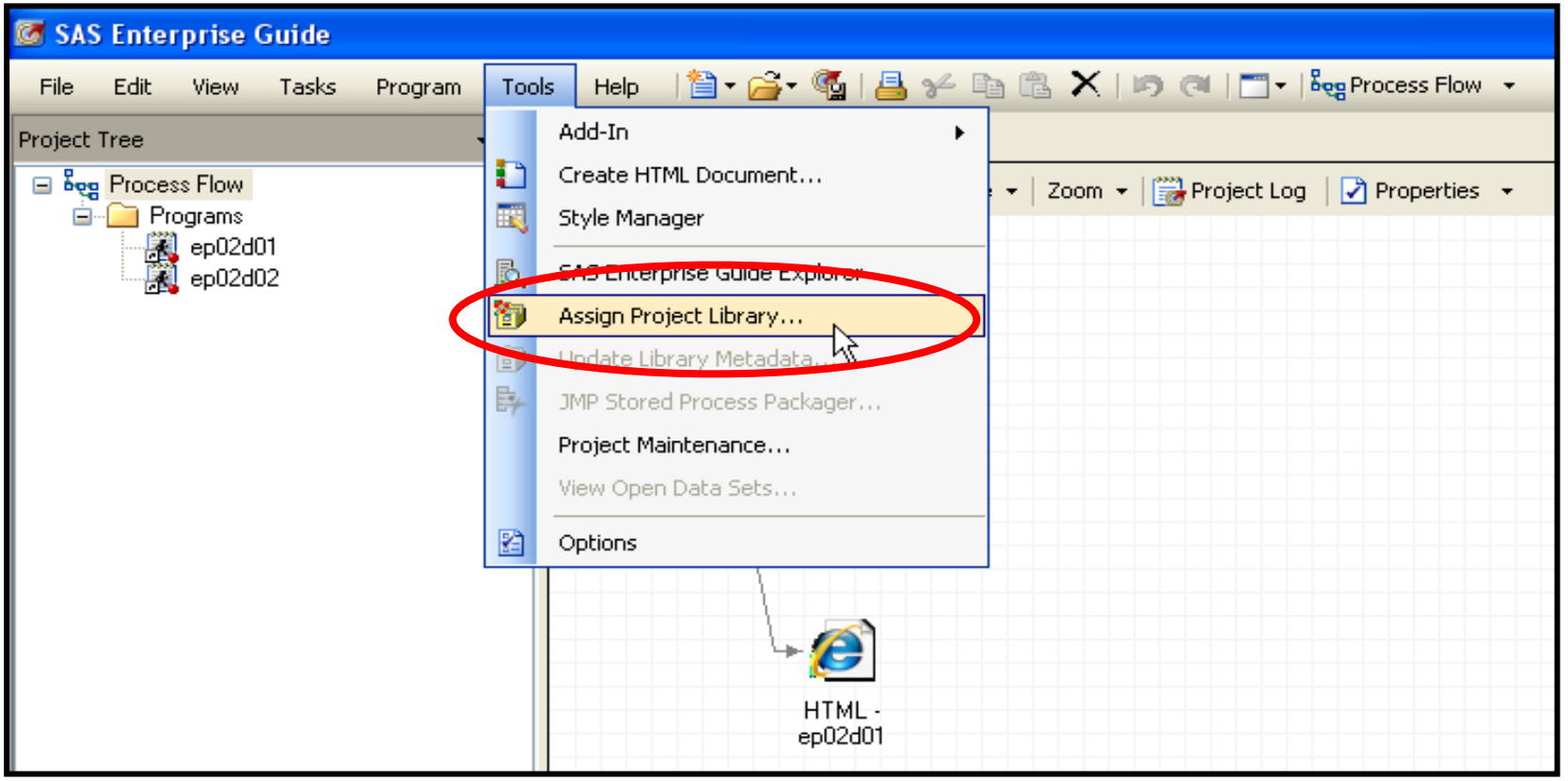


How Do You Include Data in a Project?



Assigning a Libref

- You can use the Assign Project Library task to define a SAS library for an individual project.

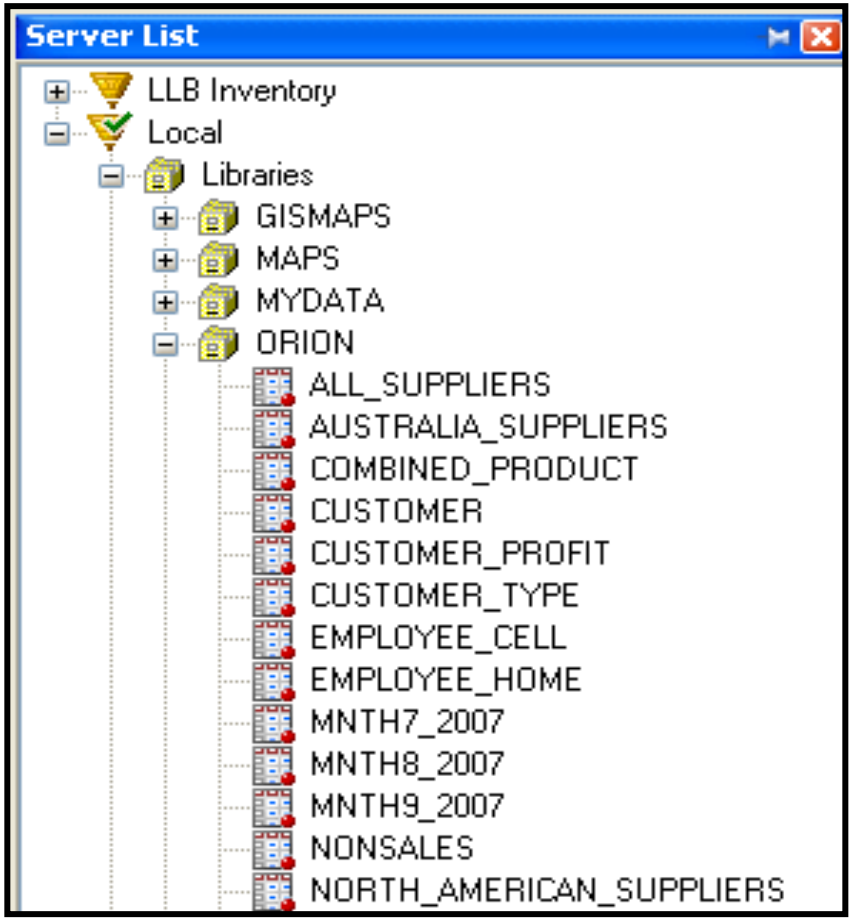


Browsing a SAS Library

- During an interactive SAS Enterprise Guide session, the Server List window enables you to manage your files in the windowing environment.

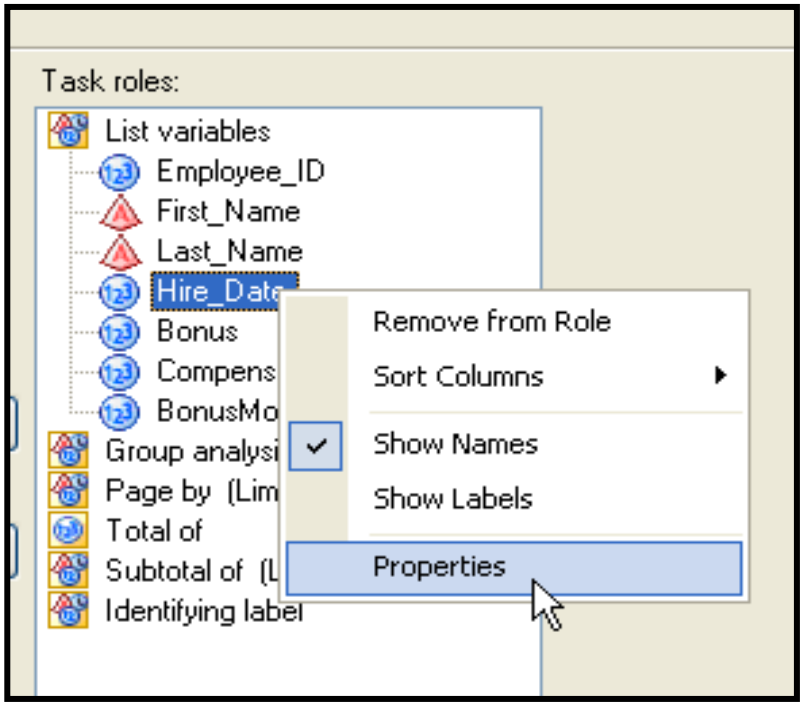
In the Server List window, you can do the following:

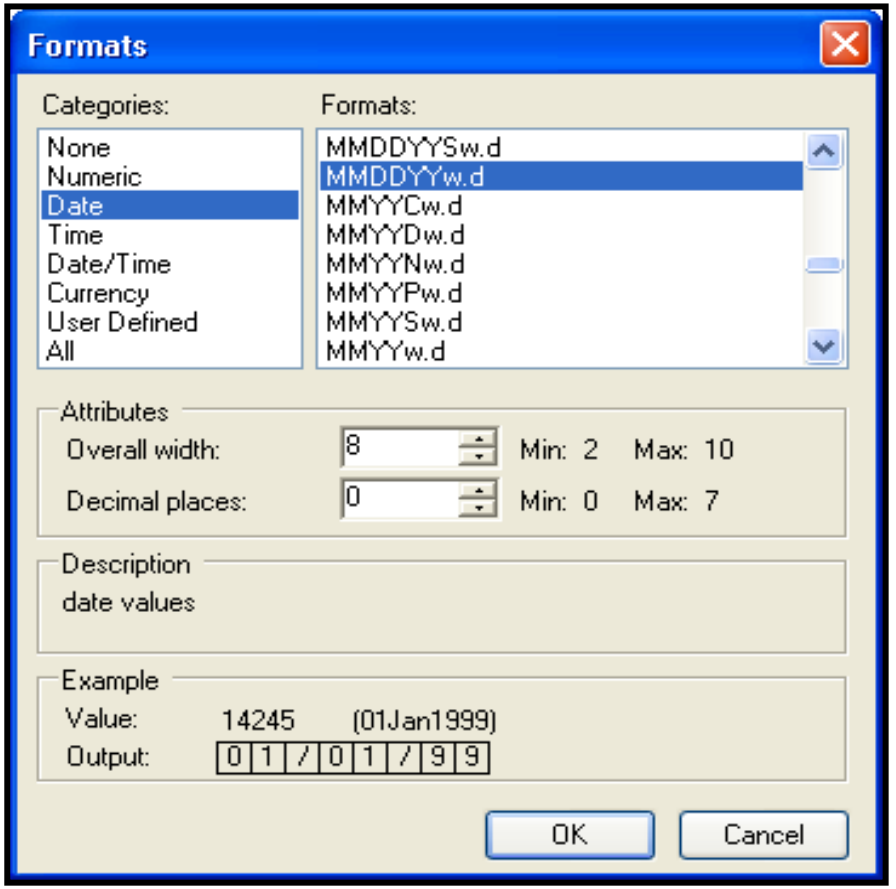
- view a list of all the servers and libraries available during your current SAS Enterprise Guide session
- drill down to see all tables in a specific library
- display the properties of a table
- delete tables
- move tables between libraries



Applying Formats

- Display formats can be applied in a SAS Enterprise Guide task or query by modifying the properties of a variable.

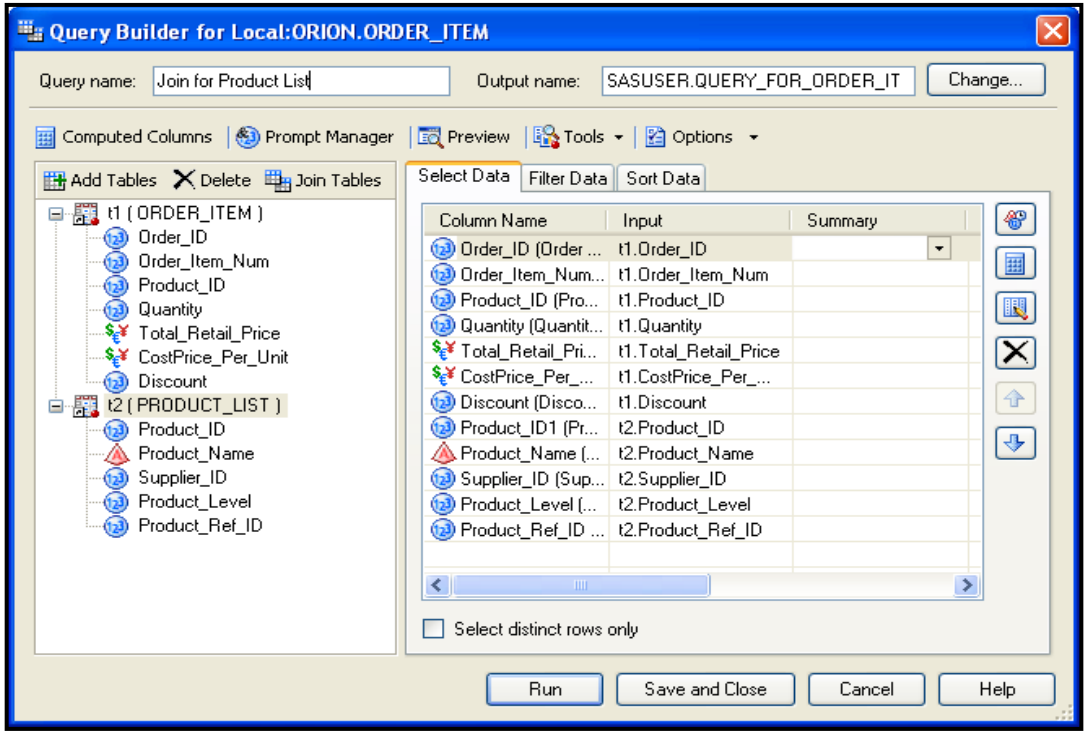




Query Builder Join

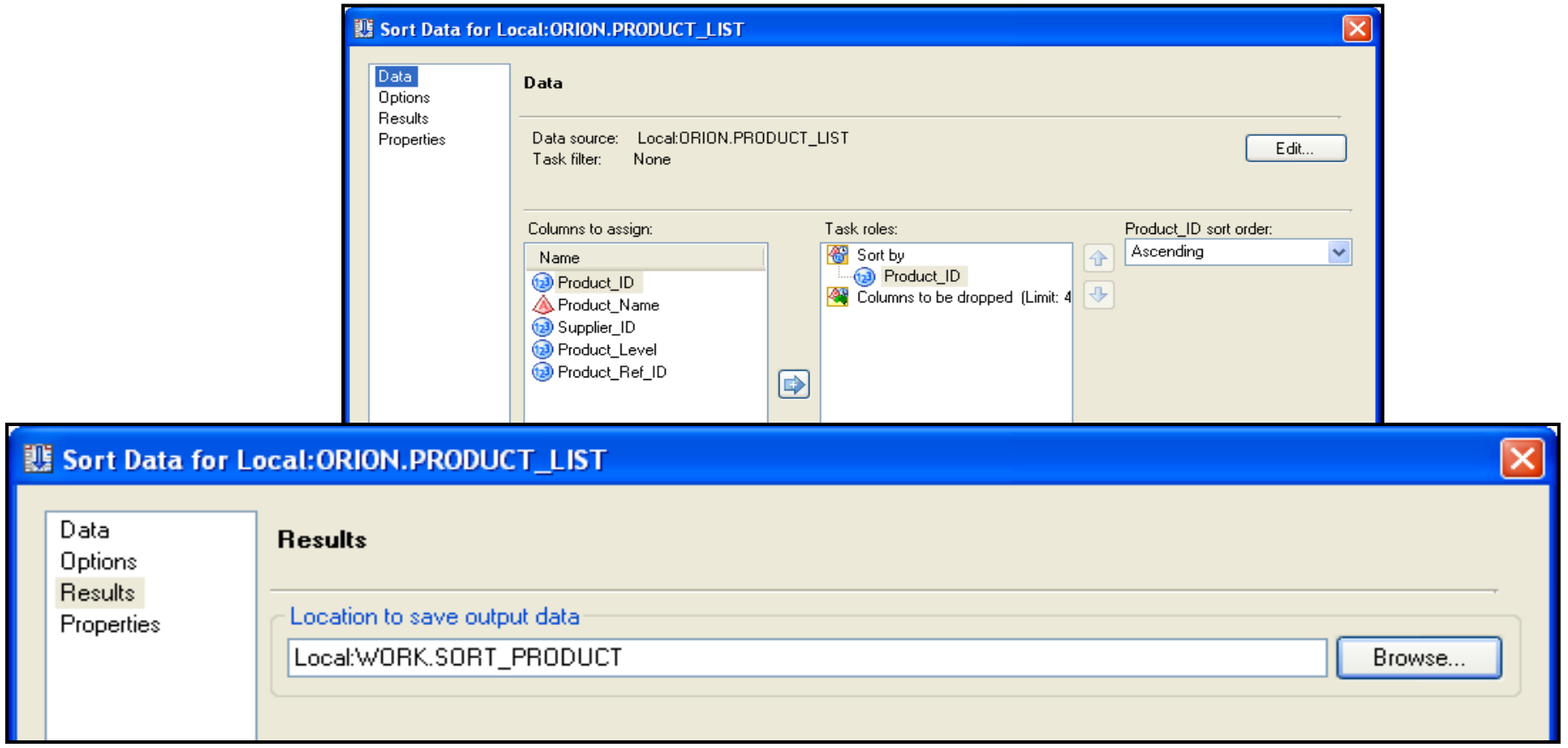
- When you use the Query Builder to join tables in SAS Enterprise Guide, SQL code is generated.

- SQL does not require sorted data.
- SQL can easily join multiple tables on different key variables.
- SQL provides straightforward code
  to join tables
  based on a non-equal comparison of common columns (greater than, less than, between).



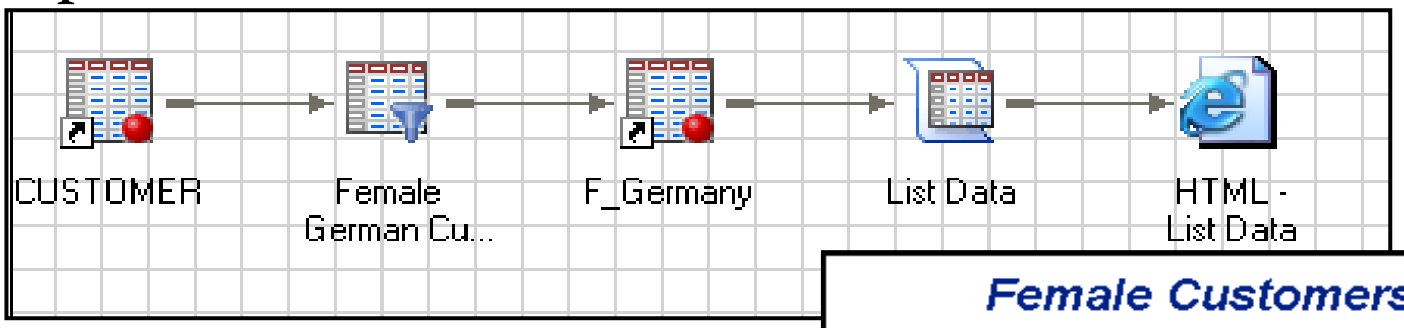
Sort Data Task

- The Sort Data task enables you to create a new data set sorted by one or more variables from the original data.



Business Scenario

- Orion Star wants to send information about a specific promotion to female customers in Germany. The report can be created by querying the **`orion.customer`** data set to include only the desired customers, and then by producing a report with the List Data task.

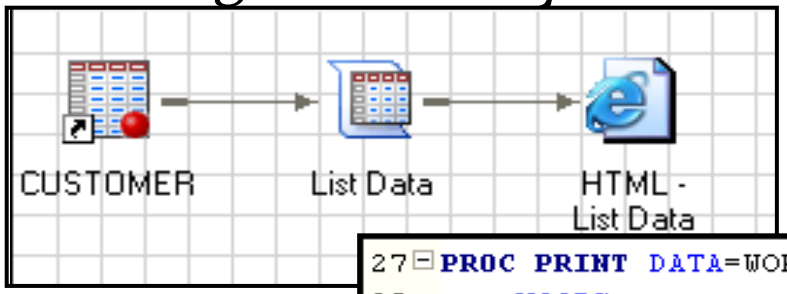


Female Customers in Germany

Customer Country	Customer First Name	Customer Last Name	Customer Birth Date
DE	Cornelia	Krahl	27FEB1974
DE	Elke	Wallstab	16AUG1974
DE	Ines	Deisser	20JUL1969

Business Scenario

- The same report can be generated more efficiently by subsetting the data directly within the List Data task. This requires modification of the code generated by SAS Enterprise Guide.



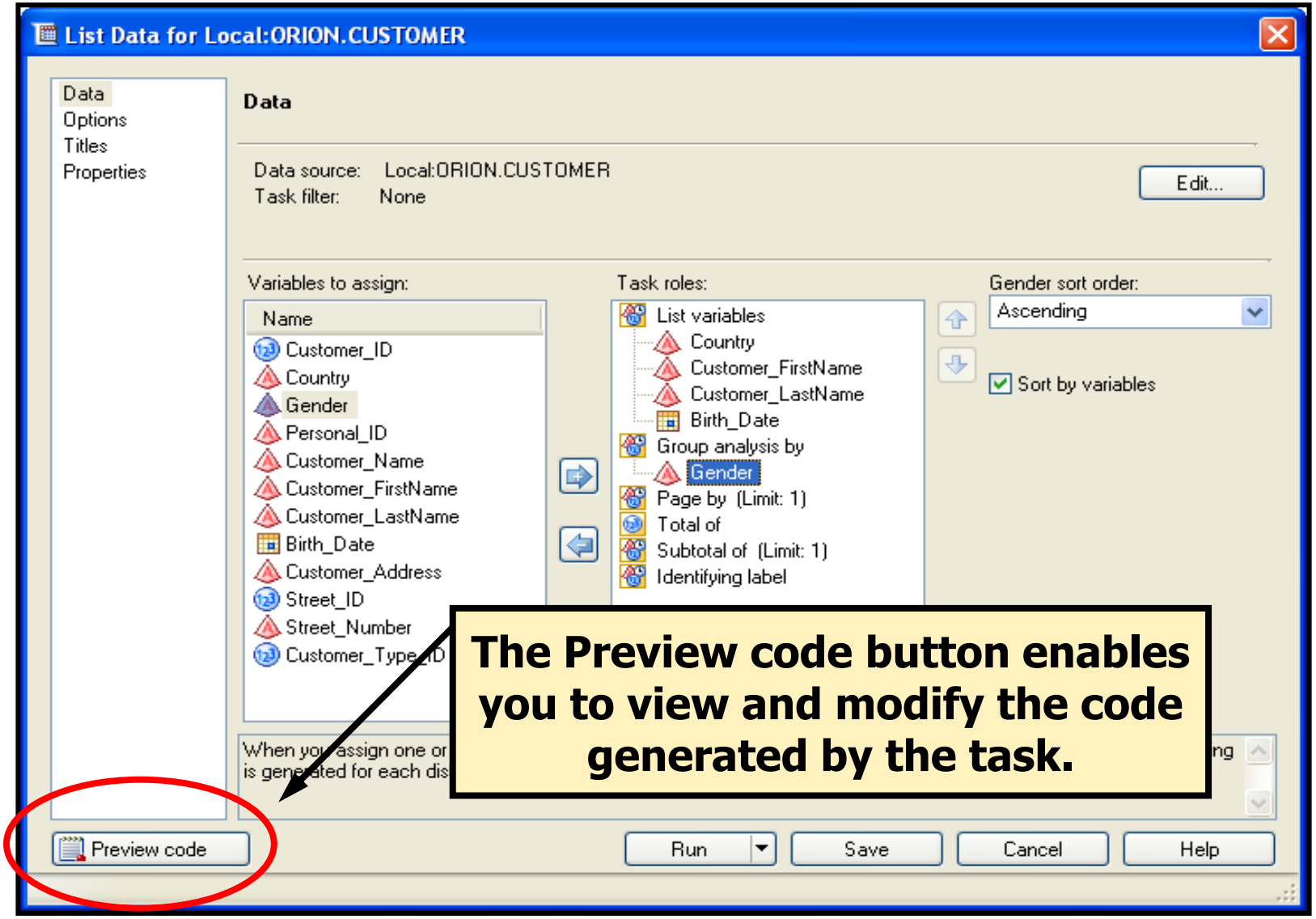

```
27 PROC PRINT DATA=WORK.SORTTempTableSorted
28     NOOBS
29     LABEL
30     ;
31
32 /* Start of custom user code. */
33 where Country = 'DE' and Gender = 'F';
34
35 /* End of custom user code. */
36     VAR Country Customer_FirstName Customer_LastName Birth_Date;
37 RUN;
38 /* ---------------------------------------------------------------
39    End of task code.
40    --------------------------------------------------------------- */
41 RUN; QUIT;
```

Understanding Generated Task Code

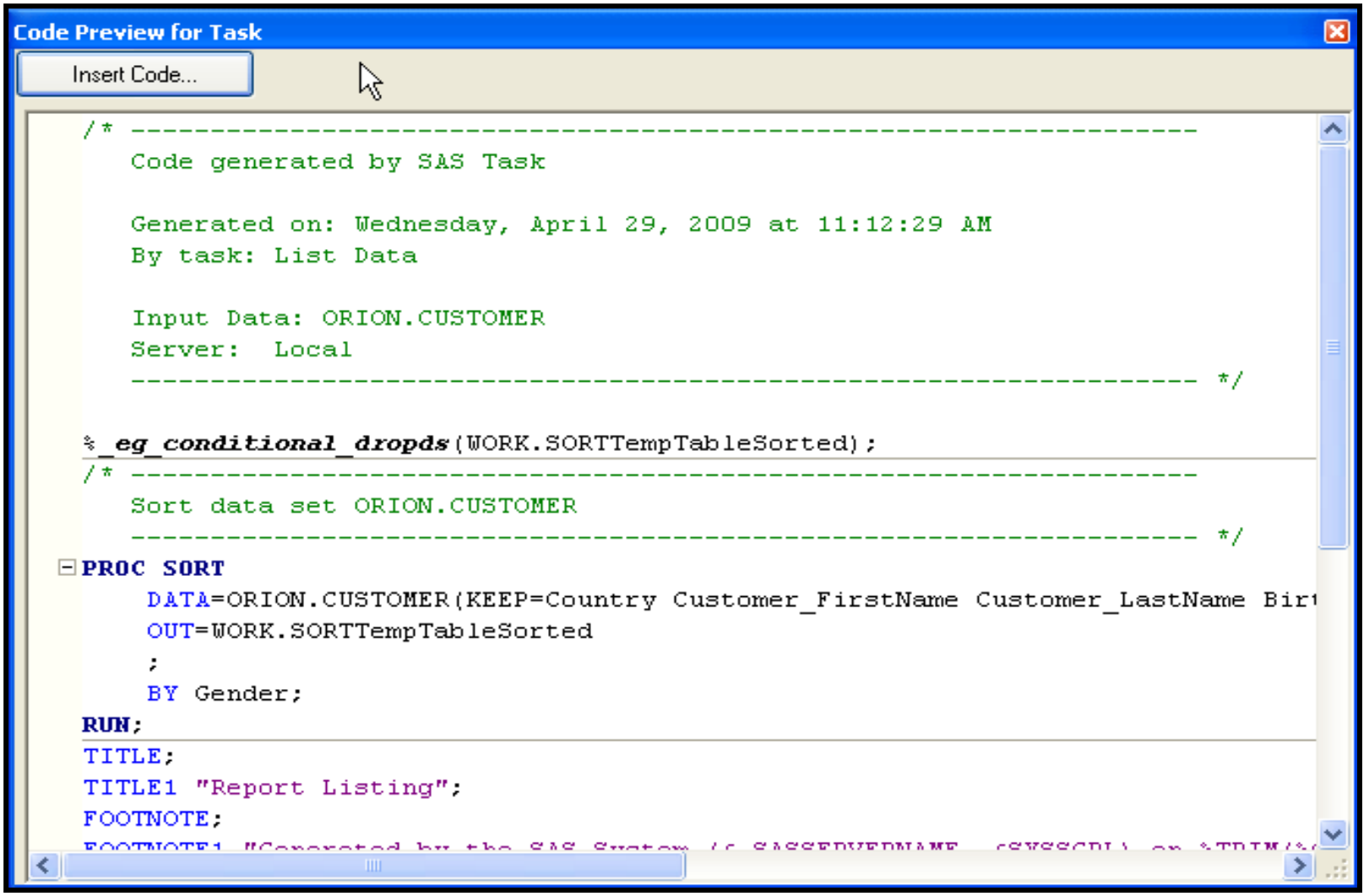
- There are many situations where task results created by SAS Enterprise Guide can be further enhanced or customized by modifying the code.

- However, before you can effectively modify the code, you must first understand the code that SAS Enterprise Guide generates.

List Data Task



List Data Task – Code Preview



Using the List Data Task to Generate Code

- This demonstration illustrates building a List Data task and examining the code generated by SAS Enterprise Guide.

Customer Listing

Customer Gender=F

Customer Country	Customer First Name	Customer Last Name	Customer Birth Date
US	Sandrina	Stephano	09JUL1979
DE	Cornelia	Krahl	27FEB1974
US	Karen	Ballinger	18OCT1984
DE	Elke	Wallstab	16AUG1974

Customer Gender=M

Customer Country	Customer First Name	Customer Last Name	Customer Birth Date
US	James	Kvarniq	27JUN1974
US	David	Black	12APR1969
DE	Markus	Sepke	21JUL1988
DE	Ulrich	Heyde	16JAN1939

List Data Task – Generated Code

- The initial comment block shows information about the task.

```
/* ---------------------------------------------------------------
   Code generated by SAS Task

   Generated on: Wednesday, April 29, 2009 at 1:13:33 PM
   By task: List Data

   Input Data: ORION.CUSTOMER
   Server:  Local
   --------------------------------------------------------------- */
```

List Data Task – Generated Code

- The first line uses a macro to delete temporary tables or views if they already exist. If the Group by role is used in the task, the data must be ordered by the grouping variable. PROC SORT is used by default. Only variables assigned to roles are kept in the new data set.

```
%_eg_conditional_dropds(WORK.SORTTempTableSorted);
/* ----------------------------------------------------------
   Sort data set ORION.CUSTOMER
   ---------------------------------------------------------- */
PROC SORT
     DATA=ORION.CUSTOMER(KEEP=Country Customer_FirstName
                              Customer_LastName Birth_Date
                              Gender)
     OUT=WORK.SORTTempTableSorted
     ;
     BY Gender;
RUN;
```

List Data Task – Generated Code

- If the Group by role is **not** used, SQL creates a temporary view of the required data. Again, only variables assigned to roles in the task are included in the view. This comment i**ncorrectly** states that sorting occurs.

```
%_eg_conditional_dropds(WORK.SORTTempTableSorted);
/* ----------------------------------------------------------------
   Sort data set ORION.CUSTOMER
   ---------------------------------------------------------------- */

PROC SQL;
     CREATE VIEW WORK.SORTTempTableSorted AS
          SELECT T.Country, T.Customer_FirstName,
                 T.Customer_LastName, T.Birth_Date
     FROM ORION.CUSTOMER as T
;
QUIT;
```

List Data Task – Generated Code

- The main part of the code includes the titles, footnotes, and procedure code to generate the report. PROC PRINT is the procedure used with the List Data task.

```
TITLE;
TITLE1 "Customer Listing";
FOOTNOTE;
FOOTNOTE1 "Generated by the SAS System (&_SASSERVERNAME, &SYSSCPL)
           on %TRIM(%QSYSFUNC(DATE(), NLDATE20.))
           at %TRIM(%SYSFUNC(TIME(), NLTIMAP20.))";

PROC PRINT DATA=WORK.SORTTempTableSorted
    NOOBS="Row number"
    LABEL
    ;
    VAR Country Customer_FirstName Customer_LastName Birth_Date;
    BY Gender;
RUN;
```

TITLE and FOOTNOTE are examples of *global* statements and can be included anywhere in a SAS program.

List Data Task – Generated Code

- At the end, the final lines of code delete any temporary tables created to build the task, and delete any assigned titles and footnotes.

```
/* ---------------------------------------------------------
   End of task code.
   --------------------------------------------------------- */
RUN; QUIT;
%_eg_conditional_dropds(WORK.SORTTempTableSorted);
TITLE; FOOTNOTE;
```

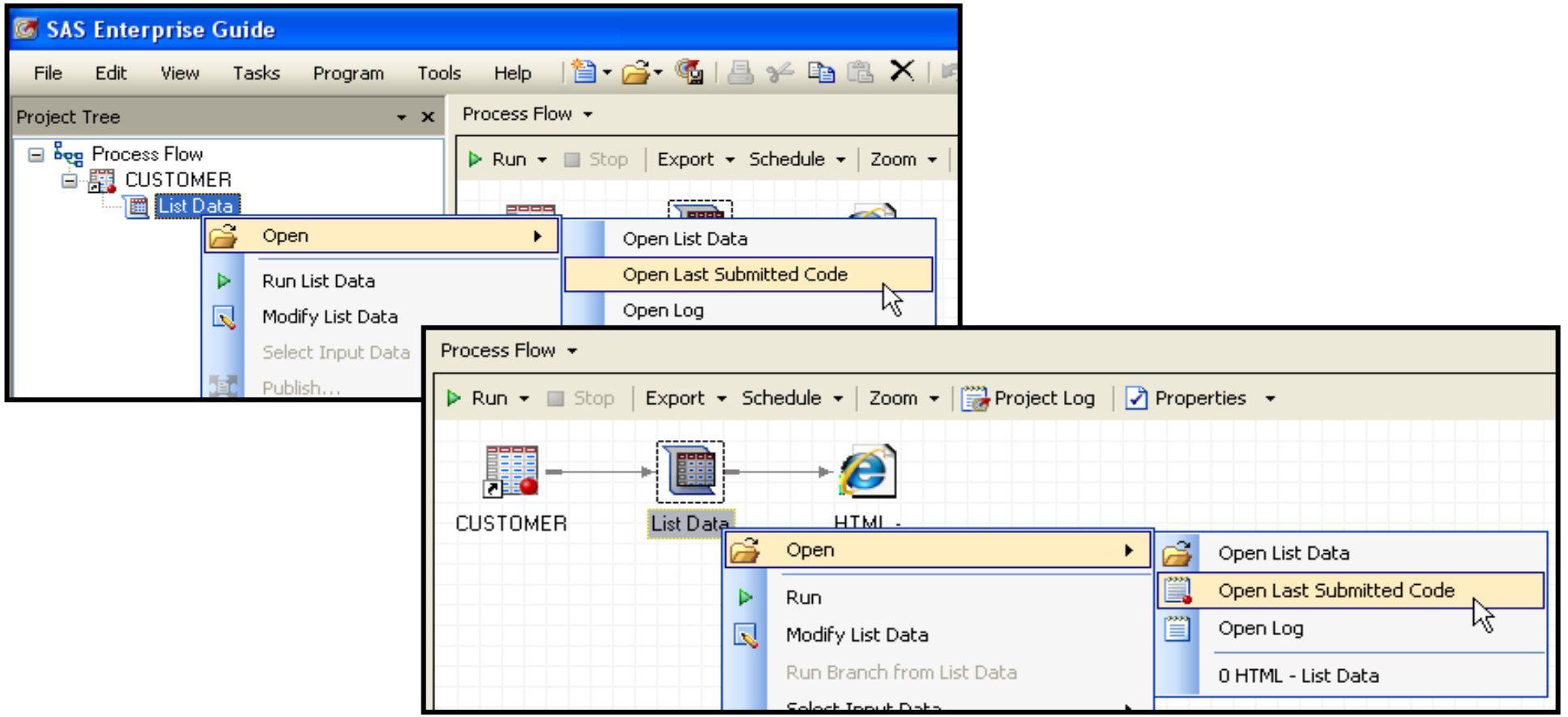
Techniques to Modify Code

- Three methods can be used to modify code generated by SAS Enterprise Guide:

1. Edit the last submitted task code in a separate Code window.
2. Automatically submit custom code before or after every task and query.
3. Insert custom code in a task.

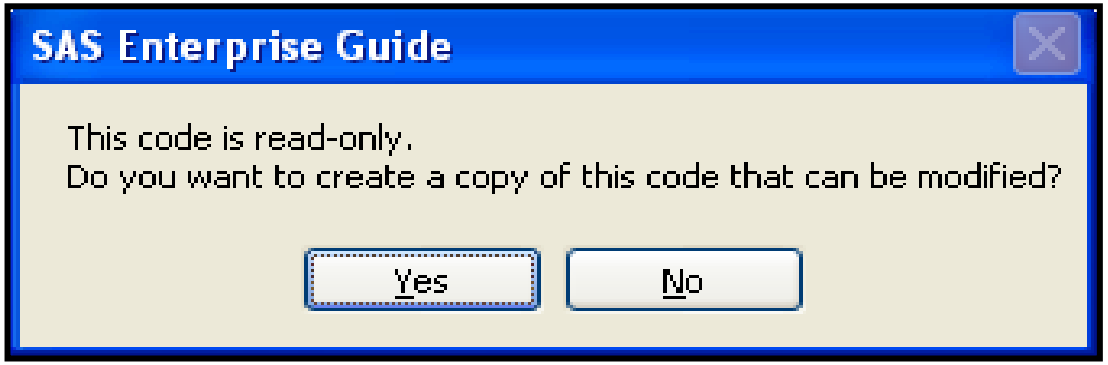
Edit Last Submitted Code

- After a task runs, the code can be viewed from either the Project Tree or Process Flow.



Edit Last Submitted Code

The task code is read-only and cannot be edited directly. To create a copy of the code from the Last Submitted Code window, select any key while in the SAS program window. SAS Enterprise Guide offers to make a copy.



After the code is copied, there is no link between the task and the new code. Any changes in the task are not reflected in the copied code, and modifications to the code do not affect the task.

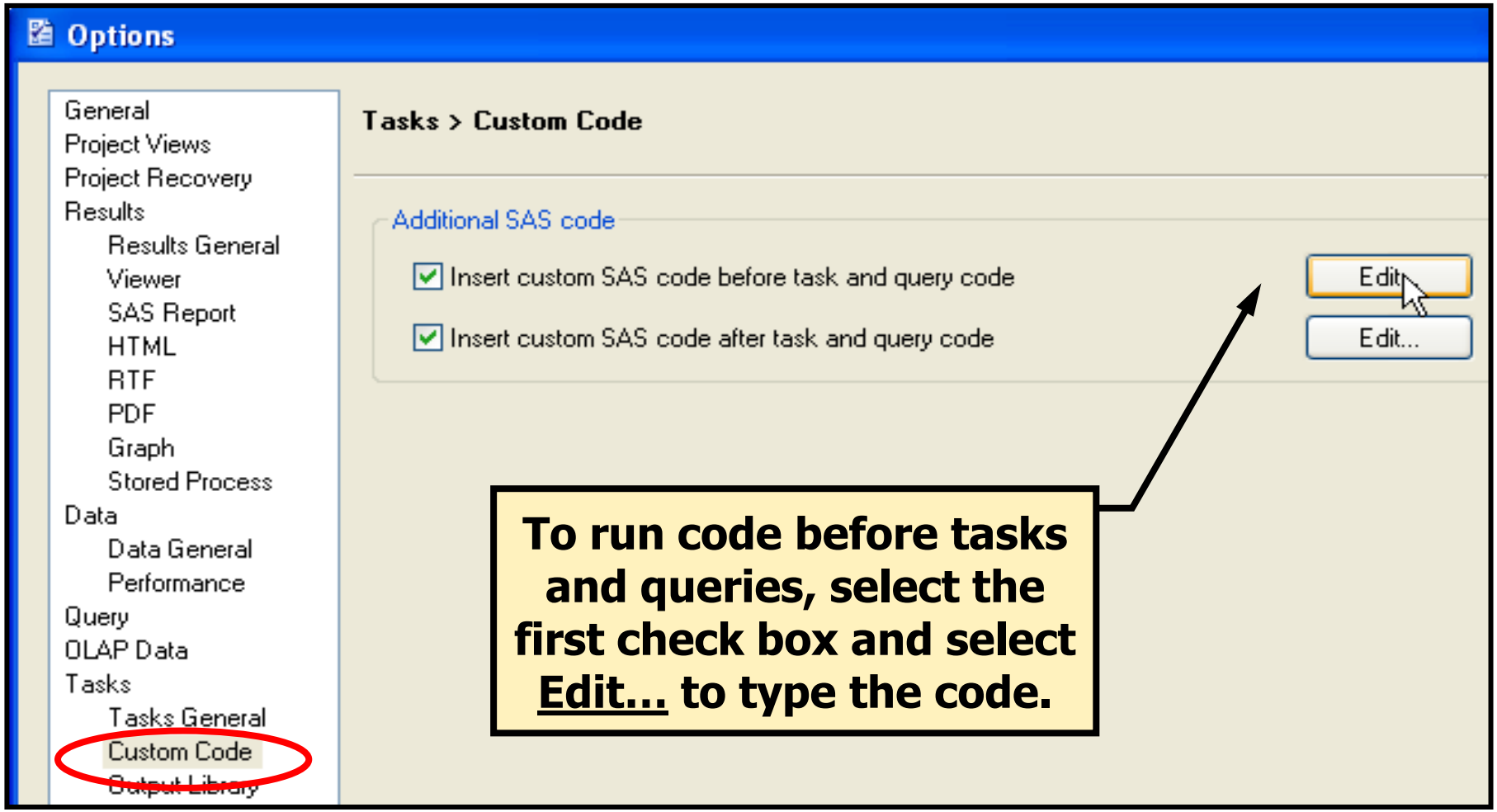
Summary of Editing Last Submitted Code

Custom code linked to task?	No
Can be used to modify query code?	Yes
Extent of modification allowed?	Anything in the program can be changed.
Custom code included when exported?	Yes. You **must** export the edited program and select the option in the Export wizard.

Automatically Submit Custom Code Before or After Every Task and Query

- There are times when you might need to run a SAS statement or program before or after any task or query is executed. The Custom Code option enables you to insert custom code before or after all tasks and queries.

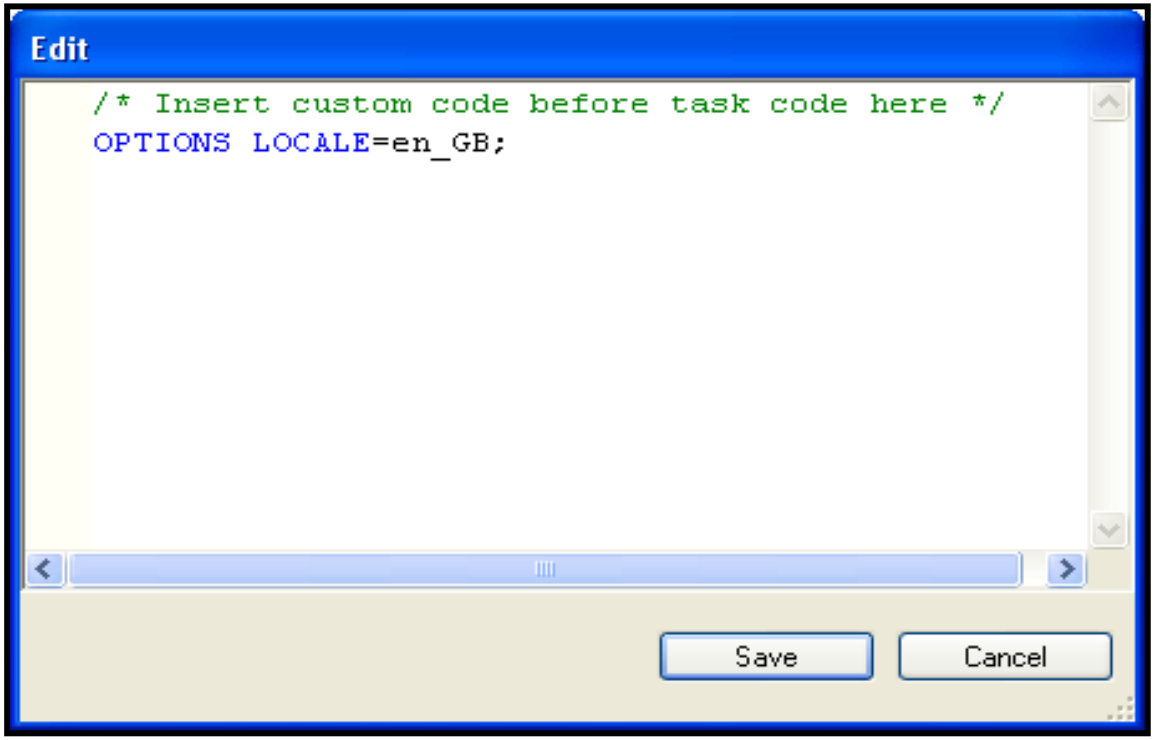
Automatically Submit Custom Code Before or After Every Task and Query



Automatically Submit Custom Code Before or After Every Task and Query

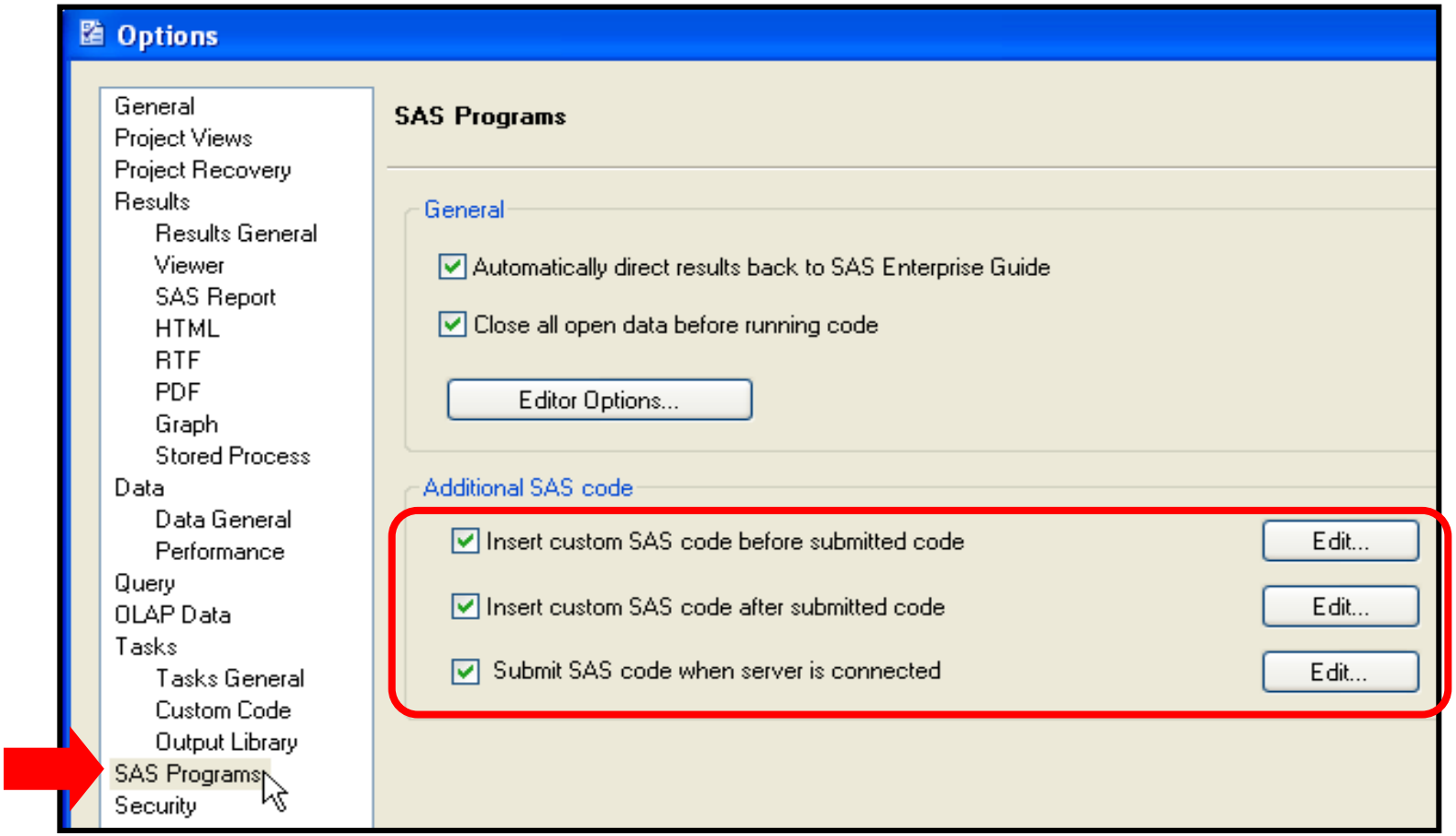
Global statements or complete program steps can be entered.

Example: Set the LOCALE= option to Great Britain.



Insert Code Before or After SAS Programs

- Similar options exist to automatically submit code before or after SAS programs written and submitted in Code windows in SAS Enterprise Guide.

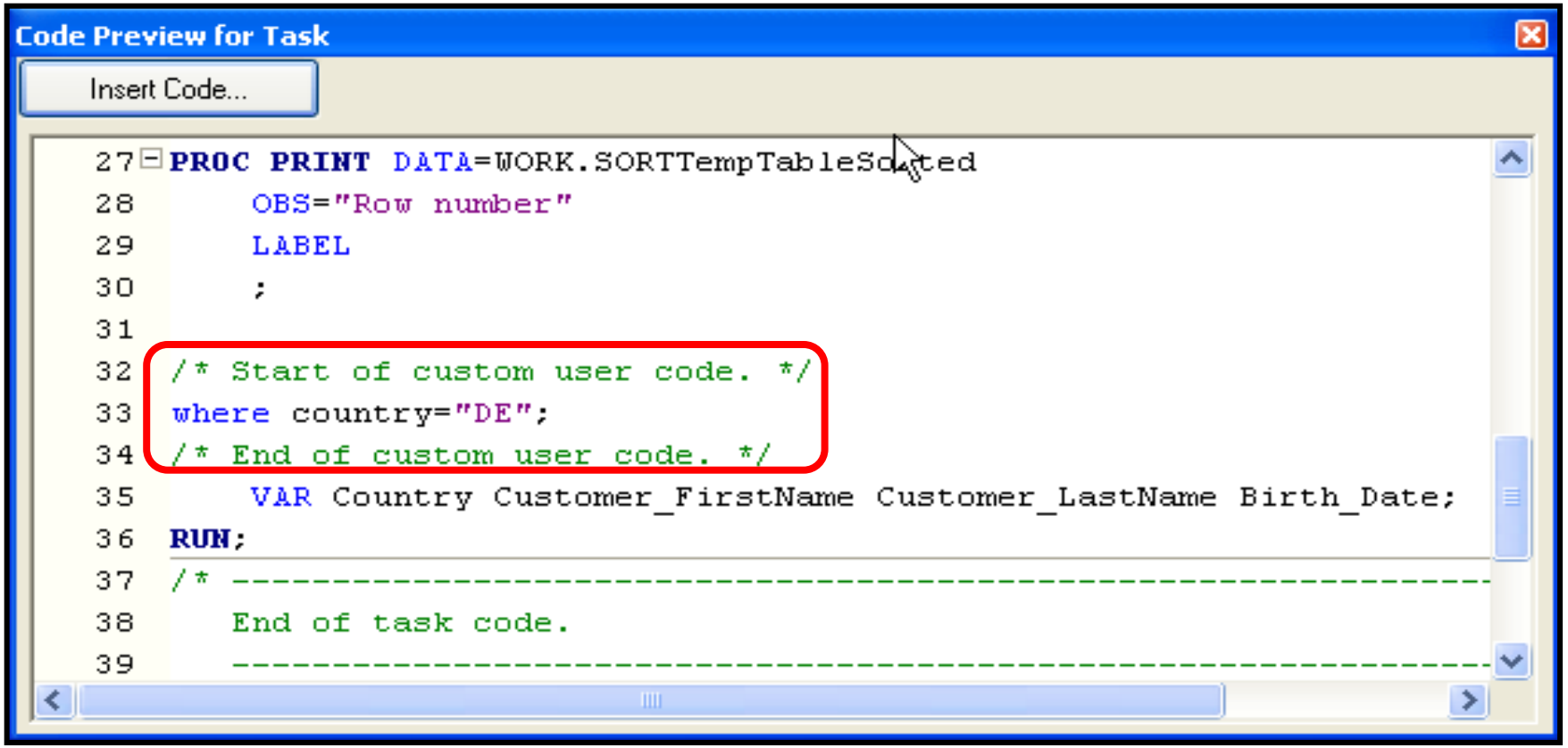


Summary of Submitting Custom Code Before or After Every Task and Query

Custom code linked to task?	Yes
Can be used to modify query code?	Yes
Extent of modification allowed?	Statements can only be submitted before or after the task code.
Custom code included when exported?	Yes, select the option in the Export wizard.

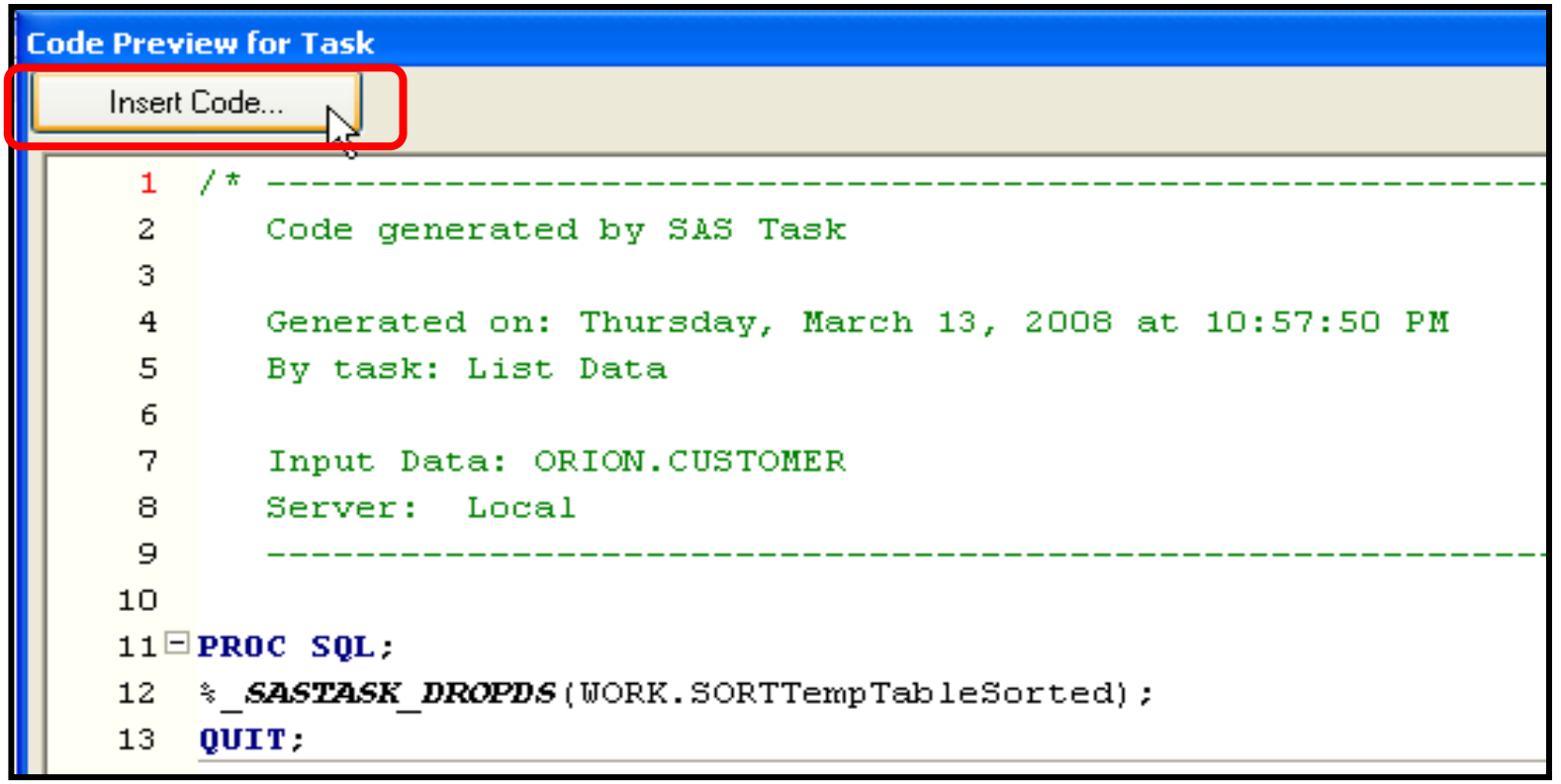
Insert Custom Code in a Task

- In most task dialog boxes, you have the ability to insert custom code within the generated SAS program. This technique has the significant benefit that the task interface can still be used to modify the report.

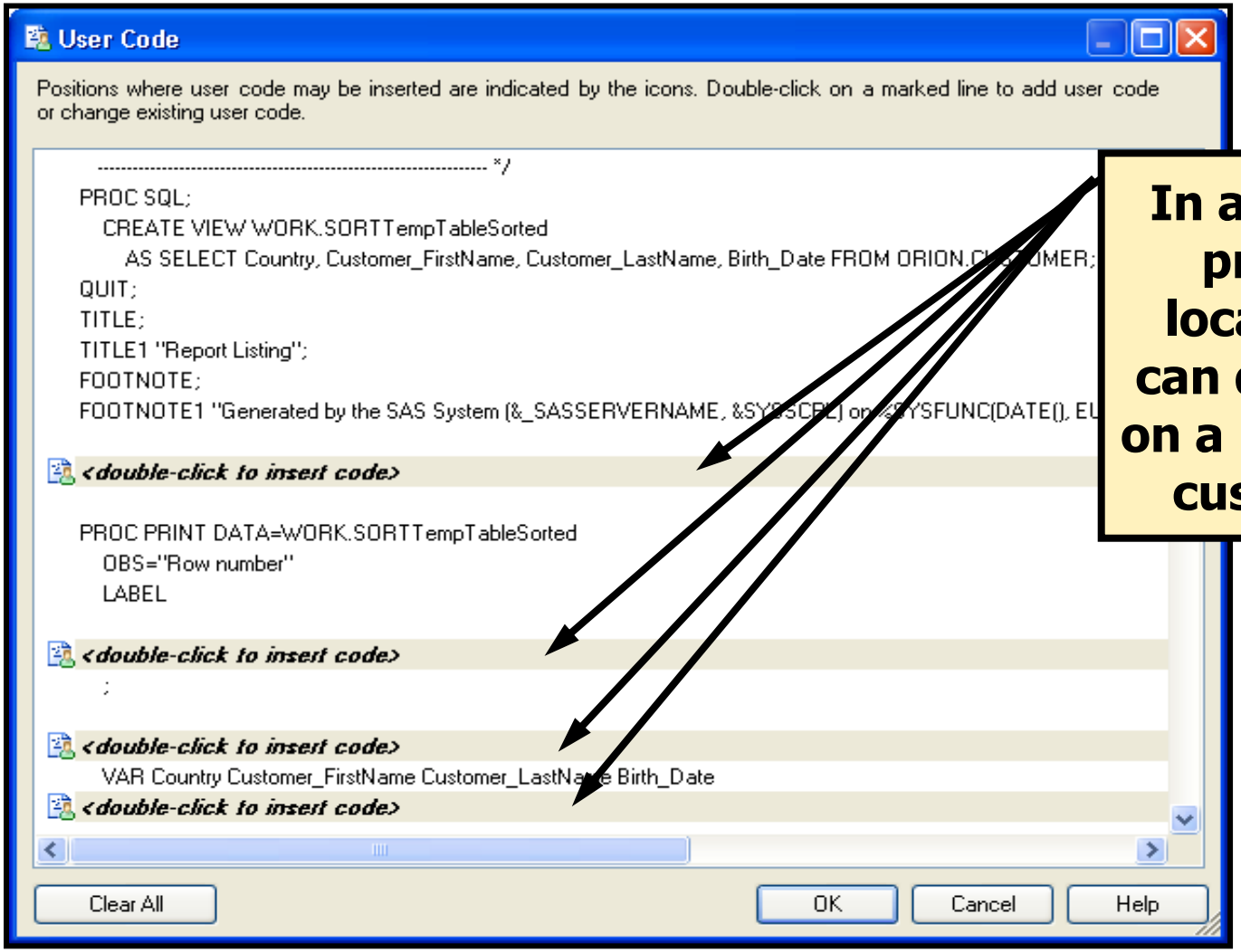


Insert Custom Code in a Task

- In the Code Preview window, select **Insert Code...** to add custom code in predefined locations in the SAS program.



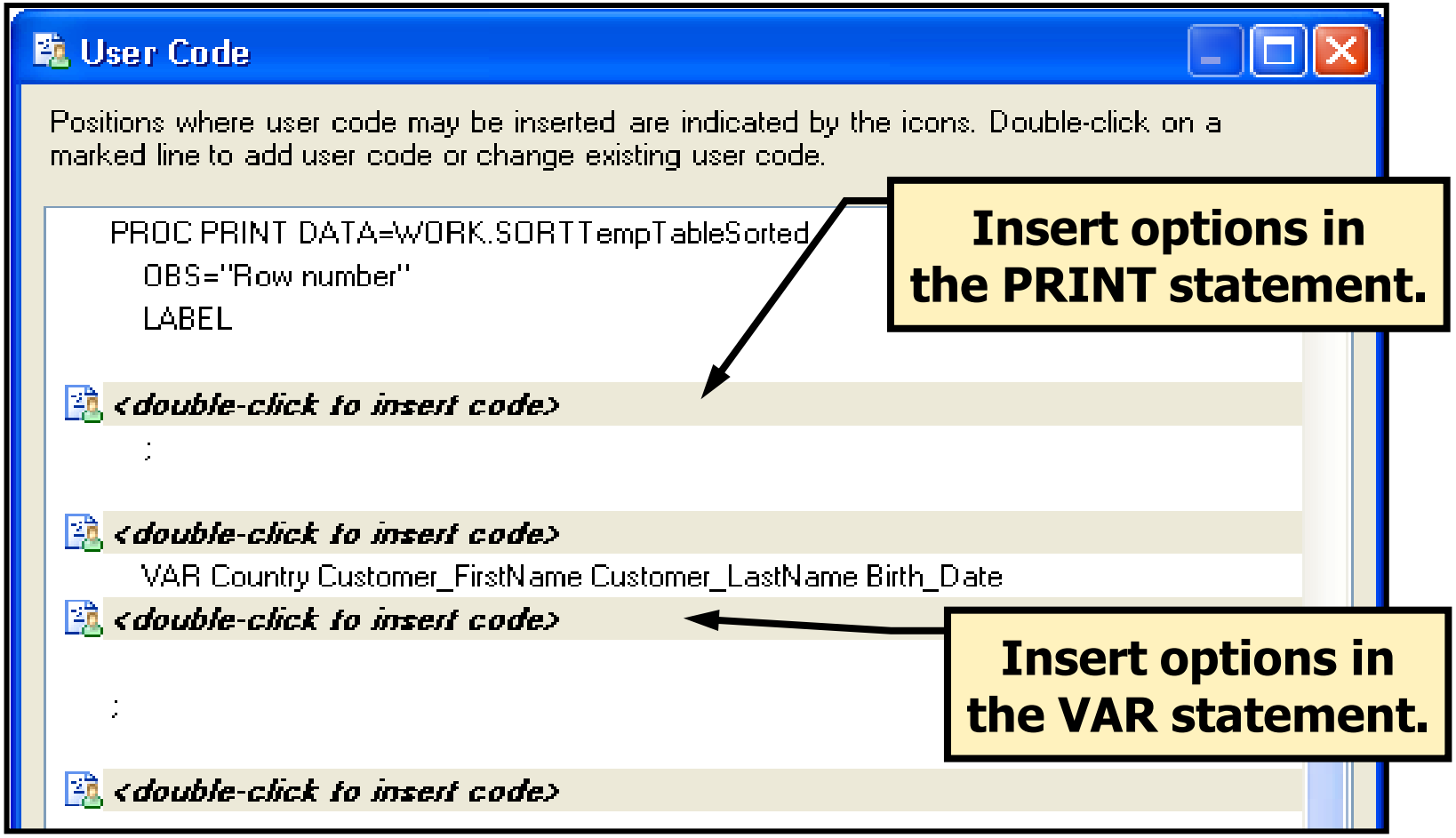
Insert Custom Code in a Task



In any of these predefined locations, you can double-click on a line to insert custom code.

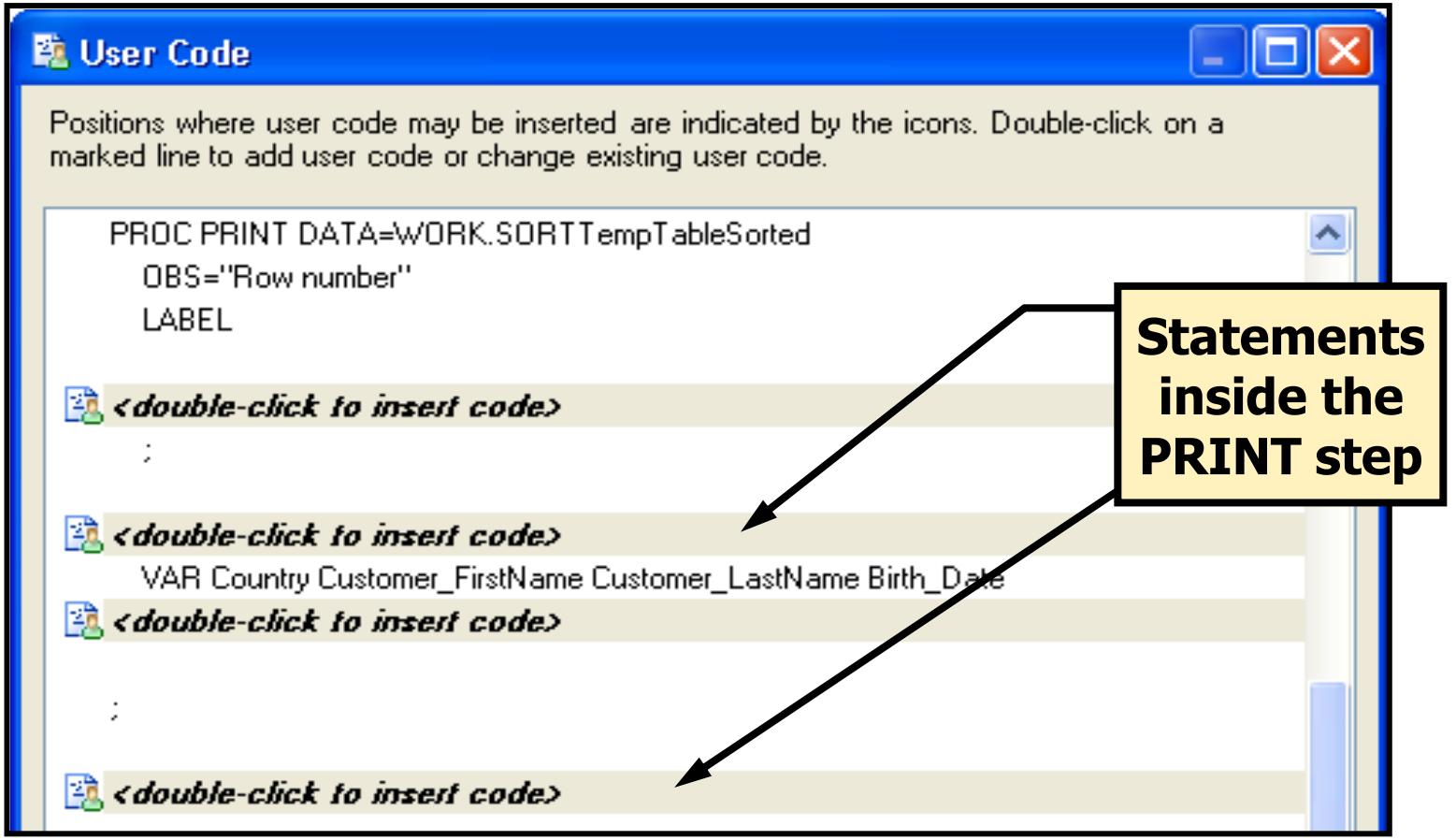
Insert Custom Code in a Task

- Some insert points enable custom options to be added to existing statements.



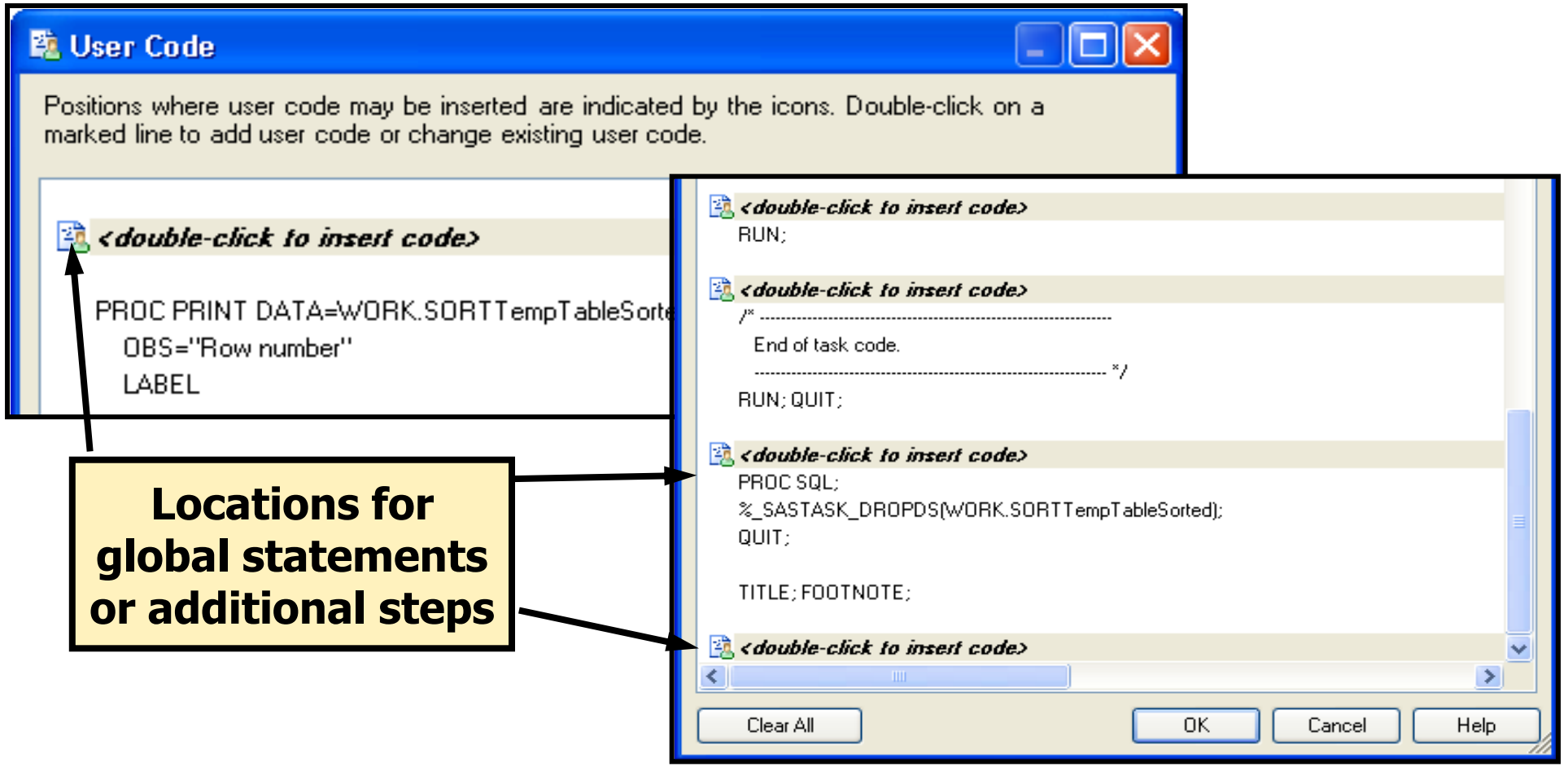
Insert Custom Code in a Task

- Other insert points enable entire statements to be added inside a step in the program.

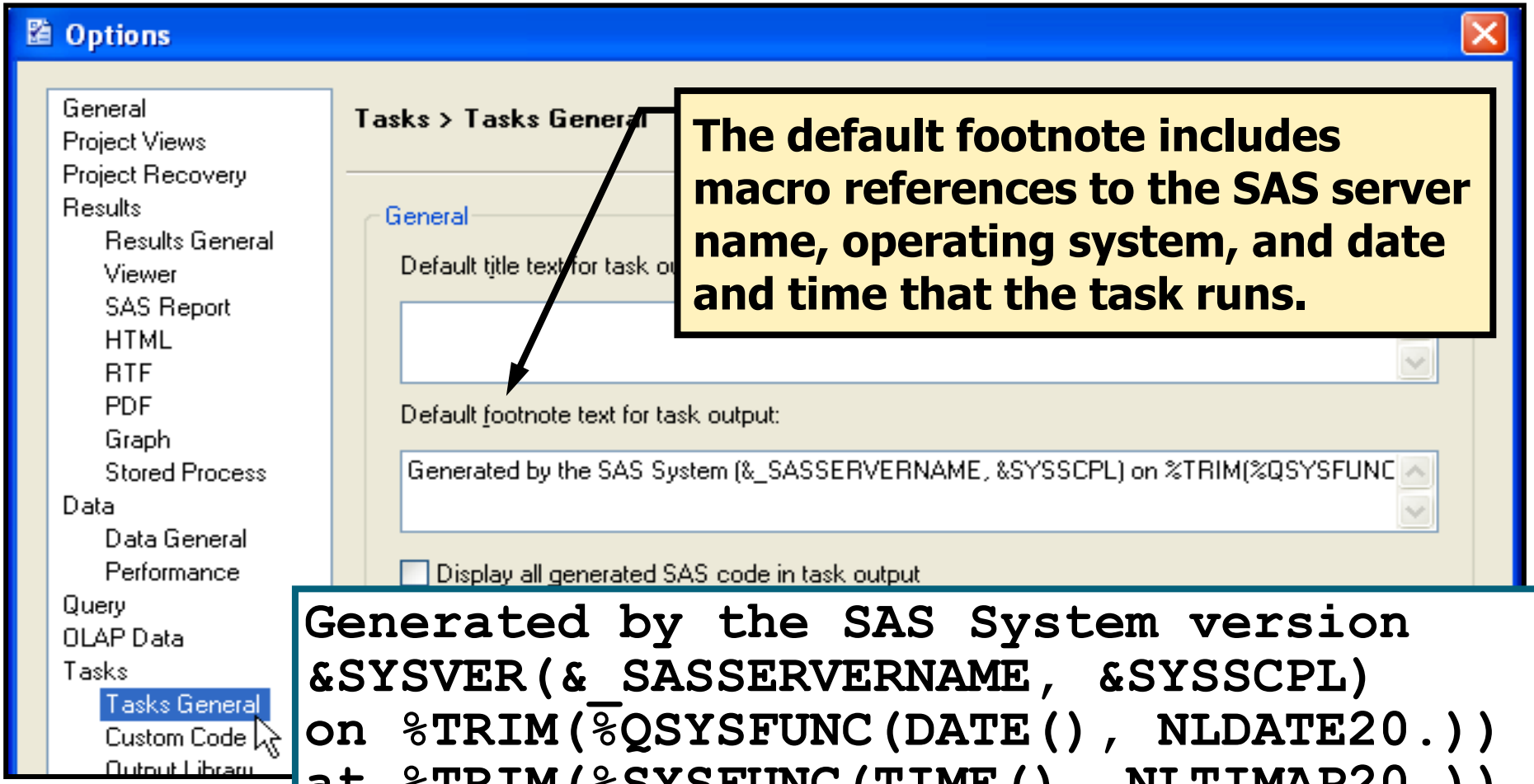


Insert Custom Code in a Task

- Additional locations enable global statements or additional steps to be inserted before or after the main code.

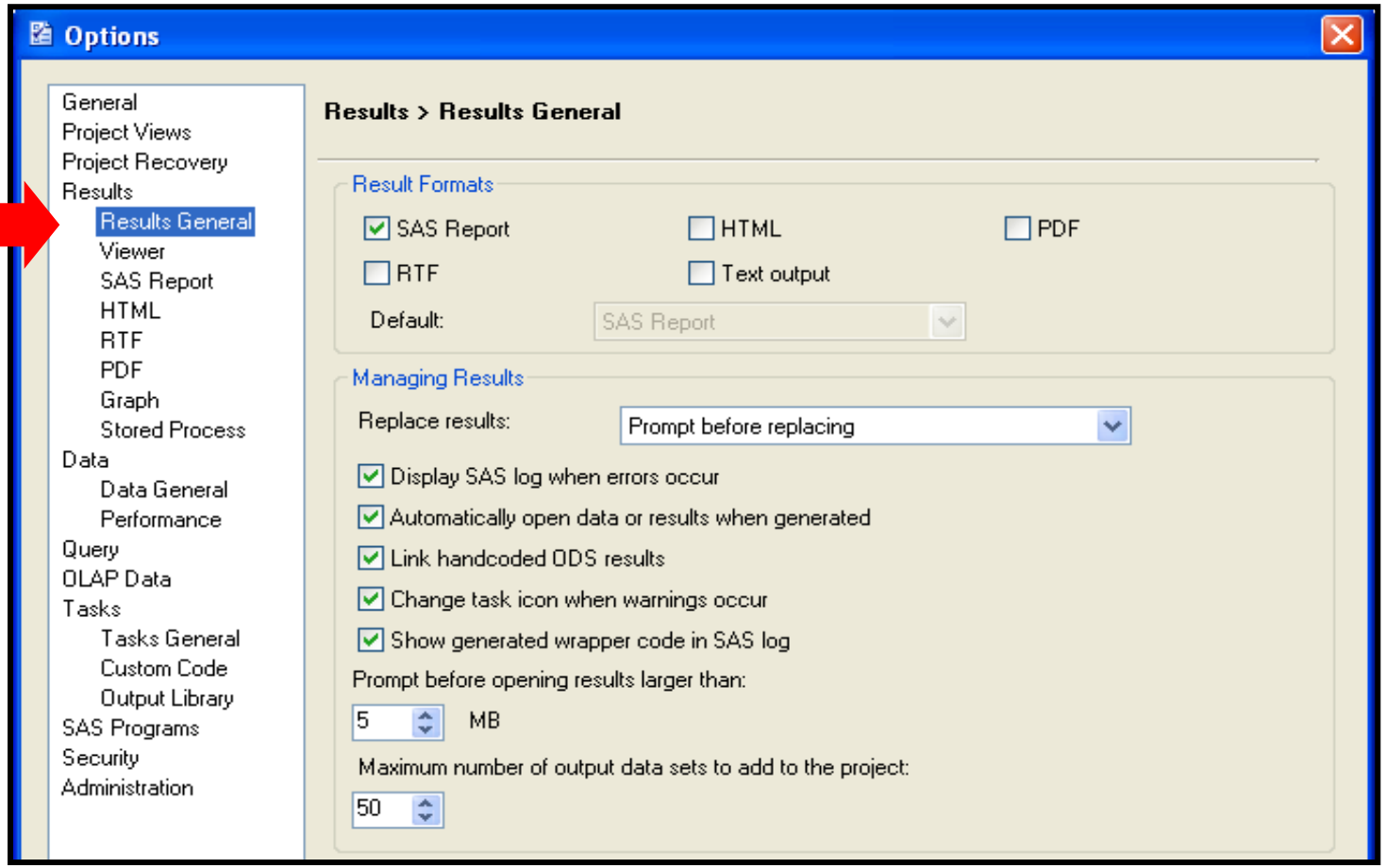


Default SAS Enterprise Guide Footnote



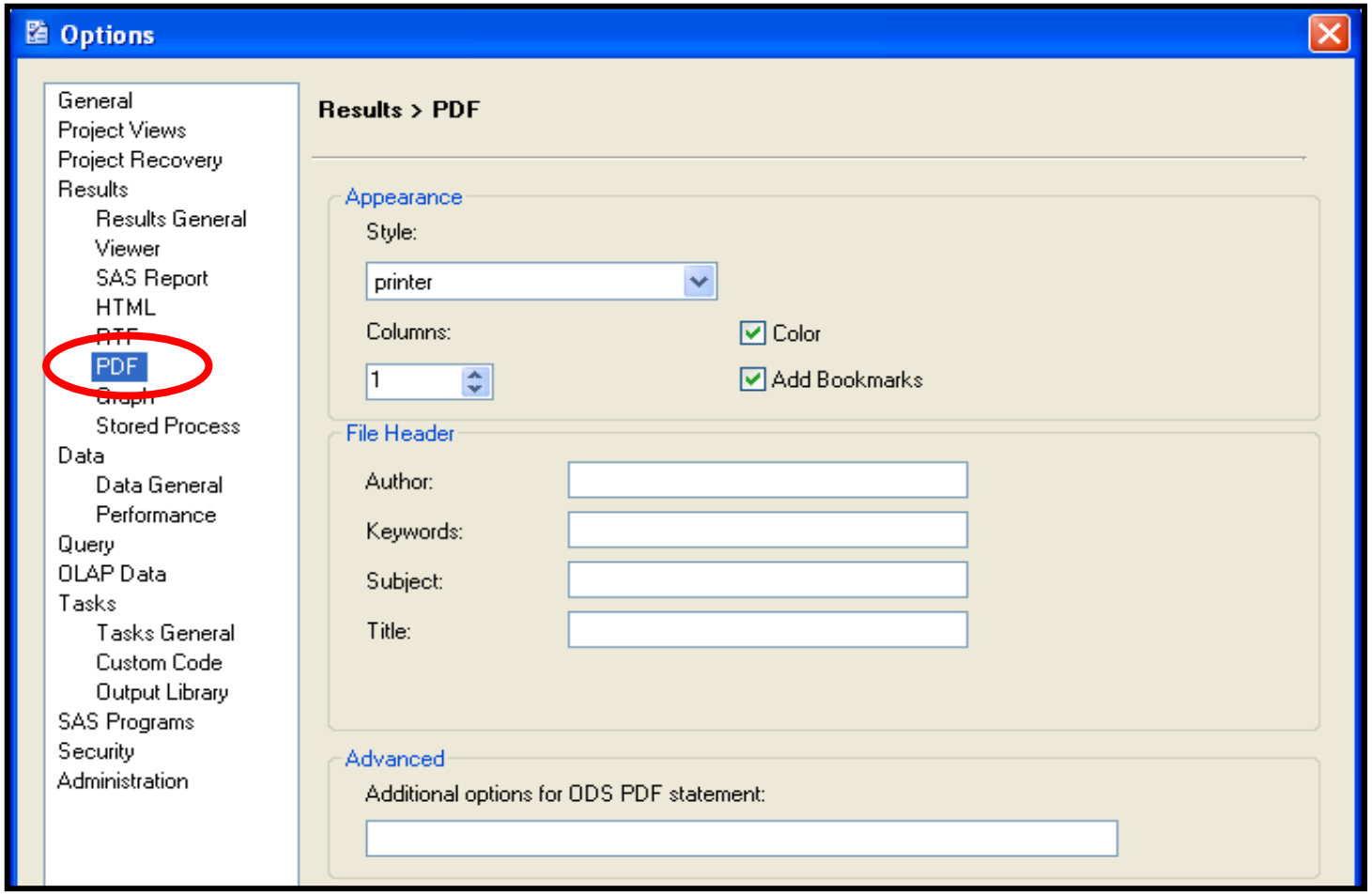
ODS and SAS Enterprise Guide

- Default result formats can be set under **Tools** ⇨ **Options**.



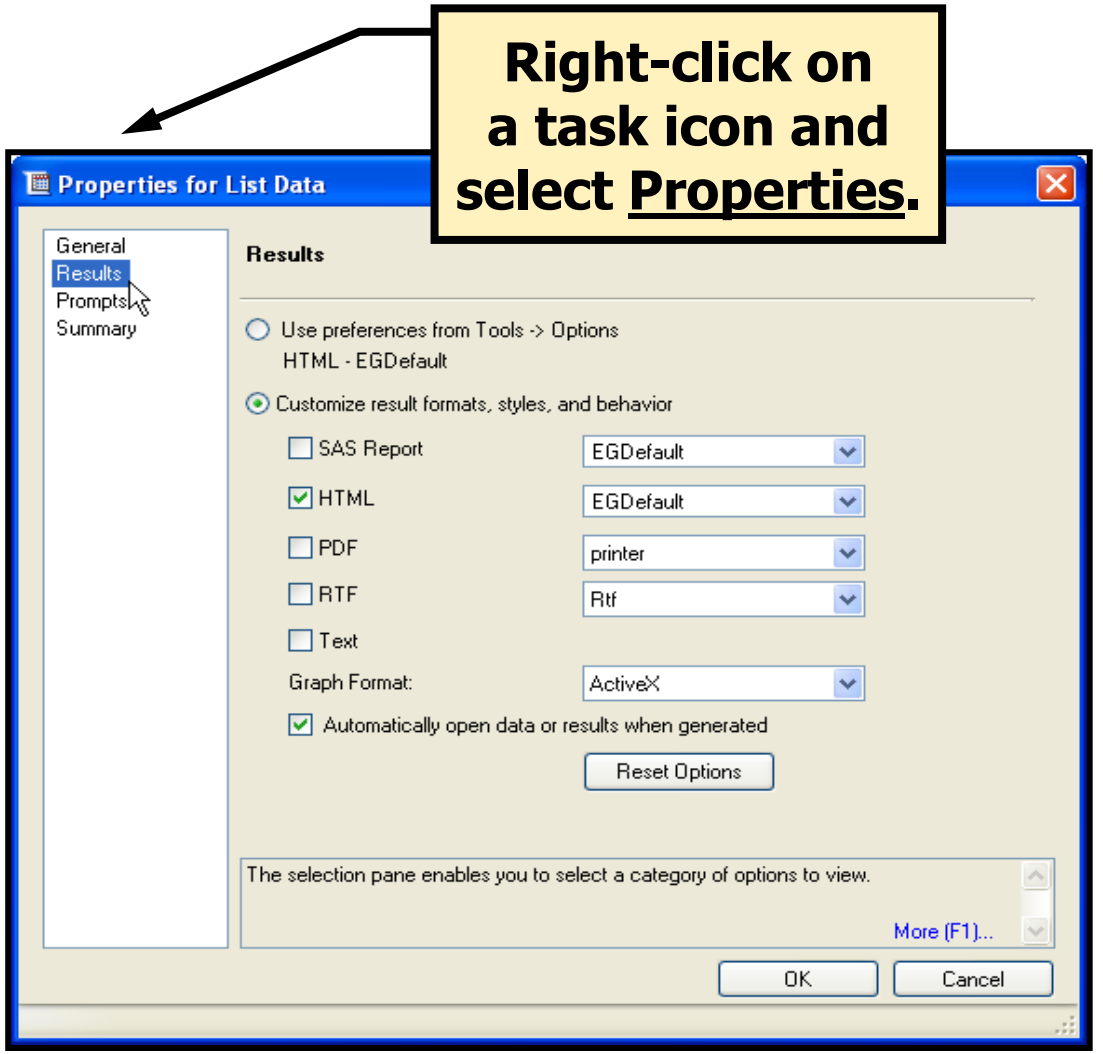
ODS and SAS Enterprise Guide

- Additional settings can be made for each result format.



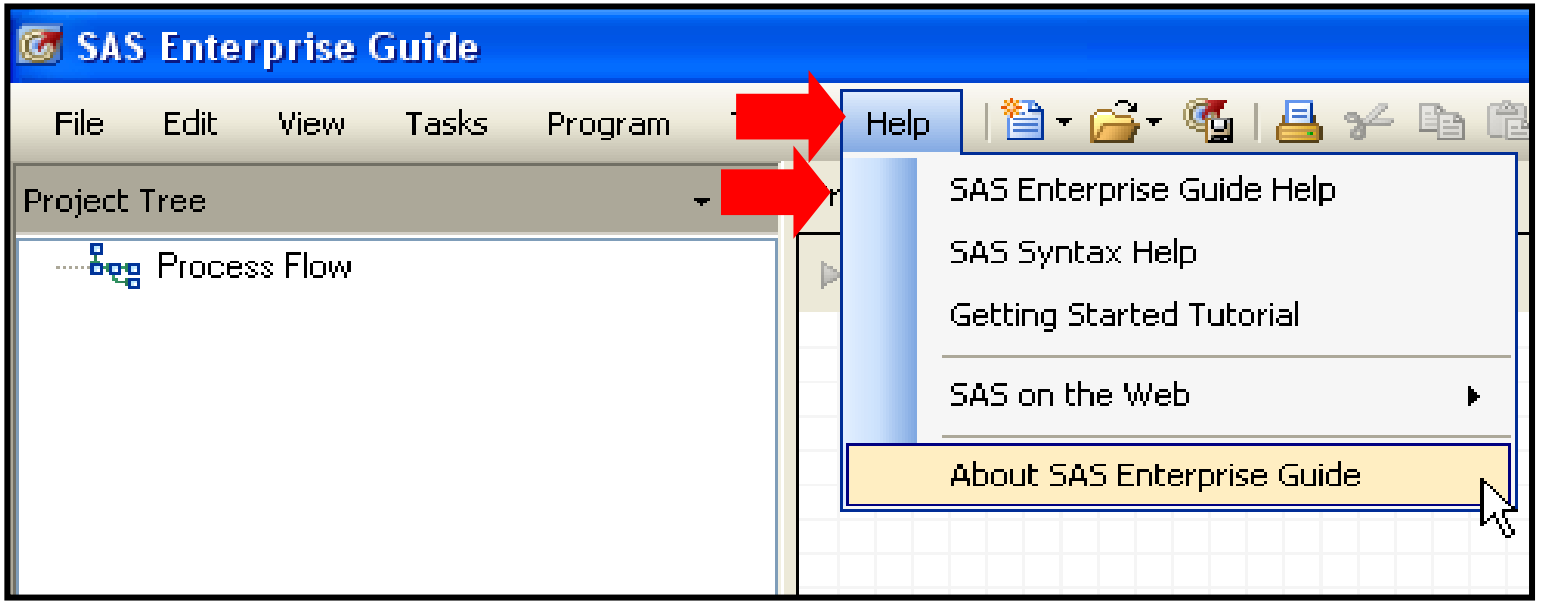
ODS and SAS Enterprise Guide

- Task properties can be used to override the default for an individual task.
- Generated output can be switched off completely and handled by inserting code.

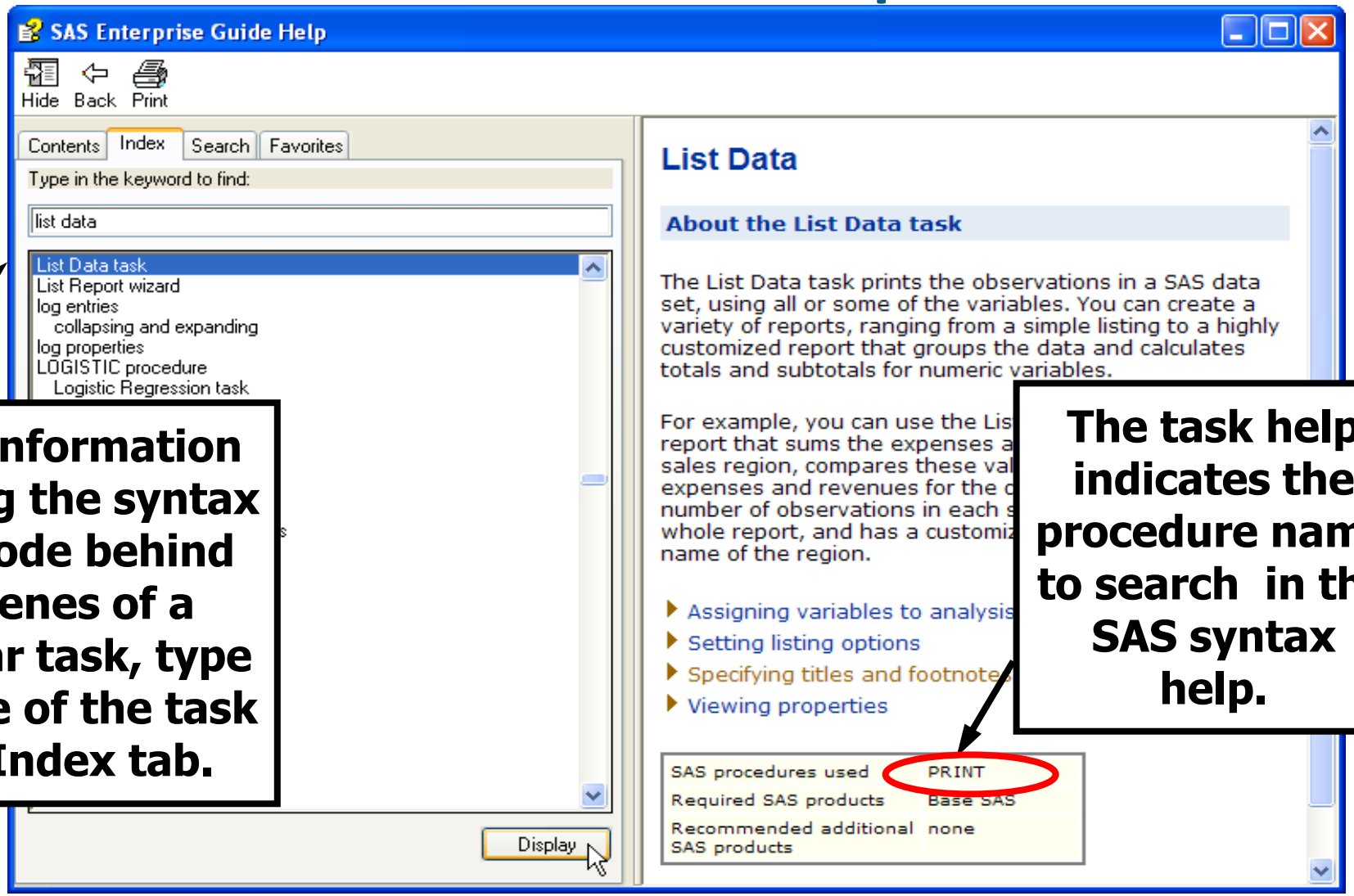


SAS Enterprise Guide Help (Review)

- If Help files were installed along with SAS Enterprise Guide, you can select **Help** to access the Help facility regarding both the point-and-click functionality of SAS Enterprise Guide as well as SAS syntax.



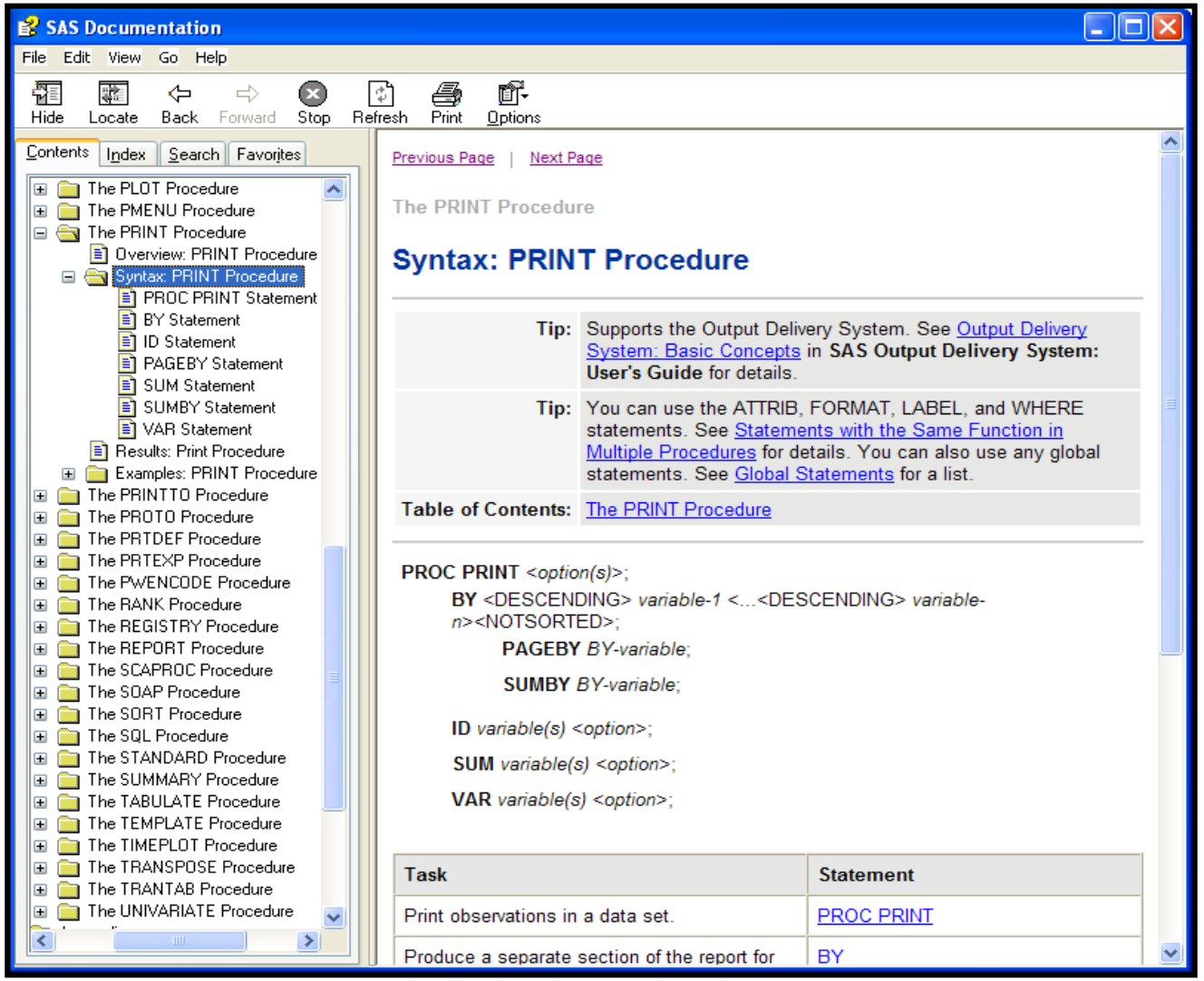
Task and Procedure Help



To find information regarding the syntax of the code behind the scenes of a particular task, type the name of the task in the Index tab.

The task help indicates the procedure name to search in the SAS syntax help.

Procedure Syntax Help



2. Pokročilé programování v SAS

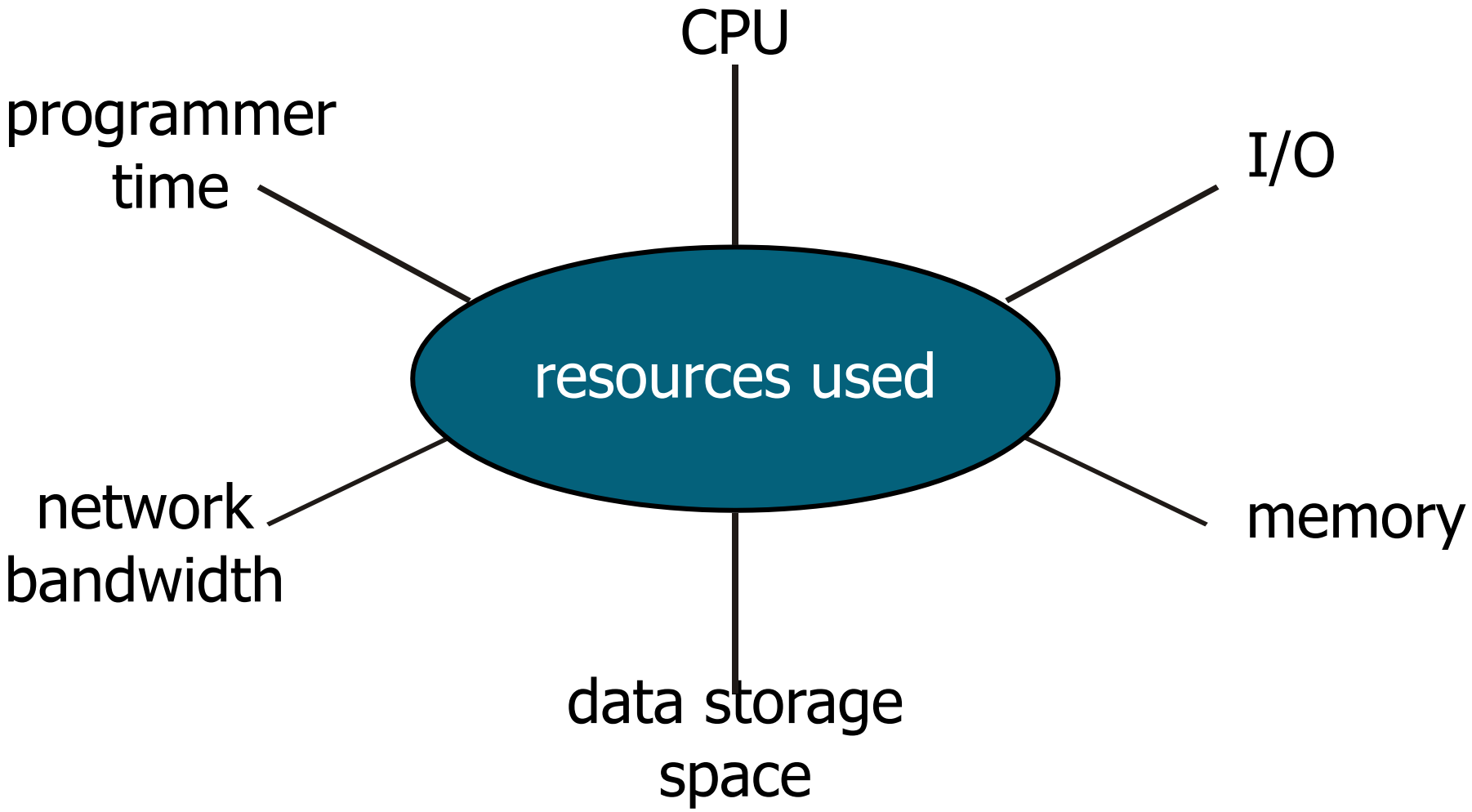
Running a SAS Program

- What resources are required to run a SAS program?
- The ***programmer*** must perform the following tasks:
  - determine program specifications
  - write the program
  - test the program
  - execute the program
  - maintain the program

Running a SAS Program

- The ***computer*** must perform the following actions:
  - load the required SAS software into memory
  - compile the program
  - read the data
  - execute the compiled program
  - store output data files
  - store output reports

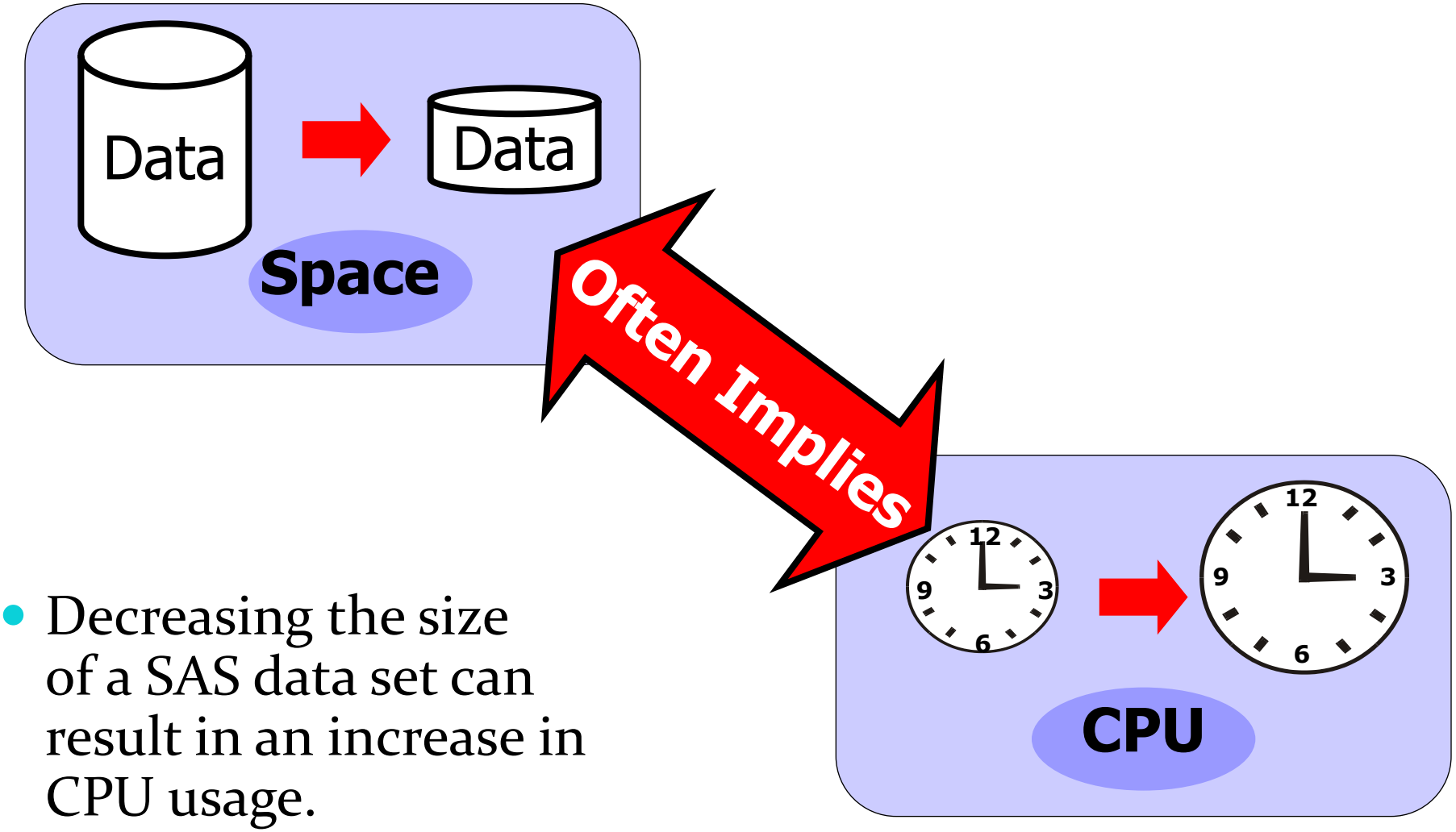
What Resources Are Used?



Understanding Efficiency Trade-offs

- When you decrease the use of one resource, the use of other resources might increase.
- Resource usage is dependent on your data. A specific technique might be more efficient with one data set and less efficient with another.

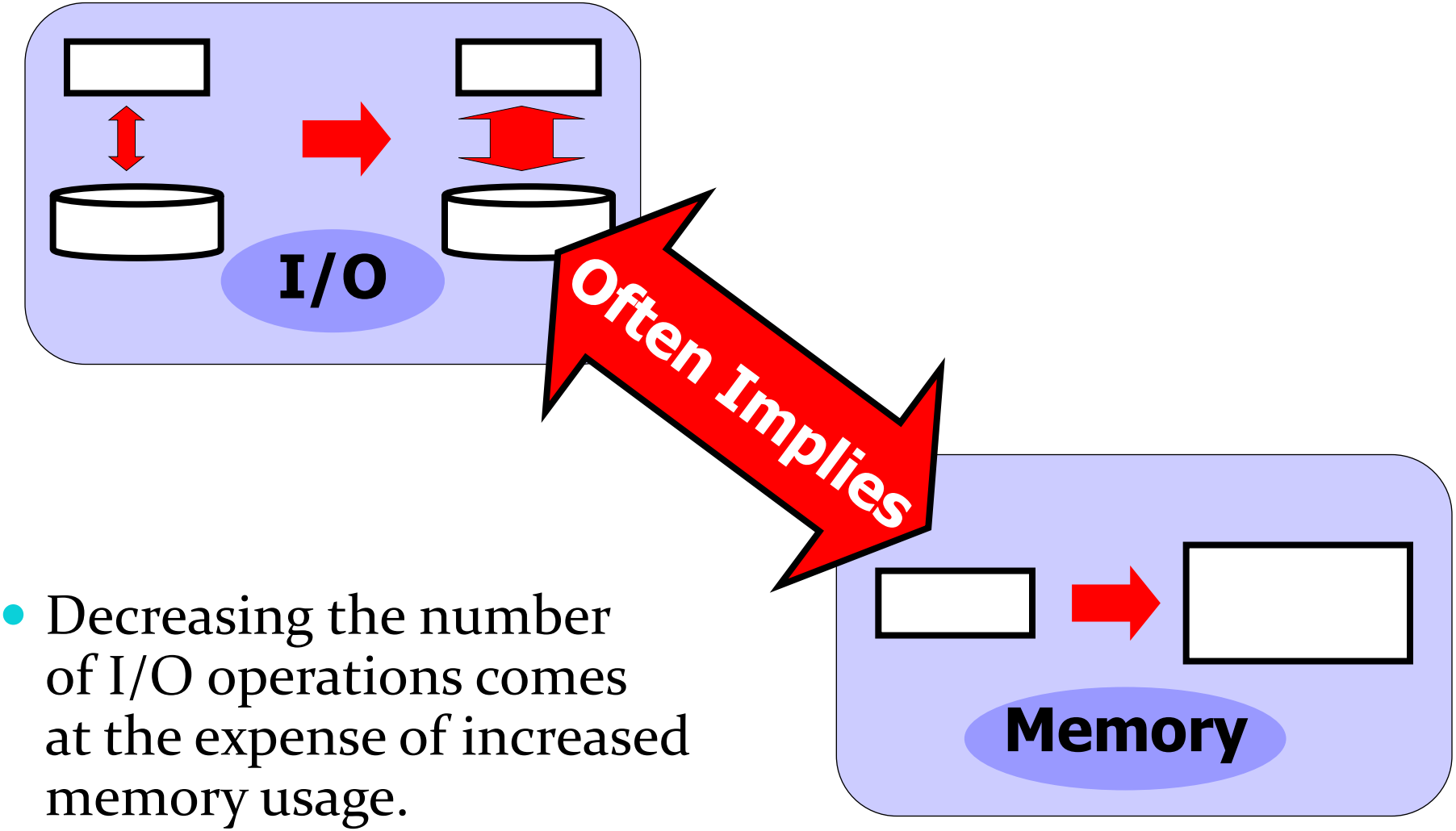
Understanding Efficiency Trade-offs



- Decreasing the size of a SAS data set can result in an increase in CPU usage.

...

Understanding Efficiency Trade-offs



- Decreasing the number of I/O operations comes at the expense of increased memory usage.

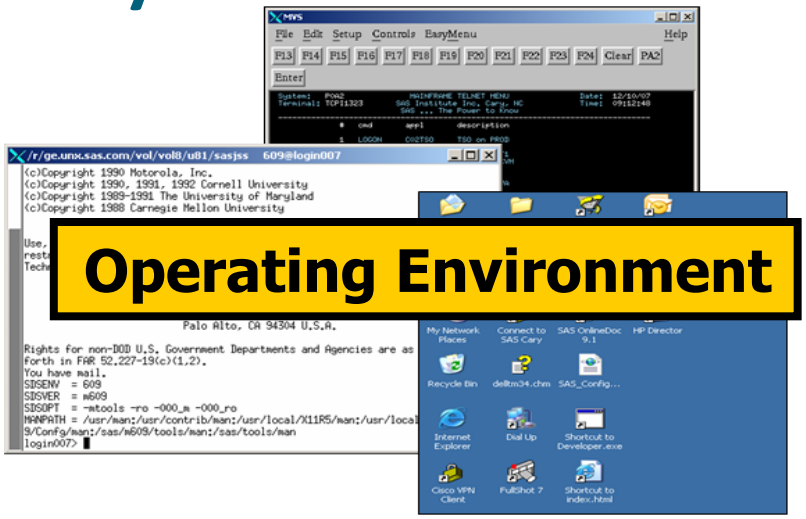
Deciding What Is Important for Efficiency

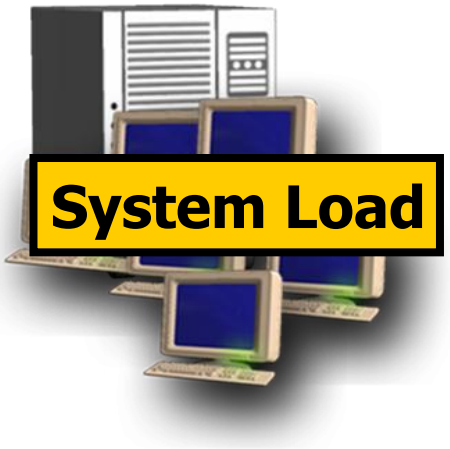
Your Programs

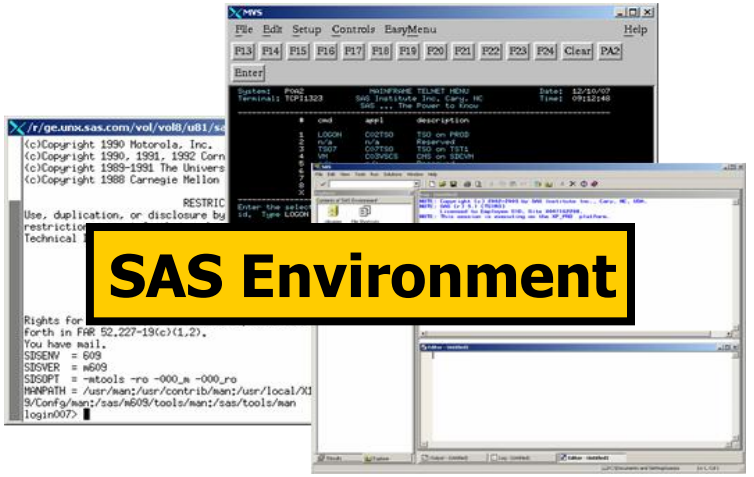
Your Site

Your Data

Understanding Efficiency at Your Site







Knowing How Your Program Will Be Used

- The importance of efficiency increases with the following:
  - the complexity of the program and/or the size of the files being processed
  - the number of times that the program will be executed

Knowing How Your Program Will Be Used

- What type(s) of data do you use?

a. SAS data sets
b. External files
c. Data from a relational database – for example, Oracle, Teradata, or SQL Server
d. Excel spreadsheets
e. OLAP cubes
f. Information maps
g. Other

Considering Trade-Offs

- In this class, many tasks are performed using one or more techniques.
- To decide which technique is most efficient for a given task, *benchmark*, or measure and compare, the resource usage of each technique.
- You should benchmark with the actual data to determine which technique is the most efficient.

The effectiveness of any efficiency technique depends greatly on the data with which you use the technique.

Running Benchmarks: Guidelines

- To benchmark your programming techniques, do the following:
  - Turn on the appropriate options to report resource usage.
  - Test each technique in a separate SAS session.
  - Test only one technique or change at a time, with as little additional code as possible.
  - Run your tests under the conditions that your final program will use (for example, batch execution, large data sets, and so on).

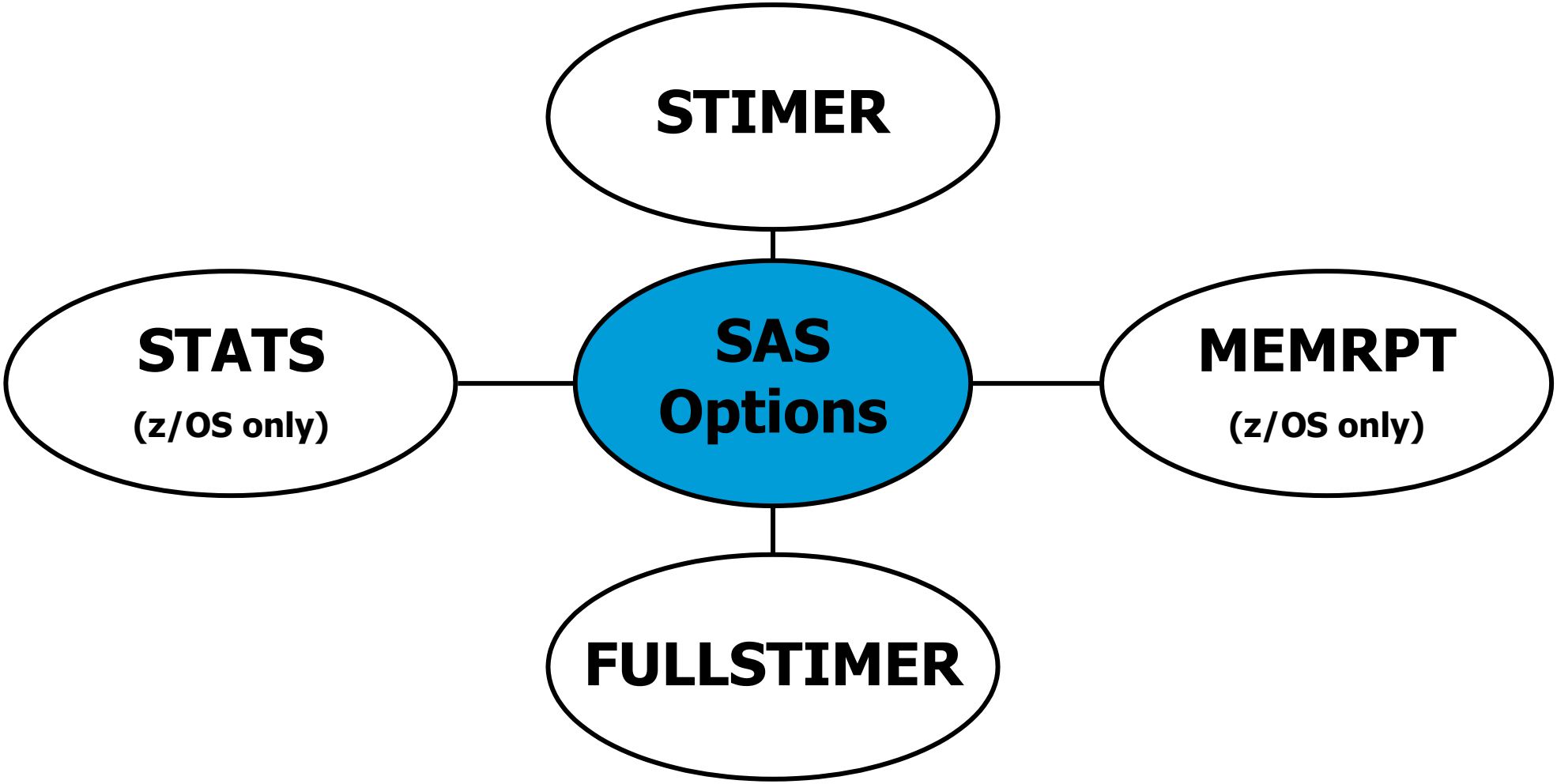
continued...

Running Benchmarks: Guidelines

- Run each program several times and base your conclusions on averages, not on a single execution. (This is more critical when you benchmark elapsed time.)
- Exclude outliers from the analysis because that data might lead you to tune your program to run less efficiently than it should.
- Turn off the options that report resource usage after testing is finished, because they consume resources.

In a multi-user environment, other computer activities might affect the running of your program.

Tracking Resource Usage



Tracking Resources with SAS Options

- **Windows, UNIX**

```
OPTIONS STIMER | NOSTIMER;
```

```
OPTIONS NOFULLSTIMER | FULLSTIMER;
```

- **z/OS**

```
STIMER | NOSTIMER
```

Invocation option only

```
OPTIONS STATS | NOSTATS;
```

```
OPTIONS MEMRPT | NOMEMRPT;
```

```
OPTIONS NOFULLSTIMER | FULLSTIMER;
```

Business Scenario

- You should benchmark to determine the most efficient technique for creating a new variable based on a condition.
- The following methods can be used:
  - IF-THEN with an assignment statement
  - IF-THEN/ELSE with an assignment statement
  - SELECT/WHEN with an assignment statement

Poll

Quiz

1.07 Quiz

1. Open and submit **p301a01a**.
   Record the user CPU: _____
   Exit SAS.
2. Start SAS.
   Open and submit **p301a01b**.
   Record the user CPU: _____
   Exit SAS.
3. Start SAS.
   Open and submit **p301a01c.**
   Record the user CPU: _____
4. Which technique is most efficient?

Sample Windows Log

```
5     options fullstimer;
6     data _null_;
7        length var $ 30;
8        retain var2-var50 0 var51-var100 'ABC';
9        do x=1 to 100000000;
10          var1=10000000*ranuni(x);
11          if var1>1000000 then var='Greater than 1,000,000';
12          if 500000<=var1<=1000000 then var='Between 500,000 and 1,000,000';
13          if 100000<=var1<500000 then var='Between 100,000 and 500,000';
14          if 10000<=var1<100000 then var='Between 10,000 and 100,000';
15          if 1000<=var1<10000 then var='Between 1,000 and 10,000';
16          if var1<1000 then var='Less than 1,000';
17       end;
18    run;

NOTE: DATA statement used (Total process time):
      real time           1.26 seconds
      user cpu time       0.98 seconds
      system cpu time     0.04 seconds
      Memory                           278k
      OS Memory                        4976k
      Timestamp            6/29/2010  12:39:21 PM
```

p301a01a

Sample UNIX Log

```
1    options fullstimer;
2    data _null_;
3       length var $30;
4       retain var2-var50 0 var51-var100 'ABC';
5       do x=1 to 10000000;
6          var1=10000000*ranuni(x);
7          if var1>10000000 then var='Greater than 1,000,000';
8          if 500000<=var1<=1000000 then var='Between 500,000 and 1,000,000';
9          if 100000<=var1<500000 then var='Between 100,000 and 500,000';
10         if 10000<=var1<100000 then var='Between 10,000 and 100,000';
11         if 1000<=var1<10000 then var='Between 1,000 and 10,000';
12         if var1<1000 then var='Less than 1,000';
13      end;
14   run;

NOTE: DATA statement used (Total process time):
      real time           6.62 seconds
      user cpu time       5.14 seconds
      system cpu time     0.01 seconds
      Memory                              526k
      OS Memory                           5680k
      Timestamp            6/29/2010  11:55:32 AM
      Page Faults                         82
      Page Reclaims                       0
      Page Swaps                          0
      Voluntary Context Switches          91
      Involuntary Context Switches        48
      Block Input Operations              91
      Block Output Operations             0
```

SAS DATA Step Processing

Objectives

- List the attributes of a data set page and define how it relates to the structure of SAS data sets.
- Describe how SAS reads and writes data.

SAS Data Set Pages

- A SAS *data set page* has the following attributes:
  - It is the unit of data transfer between the operating system buffers and SAS buffers in memory.
  - It includes the number of bytes used by the descriptor portion, the data values, and any operating system overhead.
  - It is fixed in size when the data set is created, either to a default value or to a value specified by the programmer.

Using PROC CONTENTS to Report Page Size

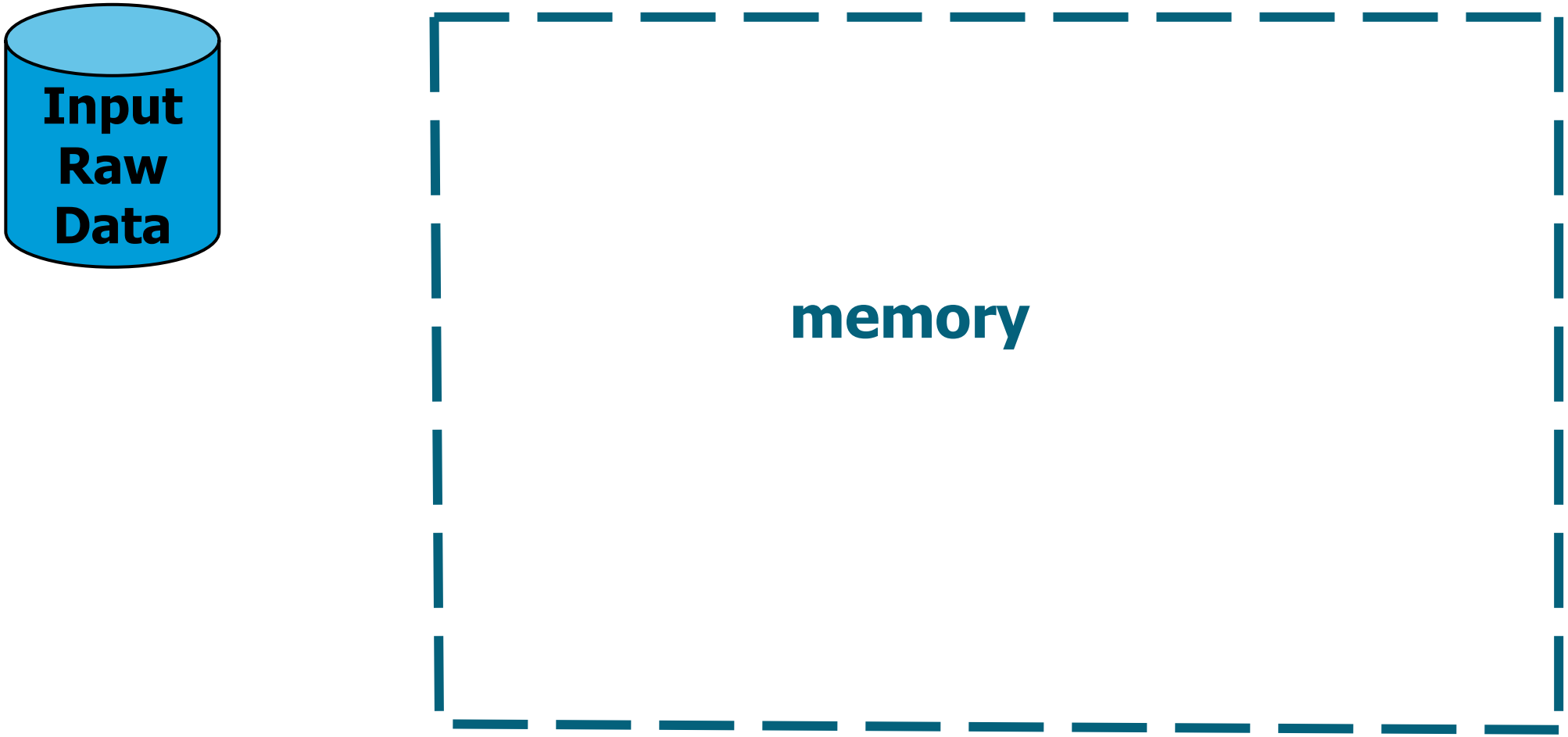
```
proc contents data=orion.sales_history;
run;
```

- Partial PROC CONTENTS Output

16,384*18= 294,912 bytes

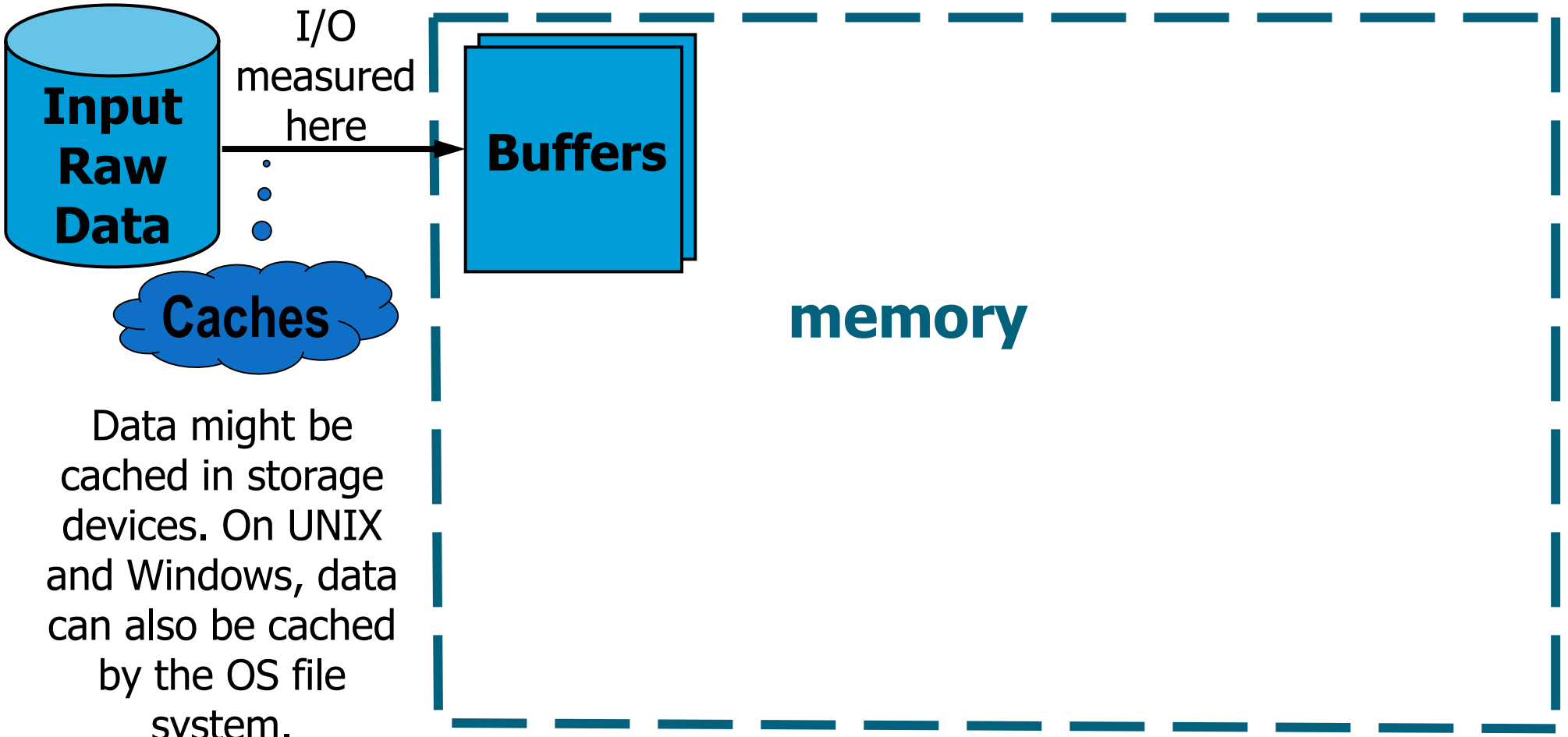
```
 Engine/Host Dependent Information

Data Set Page Size          16384
Number of Data Set Pages    18
First Data Page             1
Max Obs per Page            92
Obs in First Data Page      72
Number of Data Set Repairs  0
File Name                   S:\workshop\sales_history.sas7bdat
Release Created             9.0201M0
Host Created                XP_PRO
```

Reading External Files



...

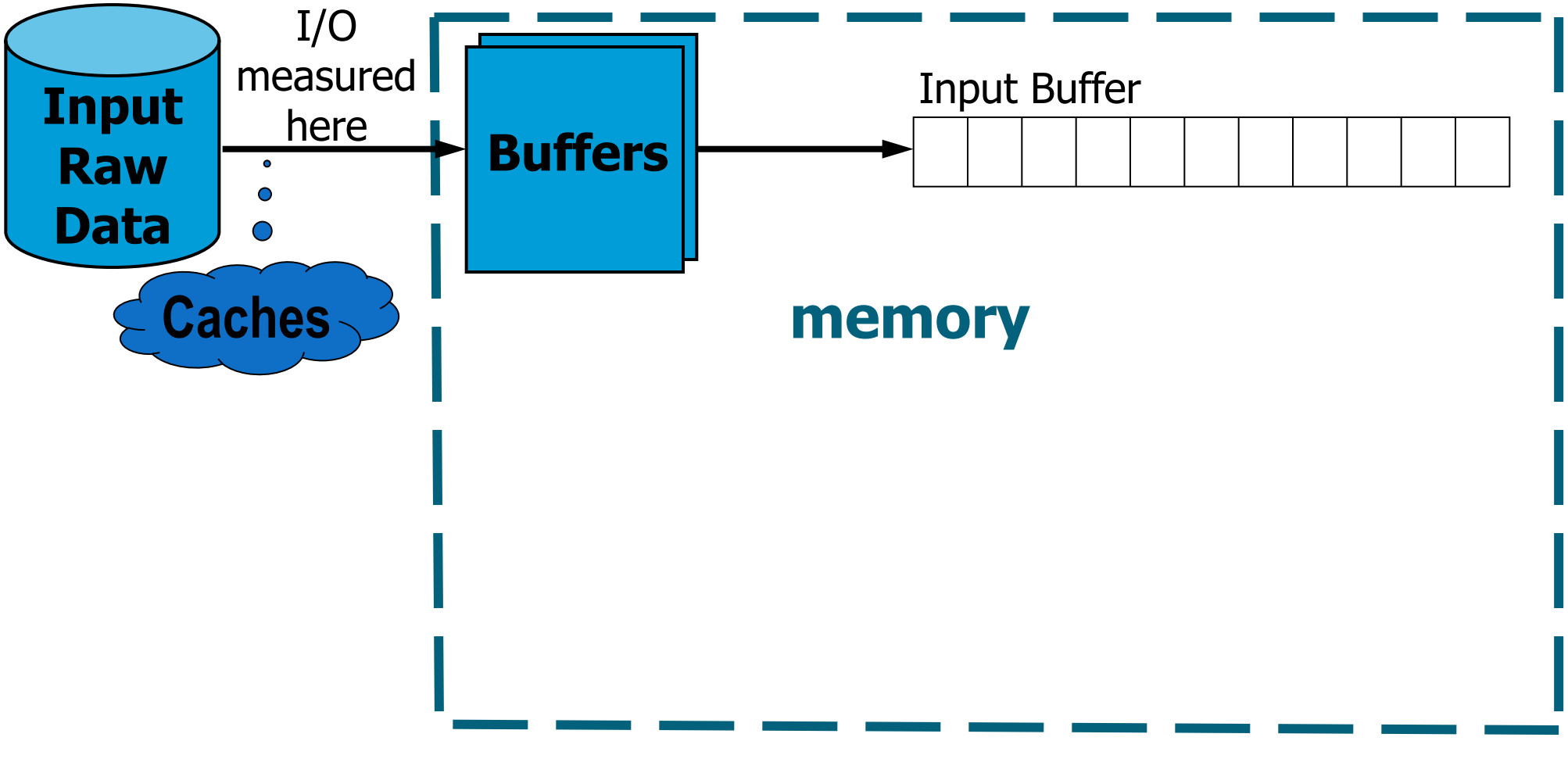
Reading External Files



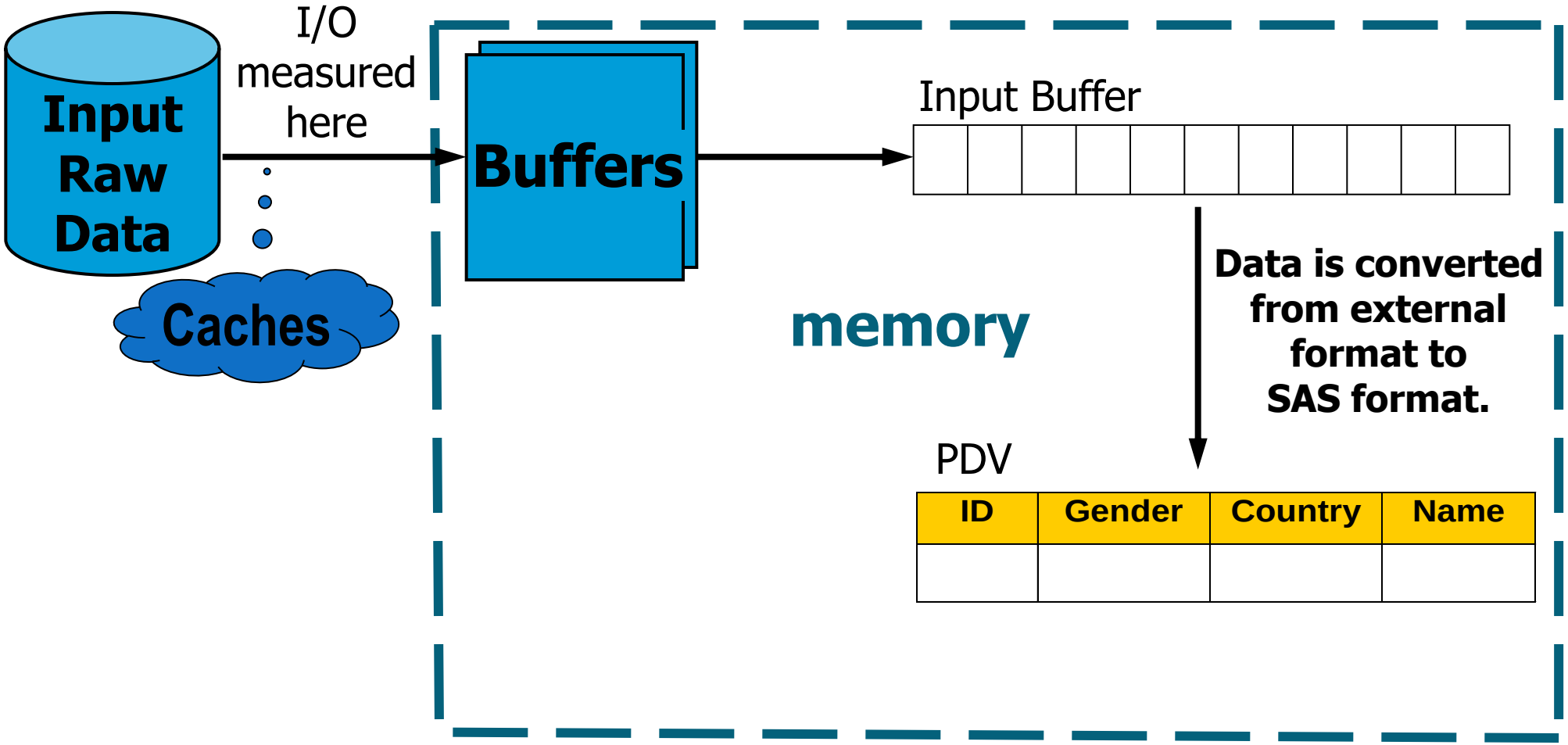
Data might be cached in storage devices. On UNIX and Windows, data can also be cached by the OS file system.

...

Reading External Files

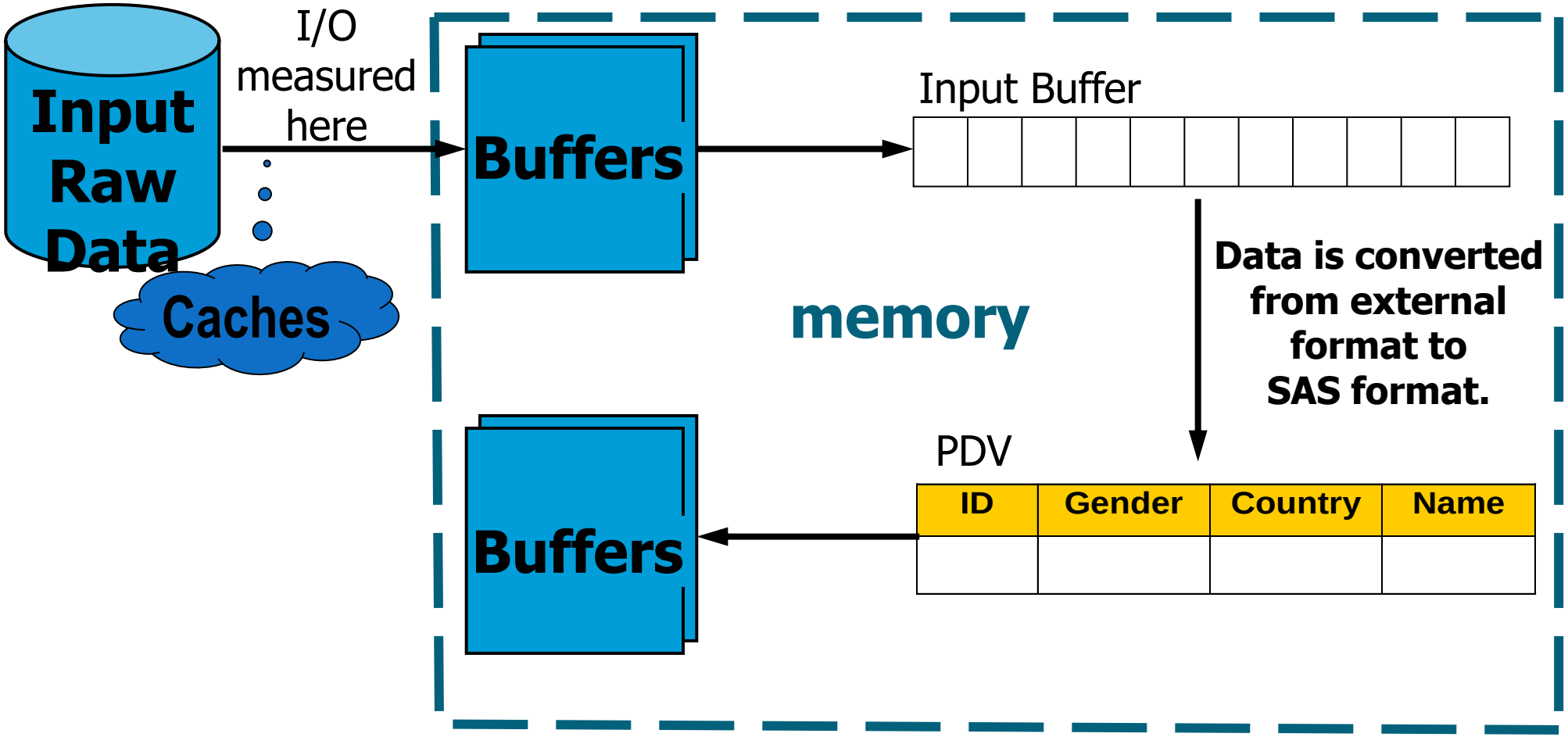


Reading External Files



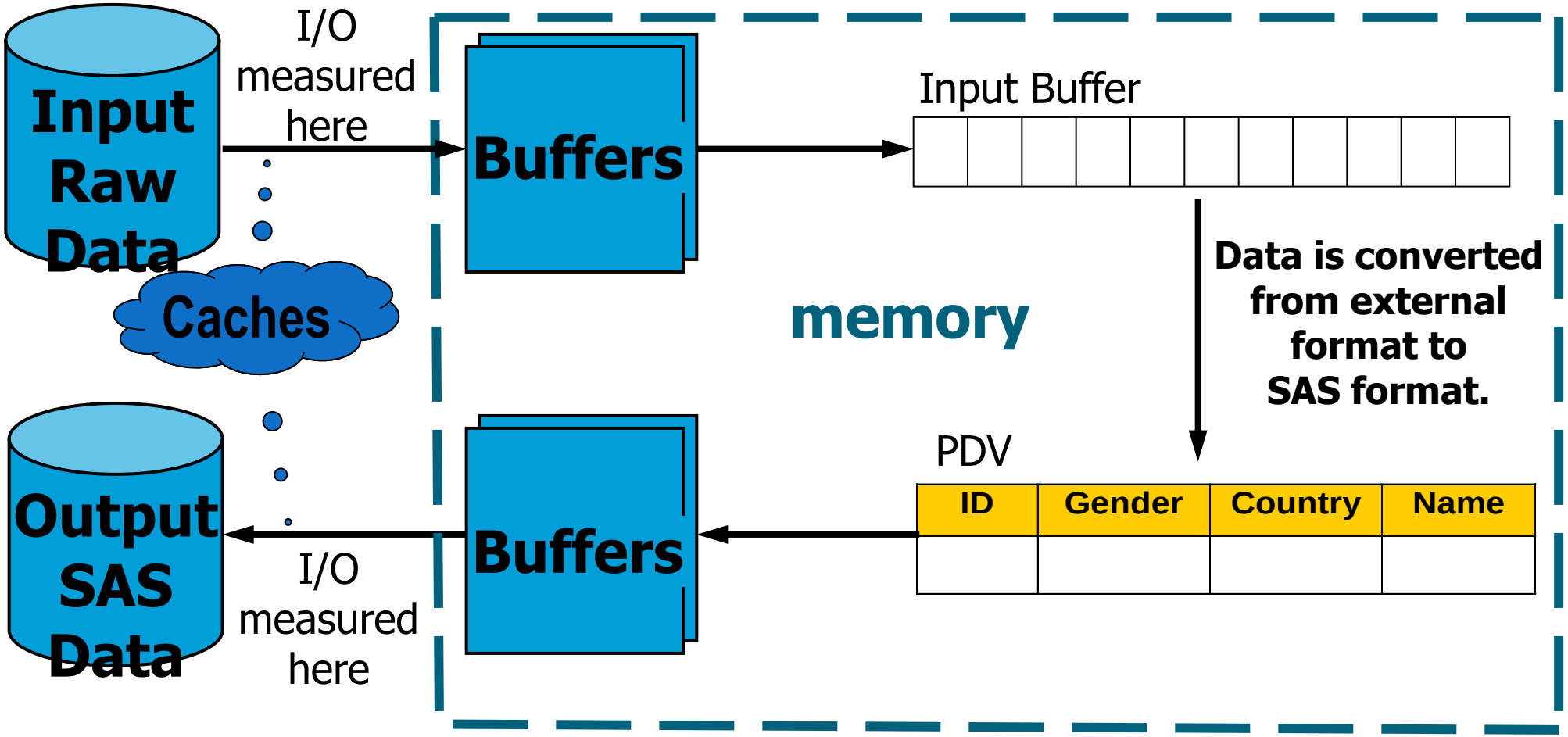
...

Reading External Files

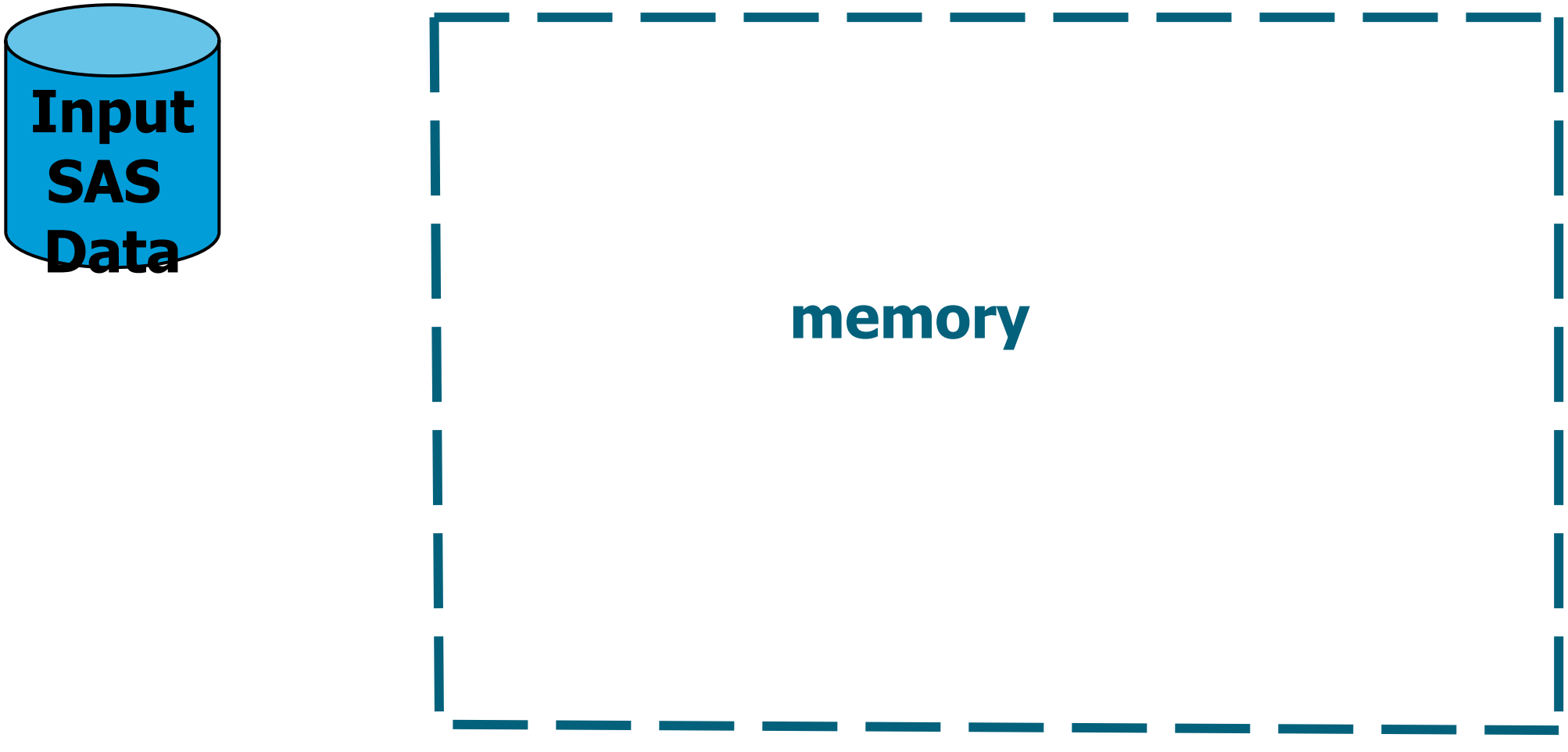


...

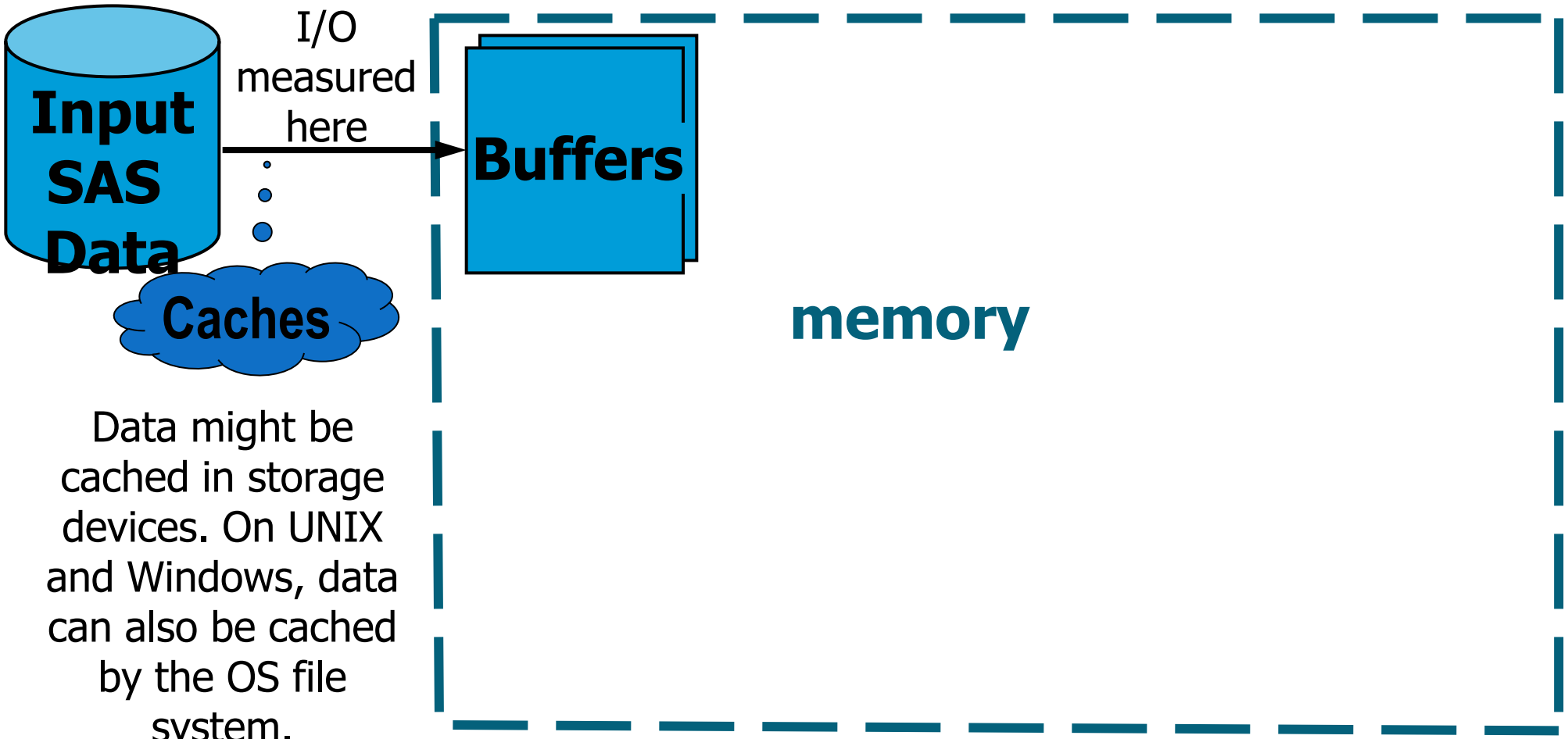
Reading External Files



Reading a SAS Data Set with a SET Statement



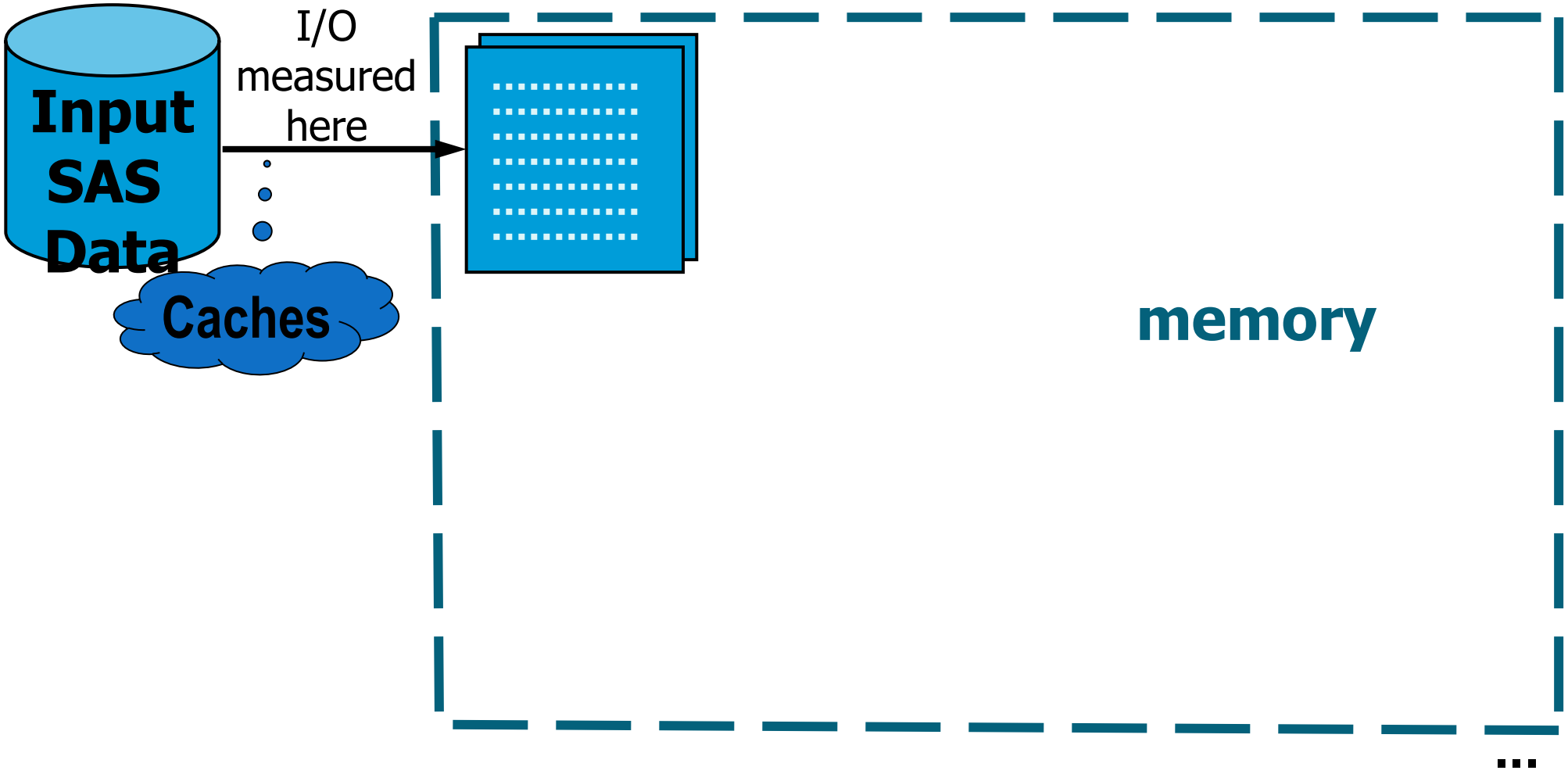
Reading a SAS Data Set with a SET Statement



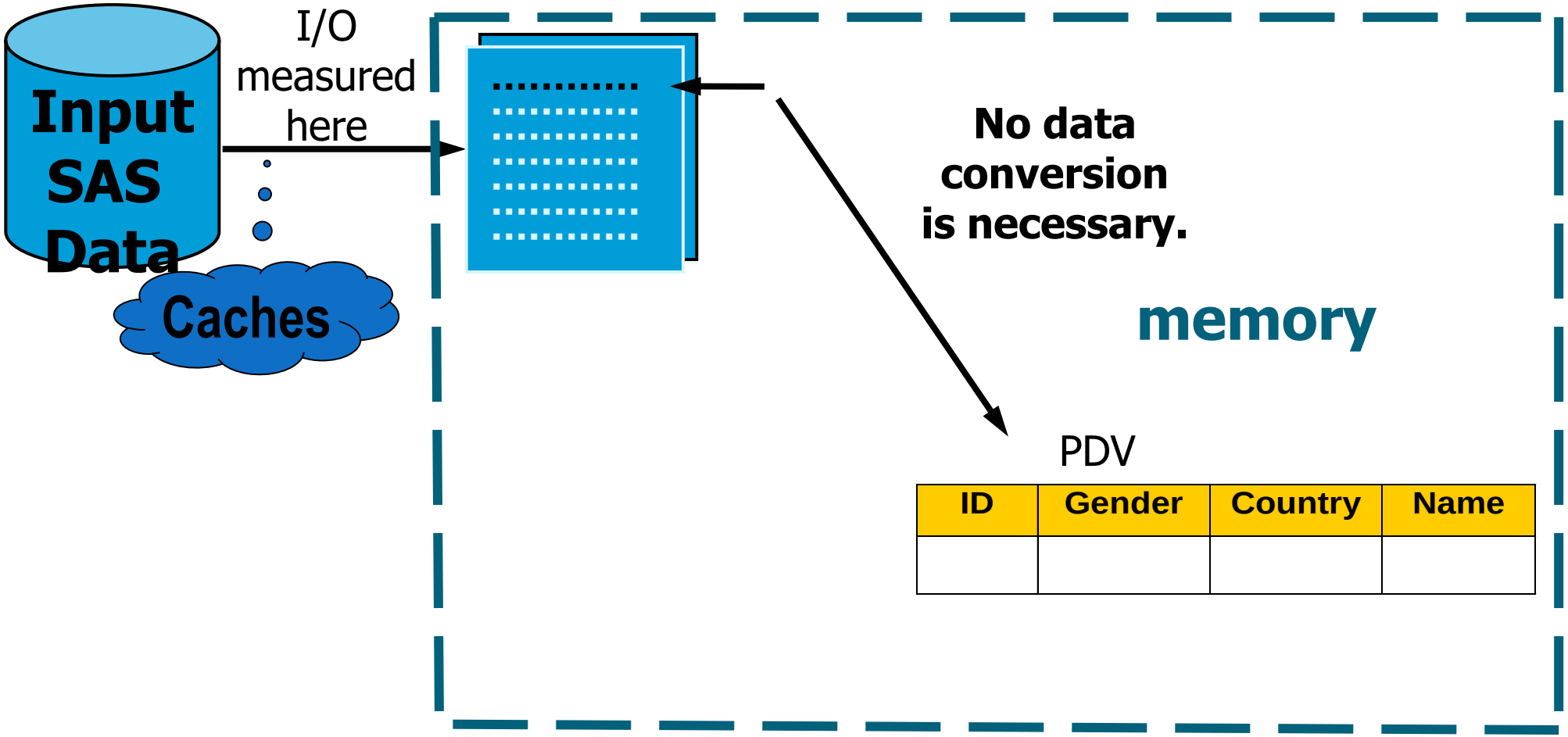
Data might be cached in storage devices. On UNIX and Windows, data can also be cached by the OS file system.

...

Reading a SAS Data Set with a SET Statement

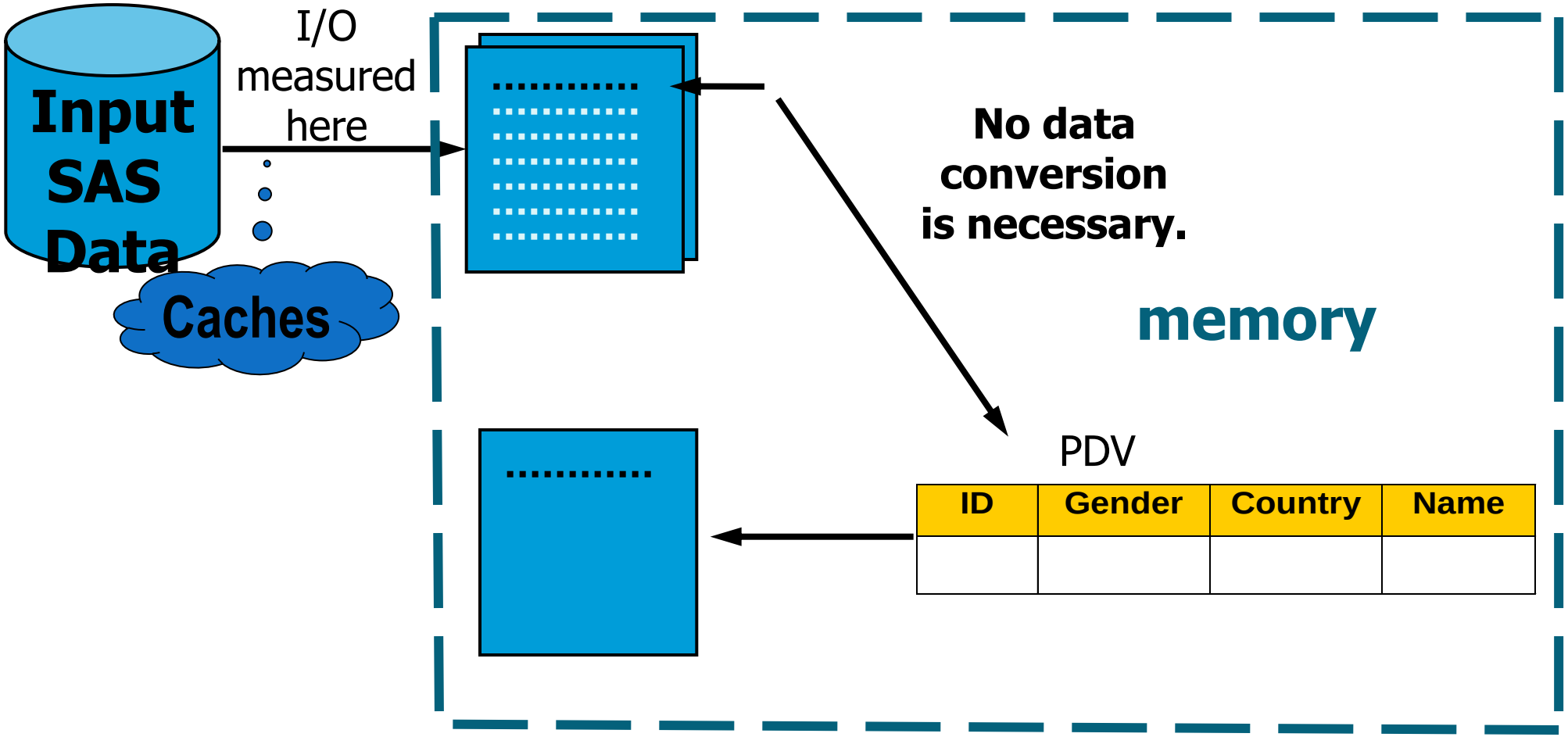


Reading a SAS Data Set with a SET Statement



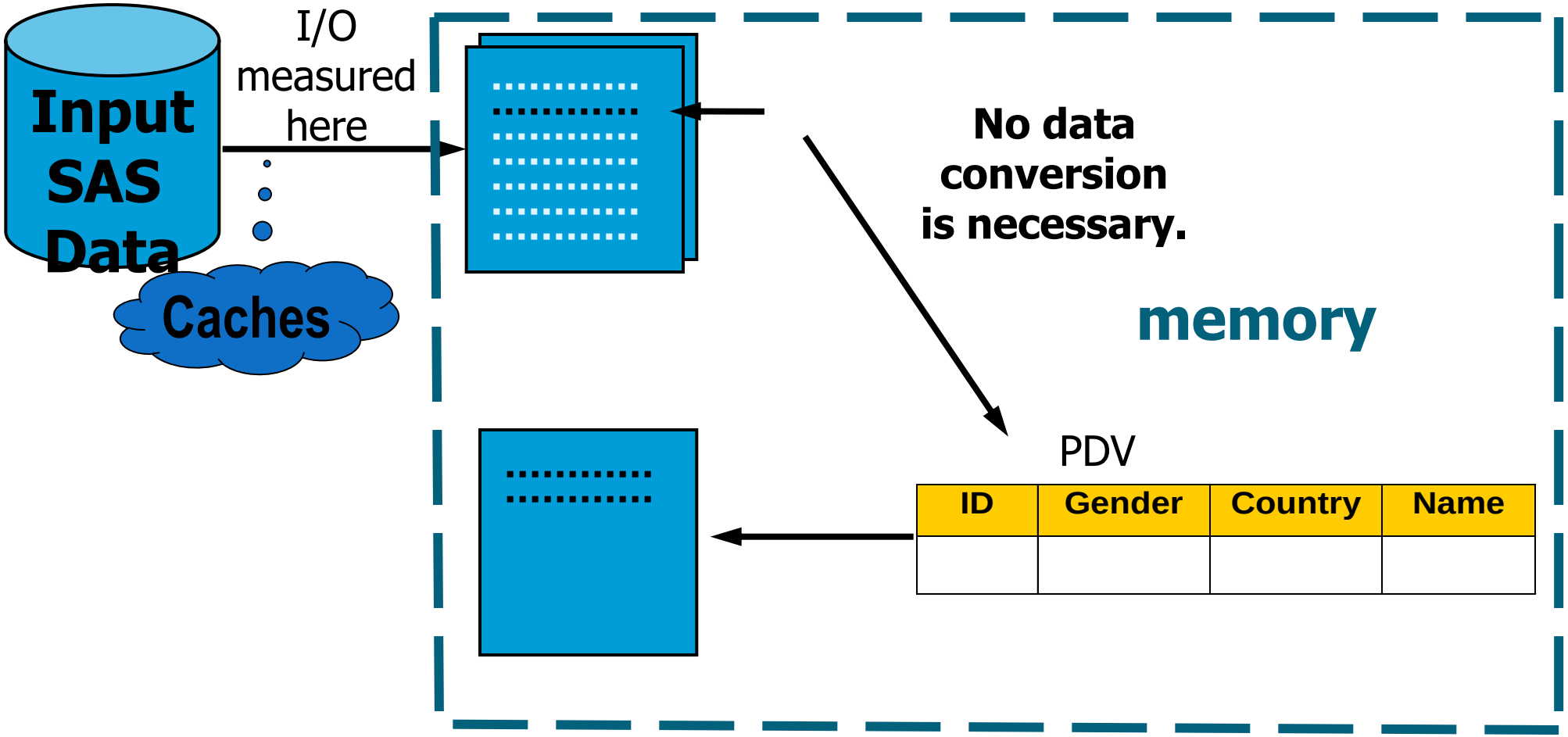
...

Reading a SAS Data Set with a SET Statement



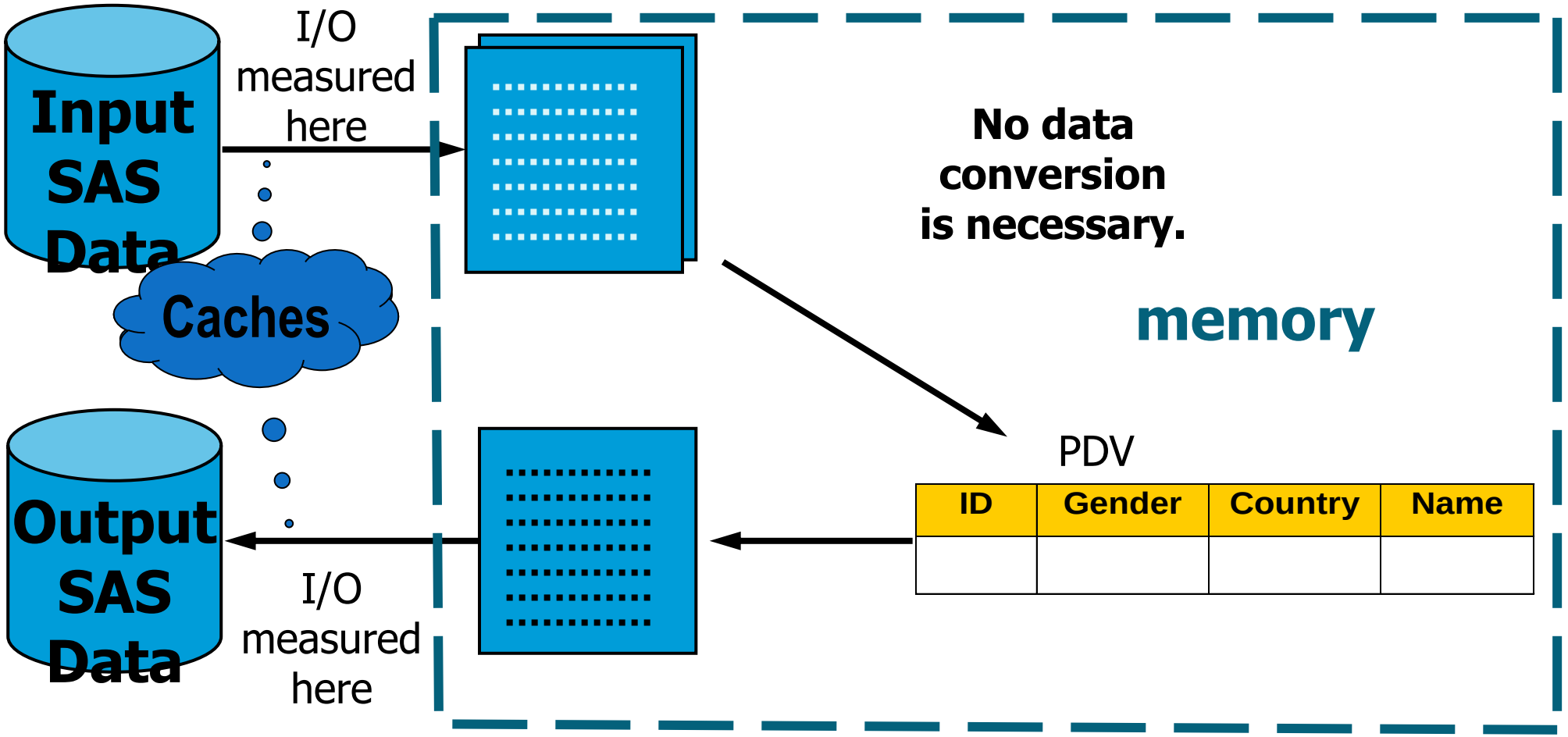
...

Reading a SAS Data Set with a SET Statement



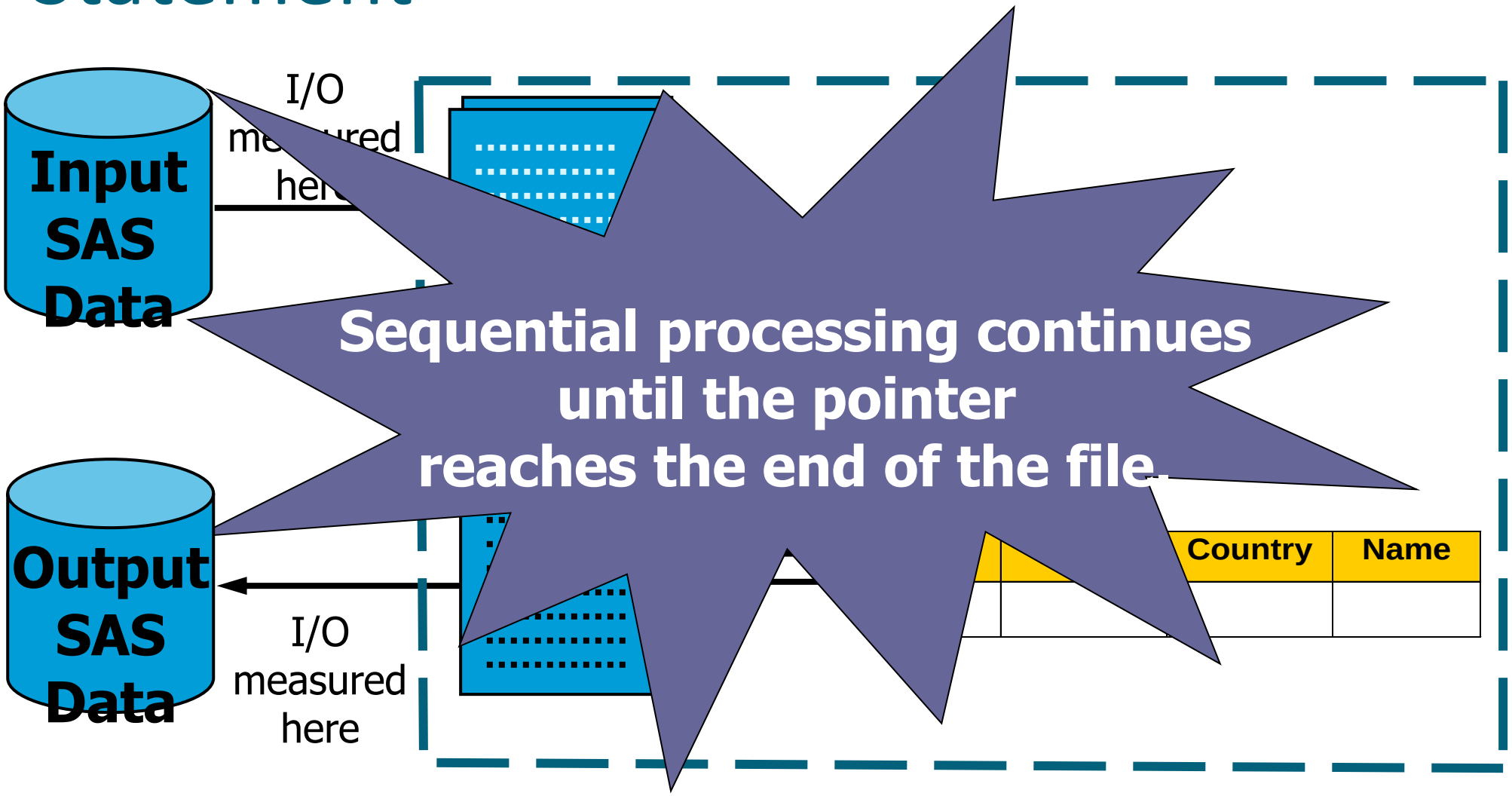
...

Reading a SAS Data Set with a SET Statement



...

Reading a SAS Data Set with a SET Statement



Chapter 2: Controlling I/O Processing and Memory

2.1: Controlling I/O

2.2: Controlling Data Set Size

2.3: Compressing SAS Data Sets

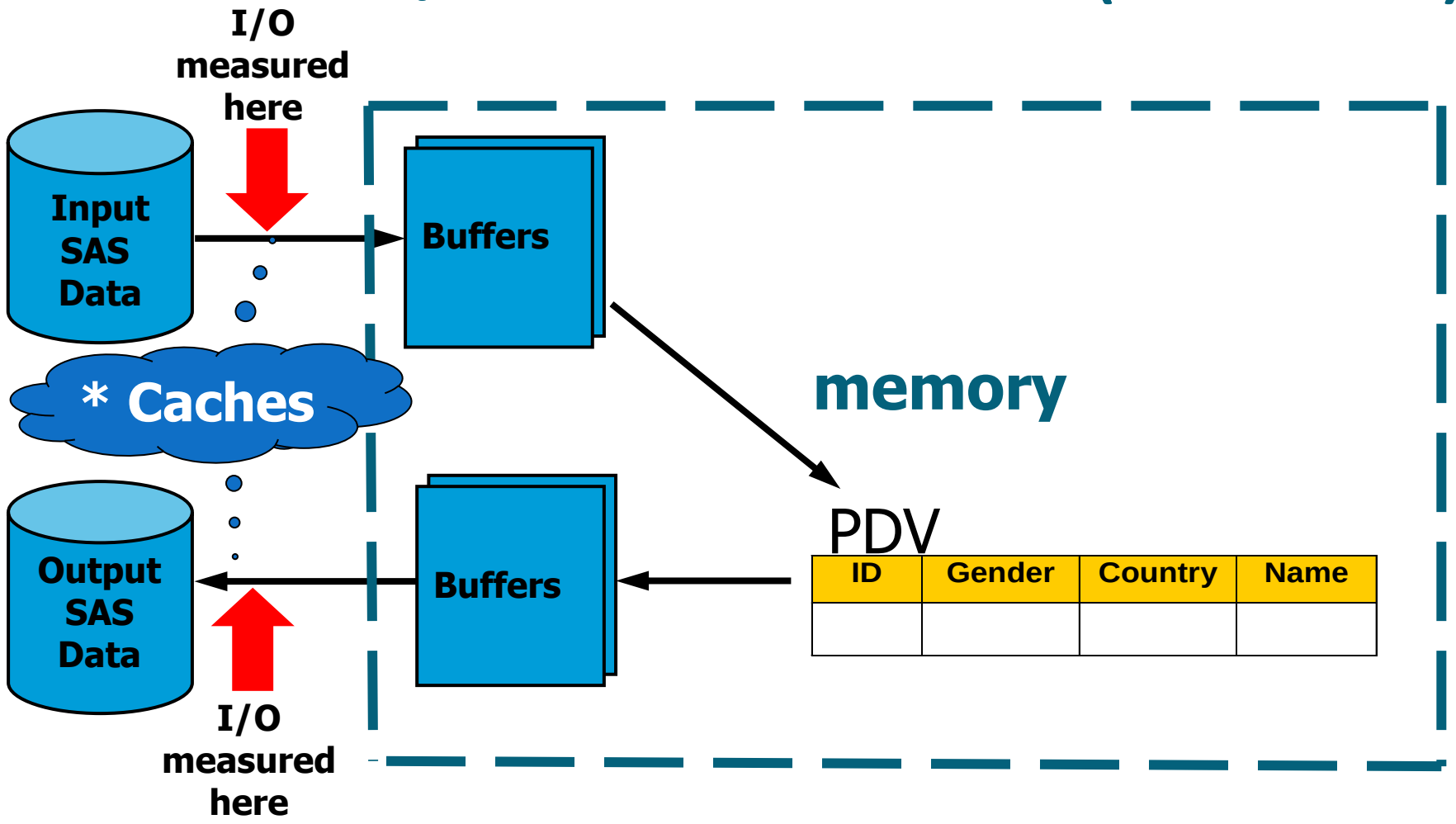
2.4: Controlling Memory (Self-Study)

I/O (Review)

- SAS programs typically perform the following tasks:
  - reading data sets sequentially
  - performing analysis and data manipulation
  - writing data sets sequentially or writing reports

I/O is one of the most important factors for optimizing performance.

Where Is I/O Measured? (Review)



* Windows and UNIX Only

Using the Operating Environment Cache in Windows and UNIX

- Windows and UNIX use a file-caching mechanism in the background.
  - A file *cache* is an area in memory that holds recently accessed data.
  - By default, SAS reads and writes data through the operating environment file cache, not by direct I/O.
  - File caching is beneficial if the same data is used more than once.
  - File caching adds overhead to sequential I/O.

Techniques for Reducing I/O Operations

1. Minimize the number of variables and observations.
2. Reduce the number of times that the data is processed.
3. Create a SAS data file when you process the same raw data file repeatedly.
4. Use the SASFILE statement to process a small SAS data set repeatedly.
5. Minimize the size of the SAS data set.
6. Use appropriate BUFSIZE= and/or BUFNO= options for random or sequential access.

continued...

Techniques for Reducing I/O Operations

7. Bypass system file caching in Windows and UNIX.
8. Create views in programs that require intermediate temporary SAS data files.
9. Create indexes on variables used for WHERE processing.

Technique 1: Process Only the Necessary Variables and Observations

Simple techniques can conserve I/O. The amount of I/O saved depends on the size of the subset being processed.

- Reduce the number of variables.
  - DROP or KEEP statements
  - DROP= or KEEP= data set options
- Reduce the number of observations.
  - WHERE statement
  - WHERE= data set option
  - OBS= and FIRSTOBS= data set options

Subsetting Data

Program 1: Subsetting in the Procedure

One way to create a subset is to use the WHERE statement in a procedure.

```
data bonus;
   set orion.staff;
   YrEndBonus=Salary*0.05;
run;
proc means data=bonus mean sum;
   where Job_Title contains 'Manager';
   class Manager_ID;
   var YrEndBonus;
run;
```

- The data set **bonus** contains 11 variables and 424 observations.

p302d01

Subsetting Data

- **Program 2:** Subsetting in the DATA Step
- Because the DATA step is required to create the variable **YrEndBonus**, it is more efficient to subset in the DATA step.

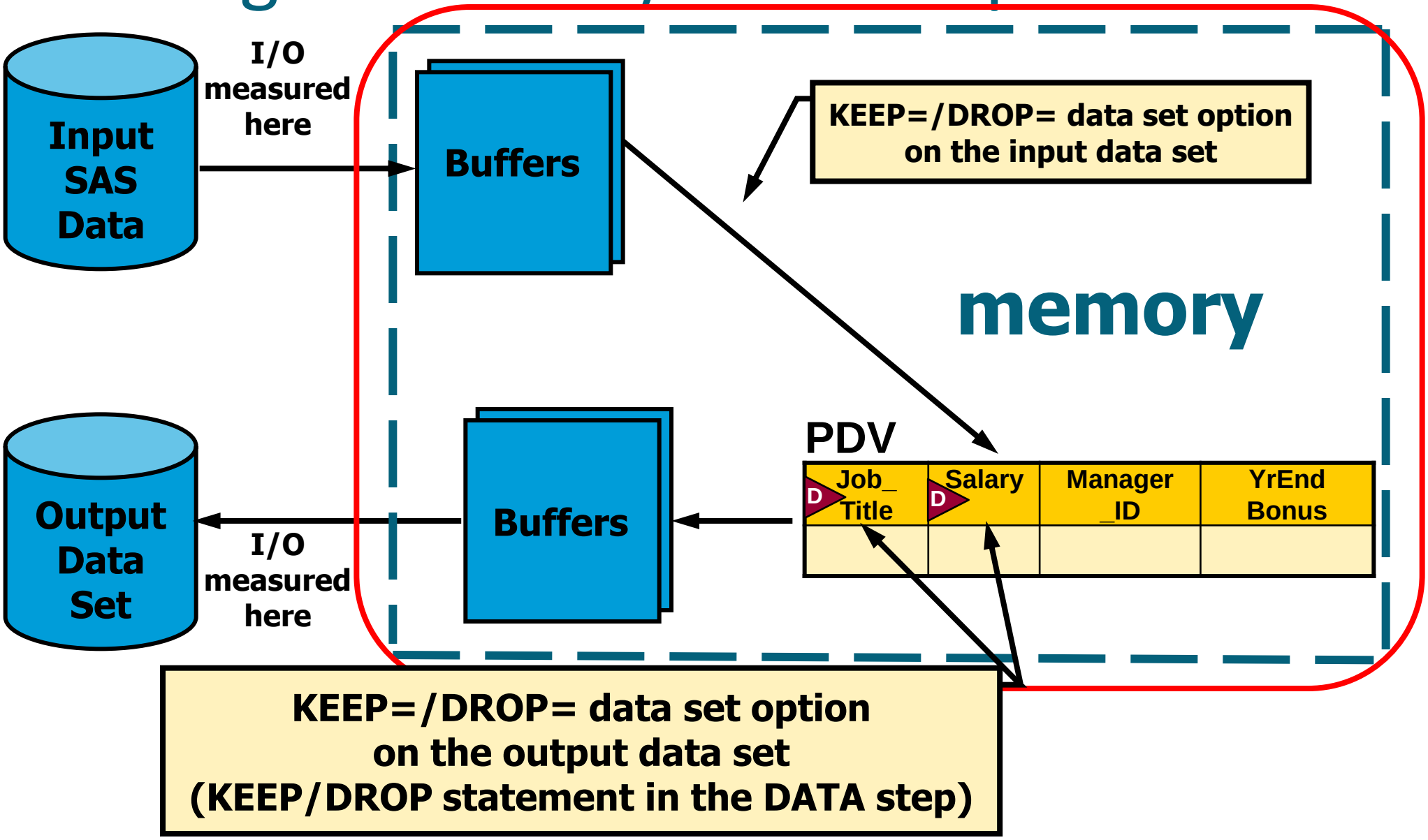
```
data bonus(keep=Manager_ID YrEndBonus);
   set orion.staff(keep=Job_Title Salary Manager_ID);
   where Job_Title contains 'Manager';
   YrEndBonus=Salary*0.05;
run;
proc means data=bonus mean sum;
   class Manager_ID;
   var YrEndBonus;
run;
```

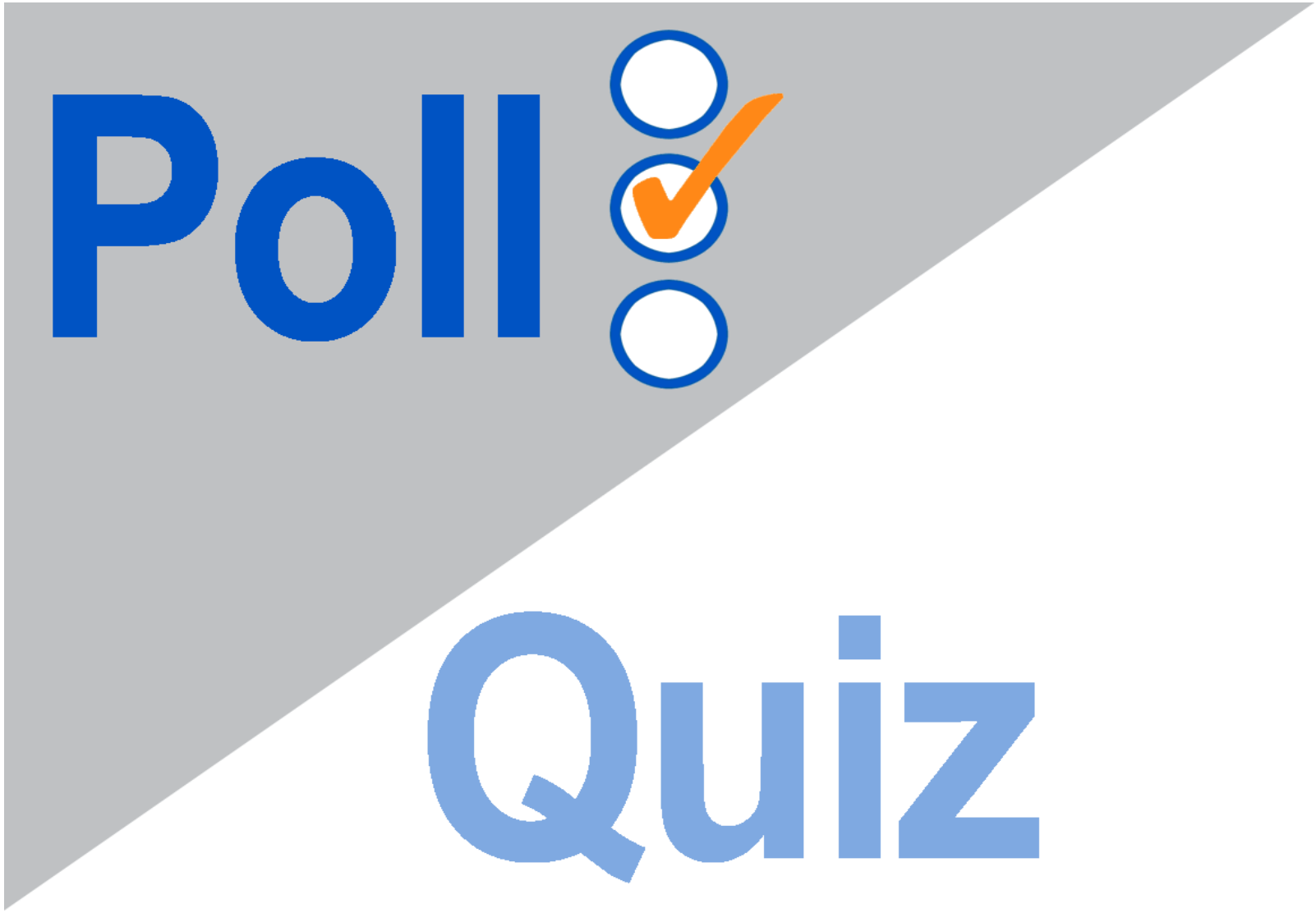
I/O savings result from reducing the number of variables and observations in the input and output data sets.

- ✎ The data set **bonus** contains two variables and 41 observations.

p302d01

Using the KEEP=/DROP= Options




Poll
Quiz

Setup for the Poll

- p302d01 Program 1

```
data bonus;
   set orion.staff;
   YrEndBonus=Salary*0.05;
run;
proc means data=bonus mean sum;
   where Job_Title contains 'Manager';
   class Manager_ID;
   var YrEndBonus;
run;
```

I/O savings results from reducing the number of variables and observations in the input and output data sets.

p302d01 Program 2

```
data bonus(keep=Manager_ID YrEndBonus);
  set orion.staff(keep=Job_Title Salary Manager_ID);
      where Job_Title contains 'Manager';
          YrEndBonus=Salary*0.05;
                run;
   proc means data=bonus mean sum;
          class Manager_ID;
           var YrEndBonus;
                run;
```

2.01 Multiple Choice Poll

- In addition to the I/O decrease when the DATA step creates **bonus**, where does Program 2 have additional decrease of I/O?

  a. Fewer variables are read into the program data vector from **orion.staff** in Program 2 because of the KEEP= data set option.

  b. The PROC MEANS in Program 2 loads a smaller version of **bonus**.

  c. There is no additional decrease in I/O; all of the decrease in I/O occurs when the data set **bonus** is created by the DATA step.

2.01 Multiple Choice Poll – Correct Answer

- In addition to the I/O decrease when the DATA step creates **bonus**, where does Program 2 have additional decrease of I/O?

  a. Fewer variables are read into the program data vector from **orion.staff** in Program 2 because of the KEEP= data set option.

  b. The PROC MEANS in Program 2 loads a smaller version of **bonus**.

  c. There is no additional decrease in I/O; all of the decrease in I/O occurs when the data set **bonus** is created by the DATA step.

Technique 2: Reducing the Number of Times that Data Is Processed

- The following techniques reduce the number of times that data is processed:
  - Subset data within a procedure step if possible.
  - Create SAS data files. SAS can process SAS data files more efficiently than raw data files.
  - Use engines efficiently.
  - Use indexes.
  - Access data through SAS views.

Subsetting Data within a Procedure Step

- **Program 1:** Subset in the DATA Step
- You can subset in the DATA step first, and then execute the

```
data big_salaries;
   set orion.staff;
   where Salary > 50000;
run;

proc means data=big_salaries mean sum;
   class Manager_ID;
   var Salary;
run;
```


p302d02

Subsetting Data within a Procedure Step

- **Program 2:** Subset in the Procedure
- A better way is to execute only one step and subset in that step.

```
proc means data=orion.staff mean sum;
    class Manager_ID;
    var Salary;
    where Salary > 50000;
run;
```

I/O savings results from avoiding an extra step for subsetting.

Technique 3: Creating a SAS Data File

Program 1: Write two programs. Each program should create a temporary SAS data set to read in the same raw data file and create a report.

```
data prices;
   infile 'prices.dat' dlm='*';
   input Product_ID : 12. Start_Date : date9. End_Date : date9.
         Unit_Cost_Price:dollar7.2 Unit_Sales_Price:dollar7.2;
run;
proc print data=prices(obs=5);
  title1 "Prices Data Set";
run;
```

```
data prices;
   infile 'prices.dat' dlm='*';
   input Product_ID : 12. Start_Date : date9. End_Date : date9.
         Unit_Cost_Price:dollar7.2 Unit_Sales_Price:dollar7.2;
run;
proc means data=prices(keep=Unit_Cost_Price Unit_Sales_Price);
  var Unit_Cost_Price Unit_Sales_Price;
run;
```


p302d03

Creating a SAS Data File

Program 2: Create a permanent SAS data set by reading in the raw data file one time. Write two programs using the SAS data set to create a report.

```
data orion.prices;
   infile 'prices.dat' dlm='*';
   input Product_ID : 12. Start_Date : date9.
         End_Date : date9.
         Unit_Cost_Price : dollar7.2
         Unit_Sales_Price : dollar7.2;
run;
proc print data=orion.prices(obs=5);
  title1 "Prices Data Set";
run;
```

I/O savings from reading the raw data once

```
proc means data=orion.prices
      (keep=Unit_Cost_Price Unit_Sales_Price);
  var Unit_Cost_Price Unit_Sales_Price;
run;
```

Technique 4: SASFILE Global Statement

If your program uses the same data set multiple times, the SASFILE statement can reduce I/O resources.

- The SASFILE statement loads the SAS data set into memory in its entirety, instead of a few pages at a time.
- After it is loaded, the data set is held in memory for subsequent DATA and PROC step processing.
- A second SASFILE statement closes the file and frees the SAS buffers.
- It is useful for data sets that fit entirely into memory.
- The SASFILE statement should not be used if the data set is larger than available physical memory.

The reduction in I/O resources comes at the cost of increased memory usage.

Business Scenario

Create reports using the PRINT, TABULATE, MEANS, and FREQUENCY procedures against a single SAS data set.

```
sasfile orion.customer_dim load;

proc freq data=orion.customer_dim;
   tables Customer_Country Customer_Ty
run;
proc print data=orion.customer_dim no
   where Customer_Type='Orion Club Go
   var Customer_ID Customer_Name Custo
run;
proc means data=orion.customer_dim me
   var Customer_Age;
   class Customer_Group;
run;
proc tabulate data=orion.customer_dim
   class Customer_Age_Group Customer_
   table Customer_Type All=Total,
         Customer_Age_Group*n=' ' All=Total*n=' '/rts=45;
run;

sasfile orion.customer_dim close;
```

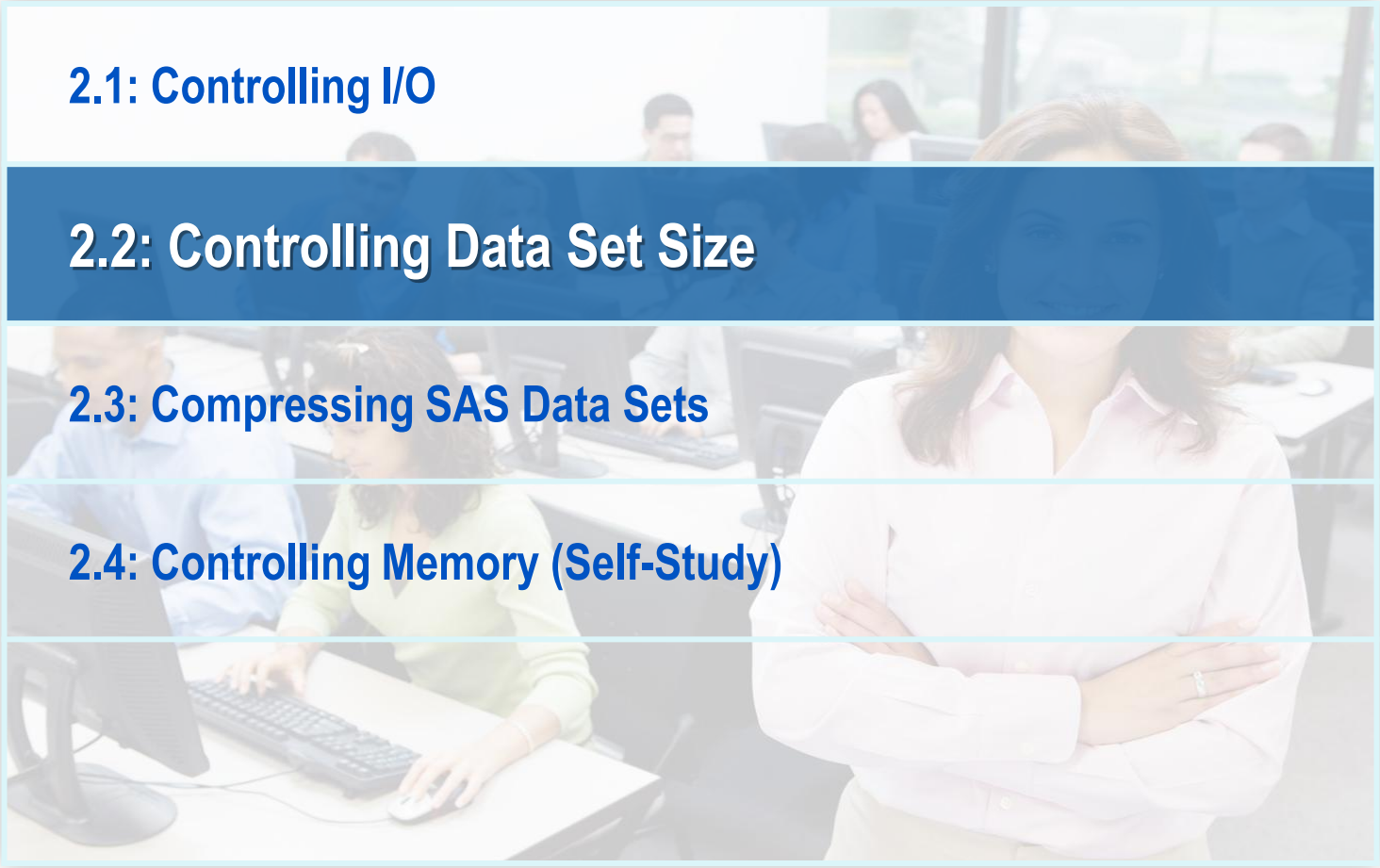
The **orion.customer_dim** data set is read into memory only once instead of four times. This results in one-fourth as many I/O operations, which can also reduce elapsed time. However, it comes at the expense of increased memory usage.

SASFILE Global Statement

- General form of the SASFILE statement:

```
SASFILE <libref.>member-name
        <(password-data-set-option(s))>
        OPEN | LOAD | CLOSE;
```

  - When the SASFILE statement executes, SAS allocates the number of buffers based on the number of pages in the SAS data set and index file.
  - If the file in memory increases in size during processing by editing or appending data, the number of buffers also increases.



2.1: Controlling I/O

2.2: Controlling Data Set Size

2.3: Compressing SAS Data Sets

2.4: Controlling Memory (Self-Study)

Objectives

- List techniques to reduce data storage.
- Describe how SAS stores numeric values.
- Describe how to safely reduce the space required to store numeric values in SAS data sets.

Techniques for Reducing Data Set Size

1. Store integers as reduced-length numerics.
2. Compress the data set.

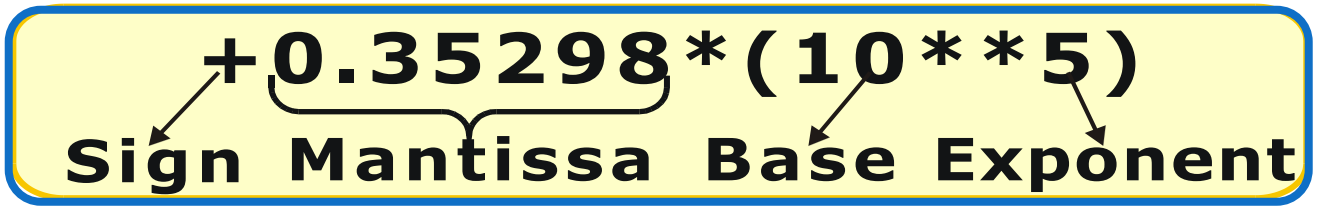
-  Reducing the size of a SAS data set reduces the I/O required to process it.

Characteristics of Numeric Variables

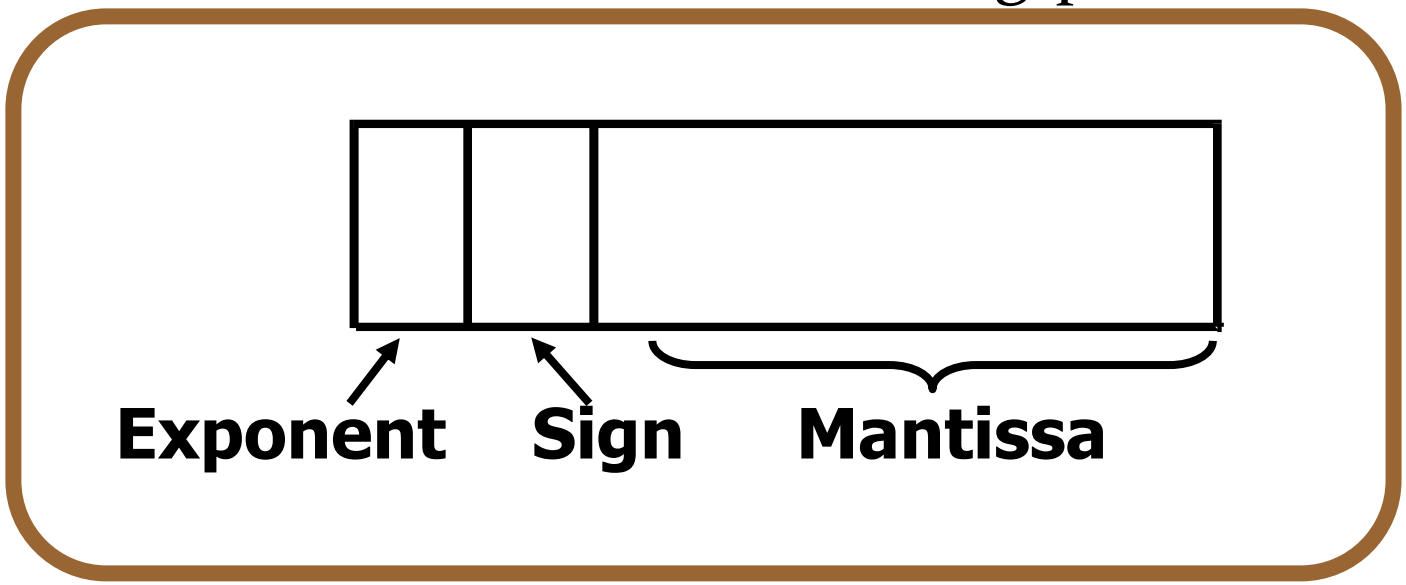
- Numeric variables have the following characteristics:
  - are stored as floating-point numbers in real-binary representation
    - store multiple digits per byte
    - use a minimum of one byte to store the sign and exponent of the value (depending on the operating environment) and use the remaining bytes to store the mantissa of the value
  - take 8 bytes of storage per variable, by default, but can be reduced in size
  - always have a length of 8 bytes in the PDV

Default Length of Numeric Variables

- The number 35,298 can be written as follows:



- SAS stores numeric variables in floating-point form:



Assigning the Length of Numeric Variables

- You can use a LENGTH statement to assign a length from 2 to 8 bytes to numeric variables.

```
data emps_short;
   length Street_ID 6
          Employee_ID Manager_ID 5
          Street_Number Employee_Hire_Date
          Employee_Term_Date Birth_Date
          Salary 4
          Dependents 3;
   merge employee_addresses
         employee_organization
         employee_payroll
         employee_phones;
   by Employee_ID;
run;
```

- If the variable is numeric, the length applies only to the output data set. If the variable is character, the length applies to the program data vector and the output data set.

p302d05

Comparing Results

- To determine whether the data sets **emps_short** and **emps** are equivalent, you can use the COMPARE procedure.

```
proc compare data=emps compare=emps_short;
run;
```


p302d06

Comparing Data Sets

```
                    The COMPARE Procedure
          Comparison of WORK.EMPS with WORK.EMPS_SHORT
                          (Method=EXACT)

                      Observation Summary

  Observation      Base   Compare

  First Obs           1         1
  Last  Obs         923       923

  Number of Observations in Common: 923.
  Total Number of Observations Read from WORK.EMPS: 923.
  Total Number of Observations Read from WORK.EMPS_SHORT: 923.

  Number of Observations with Some Compared Variables Unequal: 0.
  Number of Observations with All Compared Variables Equal: 923.
```

```
NOTE: No unequal values were found. All values compared are exactly equal.
```


p302d06

Possible Storage Lengths for Integer Values

- **Windows and UNIX**

Length (bytes)	Largest Integer Represented Exactly
3	8,192
4	2,097,152
5	536,870,912
6	137,438,953,472
7	35,184,372,088,832
8	9,007,199,254,740,992

Possible Storage Lengths for Integer Values

- **z/OS**

Length (bytes)	Largest Integer Represented Exactly
2	256
3	65,536
4	16,777,216
5	4,294,967,296
6	1,099,511,627,776
7	281,474,946,710,656
8	72,057,594,037,927,936

Assigning the Length of Numeric Variables

- The use of a numeric length less than 8 bytes does the following:
  - causes the number to be truncated to the specified length when the value is written to the SAS data set

    This reduces the number of bytes available for the mantissa, which reduces the precision of the number that can be accurately stored.
  - causes the number to be expanded to 8 bytes in the PDV when the data set is read by padding the mantissa with binary zeros

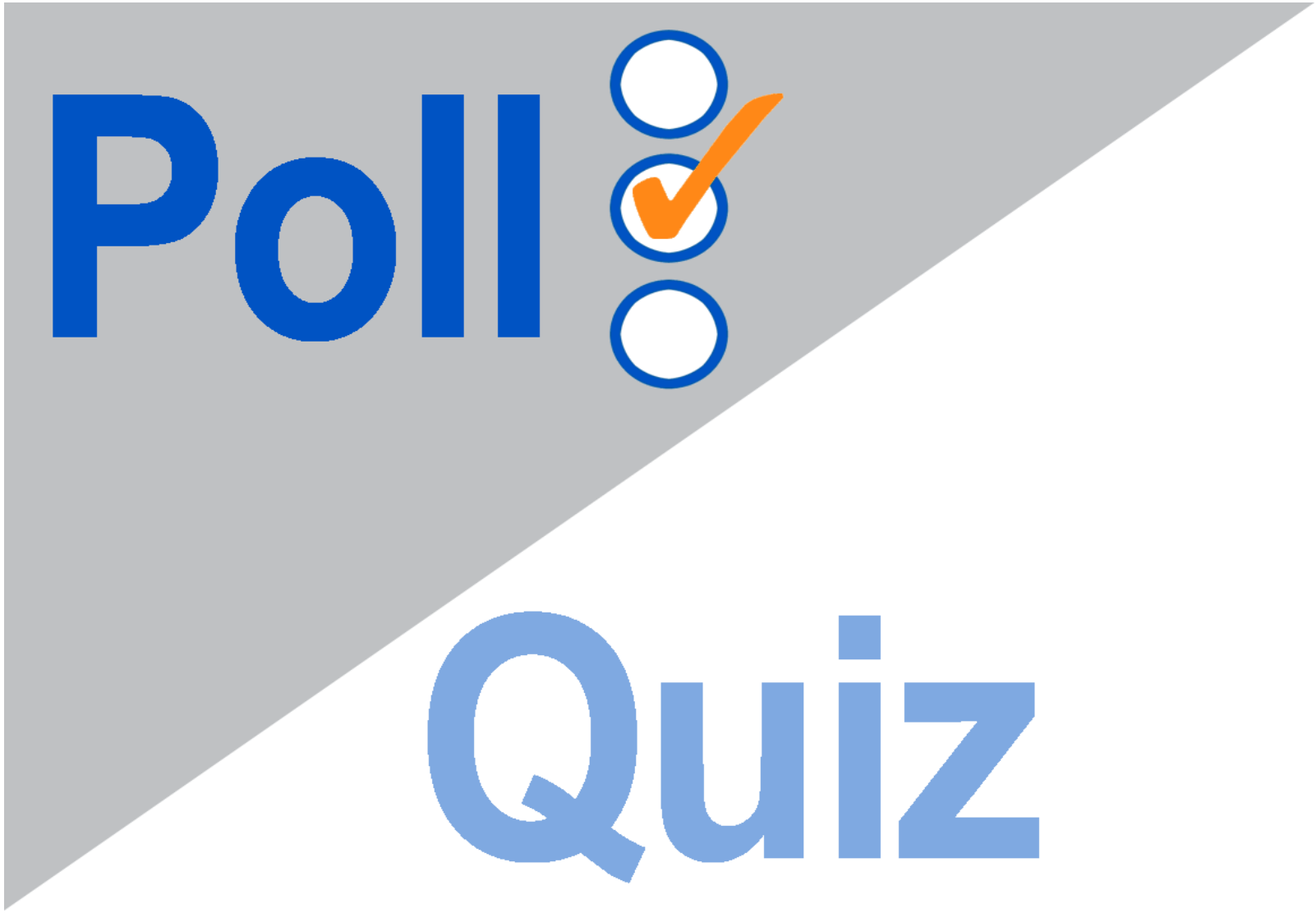
Numbers are always 8 bytes in length in the PDV.

Dangers of Reduced-Length Numeric Variables

- It is ***not*** recommended that you change the length of non-integer numeric variables.

```
data test;
   length x 4;
   X=1/10;
   Y=1/10;
run;

data _null_;
   set test;
   put X=;
   put Y=;
run;
```

p302a01


Poll
Quiz

2.02 Poll

Open the program **p302a01** and submit it.

Look at the log.

Are the values of X and Y equal?

- Yes
- No

2.02 Poll – Correct Answer

Open the program **p302a01** and submit it.

Look at the log.

Are the values of X and Y equal?

- ❍ Yes
- ❍ No

Numeric Precision

- Partial SAS Log (Windows)

```
7    data test;
8       length x 4;
9       X=1/10;
10      Y=1/10;
11   run;

NOTE: The data set WORK.TEST has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds



12
13   data _null_;
14      set test;
15      put X=;
16      put Y=;
17   run;

x=0.0999999642
y=0.1
NOTE: There were 1 observations read from the data set WORK.TEST.
NOTE: DATA statement used (Total process time):
      real time           0.03 seconds
      cpu time            0.00 seconds
```

Dangers of Reduced-Length Numeric Variables

- It is ***not*** recommended that you reduce the length of integer numeric variables inappropriately or that you reduce the length of variables that hold large integer numeric values. This example illustrates the effect of inappropriately reducing integer values.

```
data test;
   length X 3;
   X=8193;
run;

data _null_;
   set test;
   put X=;
run;
```

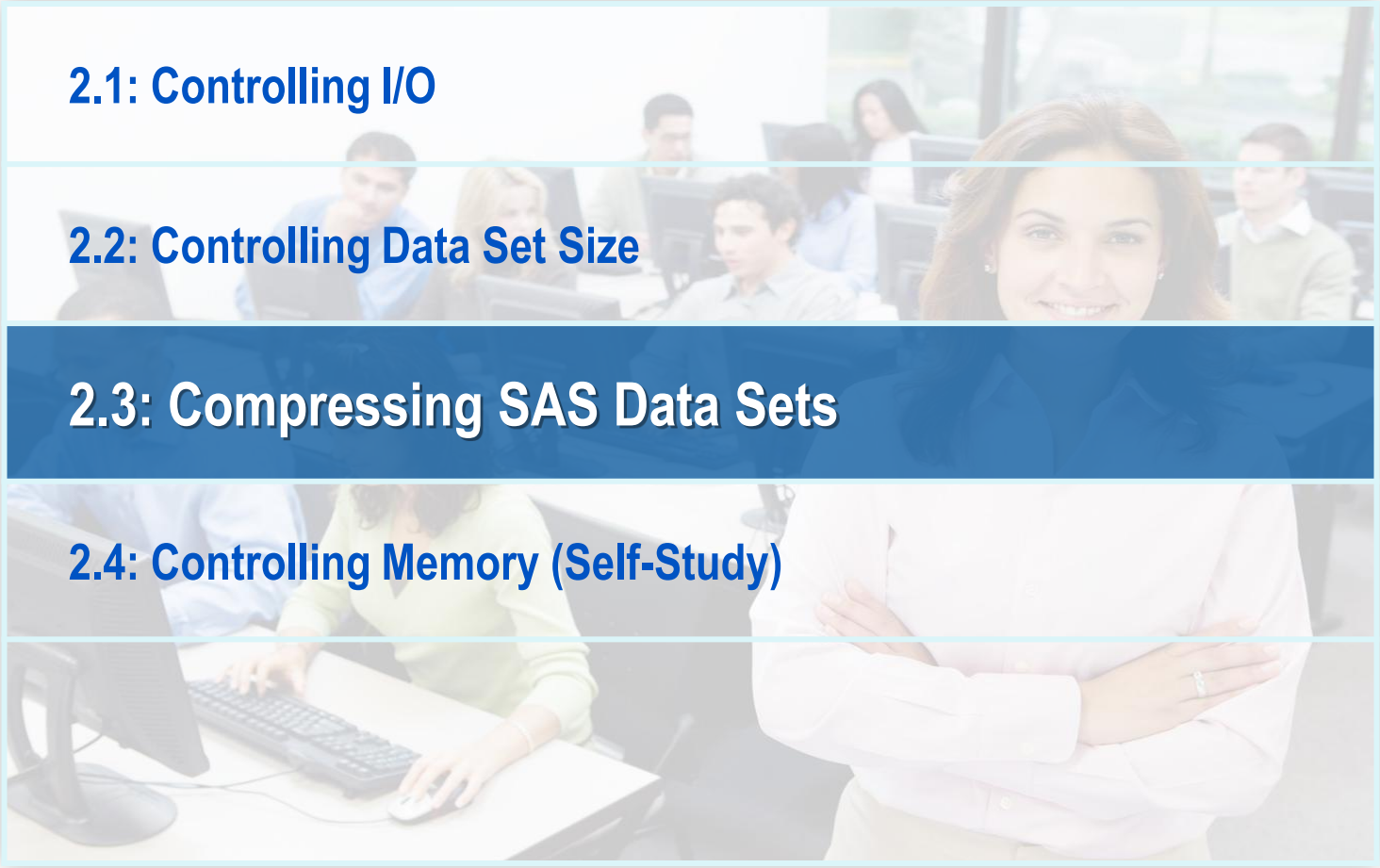
p302d07

Numeric Precision

```
120  data test;
121     length X 3;
122     X=8193;
123  run;

NOTE: The data set WORK.TEST has 1 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds

124
125  data _null_;
126     set test;
127     put X=;
128  run;

x=8192
NOTE: There were 1 observations read from the data set WORK.TEST.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```

Advantages and Disadvantages of Reduced-Length Numeric Variables

Advantages	Disadvantages
Conserves data storage space	Uses additional CPU to read
Requires less I/O to read	Can alter high-precision values such as non-integer and large integer values



2.1: Controlling I/O

2.2: Controlling Data Set Size

2.3: Compressing SAS Data Sets

2.4: Controlling Memory (Self-Study)

Objectives

- Define the structure of a compressed SAS data file.
- Create a compressed SAS data file.
- List the advantages and disadvantages of compression.

Simplified Uncompressed Data File Structure

Page 1	24 / 40 byte OH	Obs 1	Obs 2	Obs 3	Obs 4	Obs 5	*	1 bit / obs OH	Descriptor	
Page 2	24 / 40 byte OH	Obs 6	Obs 7	Obs 8	Obs 9	Obs 10	Obs 11	Obs 12	*	1 bit / obs OH
	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .
Page n	24 / 40 byte OH	Obs x	Obs y	Obs z	* Unused space					1 bit / obs OH

Uncompressed SAS Data File

The following are features of uncompressed SAS data files:

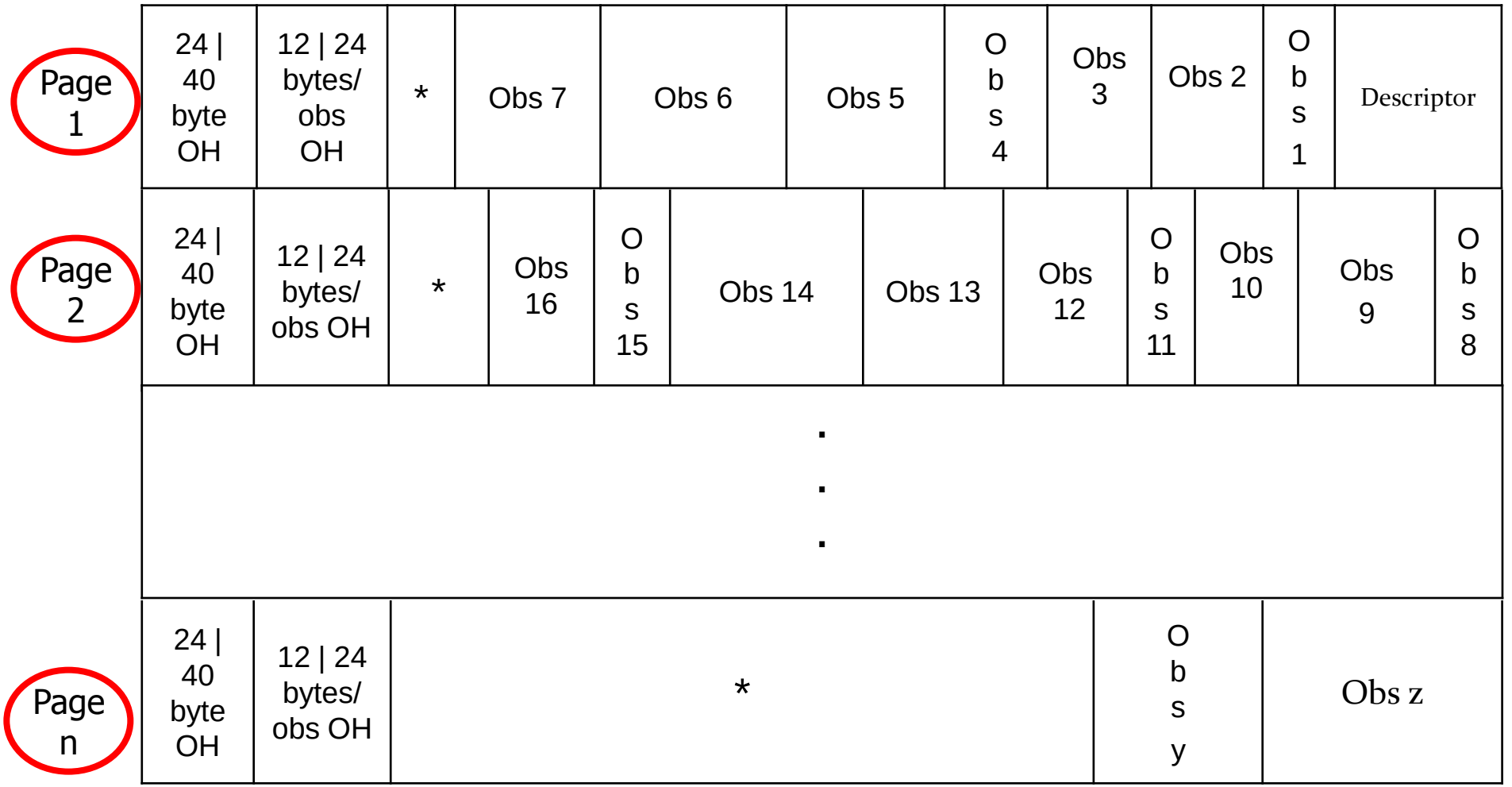
- All observations use the same number of bytes.
- Each variable occupies the same number of bytes in every observation.
- Character values are padded with blanks.
- Numeric values are padded with binary zeros.
- The descriptor portion of the data set uses part of the first data set page.

continued...

Uncompressed SAS Data File

- There is a 24-byte overhead at the beginning of each page on 32-bit systems.
- There is a 40-byte overhead at the beginning of each page on 64-bit systems.
- There is a 1-bit per observation overhead, rounded up to the nearest byte.
- New observations are added at the end of the file. If a new page is needed for a new observation, a whole data set page is added.
- Deleted observation space is never reused, unless the entire data file is rebuilt.

Simplified Structure of a Compressed Data Set



* Unused space

Compressed SAS Data File

Features of compressed SAS data files:

- Each observation is a single string of bytes. Variable types and boundaries are ignored.
- Each observation can have a different length.
- Consecutive repeating characters and numbers are collapsed into fewer bytes.
- If an updated observation is larger than its original size, it is stored on either the same data set page or on a different page with a pointer to the original page.
- The descriptor portion of the data set is stored at the end of the first data set page.

continued...

Compressed SAS Data File

- There is a 24-byte overhead at the beginning of each page on 32-bit systems.
- There is a 40-byte overhead at the beginning of each page on 64-bit systems.
- There is a 12-byte-per-observation overhead on 32-bit systems.
- There is a 24-byte-per-observation overhead on 64-bit systems.
- Deleted observation space can be reused if the REUSE=YES data set or system option was turned on when the SAS data file was compressed.

Compressing SAS Files

There are two different algorithms that can be used to compress files:

- the **RLE** (Run Length Encoding) compression algorithm (COMPRESS=YES | CHAR)
- the **RDC** (Ross Data Compression) algorithm (COMPRESS=BINARY)

The optimal algorithm depends on the characteristics of your data.

Creating a Compressed Data File

- To create a compressed data file, use the COMPRESS= output data set option or system option.
- General forms of the COMPRESS= options:

SAS-data-set(COMPRESS=NO | YES | CHAR | BINARY)

OPTIONS COMPRESS=NO | YES | CHAR | BINARY**;**

Comparing Compression Methods

- **COMPRESS=YES | CHAR**
  - is effective with character data that contains repeated characters (such as blanks).
- **COMPRESS=BINARY**
  - takes significantly more CPU time to uncompress than COMPRESS=YES | CHAR
  - is more efficient with observations greater than a 1000 bytes in length
  - can be very effective with numeric data
  - can be effective with character data that contains patterns, rather than simple repetitions.

Poll

Quiz

2.04 Quiz

Open the program **p302a02**.

1. Change the data set name to **empchar**. Add the COMPRESS=CHAR data set option to the DATA step and submit the program.
   By what percentage was the data set reduced or increased?
2. Change the data set name to **empbin**. Add the COMPRESS=BINARY data set option to the DATA step and submit the program.
   By what percentage was the data set reduced or increased?

2.04 Quiz – Correct Answer

Open the program **p302a02**.

1. Change the data set name to **empchar**. Add the COMPRESS=CHAR data set option to the DATA step and submit the program.
   By what percentage was the data set reduced or increased?

```
data empchar(compress=char);
    merge employee_addresses
          employee_organization
             employee_payroll
             employee_phones;
          by Employee_ID;
             run;
```


continued...

Using the COMPRESS=CHAR Option

- Partial SAS Log (Windows)

```
44
45   data empchar(compress=char);
46      merge employee_addresses employee_organization
47            employee_payroll employee_phones;
48      by Employee_ID;
49   run;

NOTE: There were 424 observations read from the data set WORK.EMPLOYEE_ADDRESSES.
NOTE: There were 424 observations read from the data set WORK.EMPLOYEE_ORGANIZATION.
NOTE: There were 424 observations read from the data set WORK.EMPLOYEE_PAYROLL.
NOTE: There were 923 observations read from the data set WORK.EMPLOYEE_PHONES.
NOTE: The data set WORK.EMPCHAR has 923 observations and 21 variables.
NOTE: Compressing data set WORK.EMPCHAR decreased size by 60.71 percent.
      Compressed is 11 pages; un-compressed would require 28 pages.
NOTE: DATA statement used (Total process time):
      real time           0.04 seconds
      cpu time            0.01 seconds
```

2.04 Quiz – Correct Answer

Open the program **p302a02**.

2. Change the data set name to **empbin**. Add the COMPRESS=BINARY data set option to the DATA step and submit the program.
   By what percentage was the data set reduced or increased?

```
data empbin(compress=binary);
   merge employee_addresses
         employee_organization
         employee_payroll
         employee_phones;
   by Employee_ID;
run;
```


continued...

Using the COMPRESS=BINARY Option

- Partial SAS Log (Windows)

```
50   data empbin(compress=binary);
51      merge employee_addresses employee_organization
52            employee_payroll employee_phones;
53      by Employee_ID;
54   run;

NOTE: There were 424 observations read from the data set WORK.EMPLOYEE_ADDRESSES.
NOTE: There were 424 observations read from the data set WORK.EMPLOYEE_ORGANIZATION.
NOTE: There were 424 observations read from the data set WORK.EMPLOYEE_PAYROLL.
NOTE: There were 923 observations read from the data set WORK.EMPLOYEE_PHONES.
NOTE: The data set WORK.EMPBIN has 923 observations and 21 variables.
NOTE: Compressing data set WORK.EMPBIN decreased size by 57.14 percent.
      Compressed is 12 pages; un-compressed would require 28 pages.
NOTE: DATA statement used (Total process time):
      real time           0.03 seconds
      cpu time            0.03 seconds
```

Summary of Compression Results

Data Set	Option Used	Algorithm Used	Number of Bytes	Decreased Size
employees	None	None	458,752	0%
empchar	YES | CHAR	RLE	180,224	60.71%
empbin	BINARY	RDC	196,608	57.14%

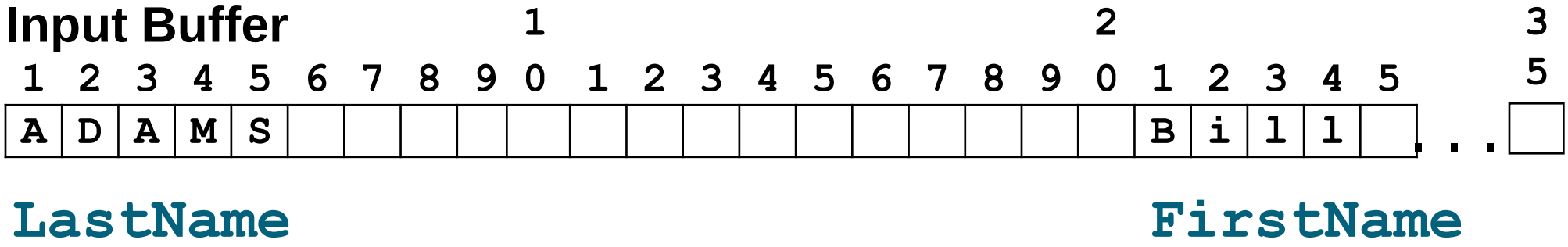
- Your results might differ, depending on SAS option settings and the operating system.

How SAS Compresses Data

- A SAS data file has these variables:

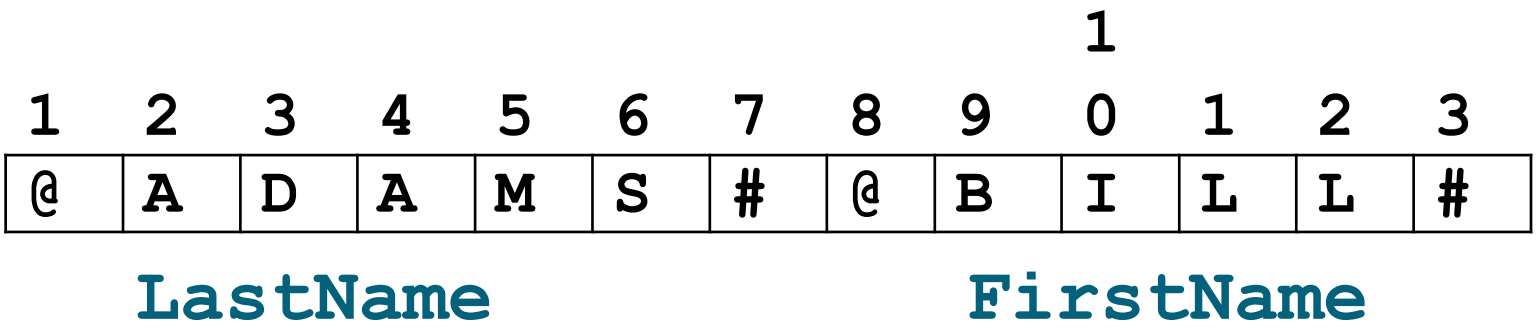
Name	Type	Length
LastName	Character	20
FirstName	Character	15

- In uncompressed form, all observations use 35 bytes for these two variables.



Using COMPRESS=CHAR | YES

- In run-length encoding compressed form, the **LastName** and **FirstName** values for this observation use only 13 bytes.



- The @ is a symbol that indicates how many uncompressed characters follow.
- The # is a symbol that indicates the number of blanks repeated at this point in the observation.

Using COMPRESS=BINARY

- Ross Data Compression uses both run-length encoding and sliding window compression.
- A SAS data set has these variables:

Name	Type	Length
Answer1	Numeric	8
...		
Answer200	Numeric	8

- In uncompressed form, the SAS data file resembles this:

Answer1	Answer2	Answer3	Answer4	Answer5	Answer6	...	Answer200
1	2	1	2	1	2	...	2
1	1	1	1	1	1	...	1
2	2	2	2	2	2	...	2

Using COMPRESS=BINARY

- In Ross Data Compression form, the first observation in the data file resembles this:

1	2	3	4	5	6	7	8	9
@	+ 1	1	#	@	+ 1	2	#	%

- The @ is a symbol that indicates how many uncompressed characters follow.
- The # is a symbol that indicates the number of binary zeros repeated at this point in the observation.
- The % is a symbol that indicates how many times these values are repeated.

Compression Dependencies

Some data sets do not compress well or at all.

- Because there is higher overhead for each observation, a data file can occupy more space in compressed form than in uncompressed form if the file has the following characteristics:
  - few repeated characters
  - small physical size
  - few missing values
  - short text strings

Compression Dependencies

```
1    data orders(compress=yes);
2       set orion.orders;
3    run;

NOTE: There were 490 observations read from the data set ORION.ORDERS.
NOTE: The data set WORK.ORDERS has 490 observations and 6 variables.
NOTE: Compressing data set WORK.ORDERS decreased size by 0.00 percent.
      Compressed is 7 pages; un-compressed would require 7 pages.
NOTE: DATA statement used (Total process time):
      real time           1.04 seconds
      cpu time            0.12 seconds
```

```
55   data orders(compress=binary);
                          56      set orion.orders;
                                  57   run;

    NOTE: There were 490 observations read from the data set ORION.ORDERS.
     NOTE: The data set WORK.ORDERS has 490 observations and 6 variables.
   NOTE: Compressing data set WORK.ORDERS increased size by 28.57 percent.
         Compressed is 9 pages; un-compressed would require 7 pages.
            NOTE: DATA statement used (Total process time):
                  real time           0.09 seconds
                  cpu time            0.09 seconds
```

p302d08

Compression Dependencies

- When you use the COMPRESS= data set option or the COMPRESS= system option, SAS knows the following:
  - the size of the overhead introduced by compression
  - the maximum size of an observation
- If the maximum size of the observation is less than the 12-byte (32-bit systems) or 24-byte (64-bit system) overhead introduced by compression, SAS does the following:
  - disables compression
  - creates an uncompressed data set
  - issues a note stating that the file was not compressed

Compression Dependencies

SAS Log (Windows)

```
18    data test(compress=yes);
19       x=1;
20    run;

NOTE: Compression was disabled for data set WORK.TEST because
      compression overhead would increase the size of the data set.
NOTE: The data set WORK.TEST has 1 observations and 1 variables.
NOTE: DATA statement used (Total process time):
      real time           0.00 seconds
      cpu time            0.00 seconds
```


p302d09

Compression Trade-Offs

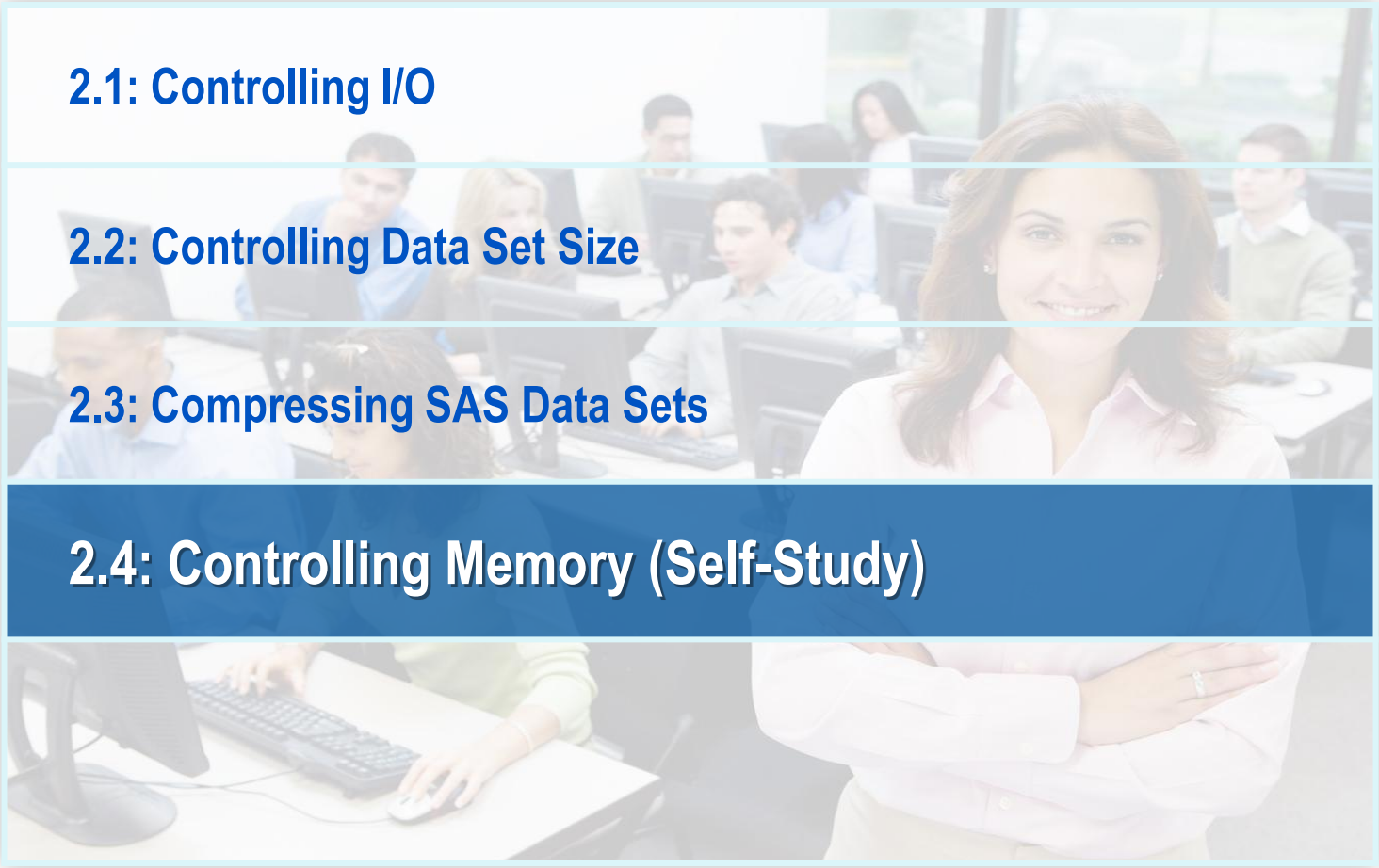
Uncompressed	Compressed
Usually requires more disk storage	Usually requires less disk storage
Requires less CPU time to prepare an observation for I/O	Requires more CPU time to prepare an observation for I/O
Uses more I/O operations	Uses fewer I/O operations

The savings in I/O operations greatly outweigh the increase in CPU time.

continued...

Compression Trade-Offs

Uncompressed	Compressed
An updated observation fits in its original location.	An updated observation might be moved from its original location.
Deleted observation space is never reused.	Deleted observation space can be reused.
New observations are always inserted at the end of the data file.	When REUSE=YES, new observations might not be inserted at the end of the data file.



2.1: Controlling I/O

2.2: Controlling Data Set Size

2.3: Compressing SAS Data Sets

2.4: Controlling Memory (Self-Study)

Objectives

- Investigate techniques for controlling memory.
- Use system options to specify the amount of available memory.

Comparing Memory, I/O, and CPU Resources

- The techniques that reduce CPU and I/O can increase memory usage.
  - Efficient use of the I/O subsystem uses larger buffers and/or multiple buffers.
  - These buffers share memory space with the other memory demands of your SAS session.

Benchmark carefully to balance the need to conserve memory with the need to reduce CPU
and I/O.

Reducing Memory Usage

- Use small data set page sizes when you create data sets that will be accessed in a sparse, random pattern.
- Use a single read buffer when the data is accessed randomly instead of sequentially.
- Use BY-group processing instead of CLASS statements in those procedures that support both, especially where you have pre-sorted data or can
  use an existing index.

Using the BY Statement

- Instead of using a CLASS statement to group data, you can use the BY statement to specify the variables whose values define the subgroup combinations for an analysis by a SAS procedure.

Using the BY Statement

- What are the differences between using a BY statement and using a CLASS statement in a procedure?

BY Statement	CLASS Statement
The data set must be sorted or indexed on the BY variables.	The data set does not need to be sorted or indexed on the CLASS variables.
BY-group processing holds only one BY group in memory at a time.	The CLASS statement accumulates aggregates for all CLASS groups simultaneously in memory.
A percentage for the entire report cannot be calculated with procedures such as the REPORT or TABULATE procedures.	A percentage for the entire report can be calculated with procedures such as the REPORT or TABULATE procedures.

PROC MEANS with a BY Statement

```
proc means data=orion.order_fact mean median
           maxdec=2;
   format Order_Date year4.;
   by Order_Date;
   var Quantity -- CostPrice_Per_Unit;
run;
```


p302d10

PROC MEANS with a BY Statement

- Partial PROC MEANS Output with a BY Statement

----------Date Order was placed by Customer=2003----------

The MEANS Procedure

Variable	Label	Mean	Median
Quantity	Quantity Ordered	1.82	2.00
Total_Retail_Price	Total Retail Price for This Product	177.59	105.95
CostPrice_Per_Unit	Cost Price Per Unit	42.72	31.80

---------- Date Order was placed by Customer=2004 ----------

Variable	Label	Mean	Median
Quantity	Quantity Ordered	1.69	1.00
Total_Retail_Price	Total Retail Price for This Product	146.13	85.65
CostPrice_Per_Unit	Cost Price Per Unit	37.10	25.68

PROC MEANS with a CLASS Statement

```
proc means data=orion.order_fact mean median
           maxdec=2;
   format Order_Date year4.;
   class Order_Date;
   var Quantity -- CostPrice_Per_Unit;
run;
```


p302d10

PROC MEANS with a CLASS Statement

The MEANS Procedure

Date Order was placed by Customer	N Obs	Variable	Label	Mean	Median
2003	128	Quantity	Quantity Ordered	1.82	2.00
		Total_Retail_Price	Total Retail Price for This Product	177.59	105.95
		CostPrice_Per_Unit	Cost Price Per Unit	42.72	31.80
2004	108	Quantity	Quantity Ordered	1.69	1.00
		Total_Retail_Price	Total Retail Price for This Product	146.13	85.65
		CostPrice_Per_Unit	Cost Price Per Unit	37.10	25.68
2005	90	Quantity	Quantity Ordered	1.70	1.00
		Total_Retail_Price	Total Retail Price for This Product	187.20	77.00
		CostPrice_Per_Unit	Cost Price Per Unit	49.45	25.20
2006	143	Quantity	Quantity Ordered	1.57	1.00
		Total_Retail_Price	Total Retail Price for This Product	149.70	92.80
		CostPrice_Per_Unit	Cost Price Per Unit	44.25	29.95
2007	148	Quantity	Quantity Ordered	1.93	2.00
		Total_Retail_Price	Total Retail Price for This Product	157.49	80.95
		CostPrice_Per_Unit	Cost Price Per Unit	37.53	21.35

Using Memory System Options

- The operating environment will use disk-based virtual memory when physical memory is depleted. This adversely affects performance.

- Use these SAS system options to limit the amount of memory used by SAS:
  - REALMEMSIZE=
  - MEMSIZE=

Set the REALMEMSIZE= and MEMSIZE= options to values below the amount of available physical memory to prevent page thrashing or memory swapping.

Using System Options to Control Memory

- The total amount of memory that a SAS job can consume is limited by the MEMSIZE= **invocation** system option.

```
-MEMSIZE=n | n K | n M | n G | hexX | MIN | MAX
```

Using the MEMSIZE= Option

- The MEMSIZE= option places a limit on the total amount of memory that SAS dynamically allocates at any time. This memory is supported by a combination of real memory and space for utility files on disk.
- Increase this option's value in small increments until you find your optimal value.

-  SAS does not automatically reserve or allocate the amount of memory that you specify in the MEMSIZE= system option. SAS will use only as much memory as it needs to complete a process.

Using the REALMEMSIZE= System Option

- Some SAS procedures use the REALMEMSIZE= **invocation** option to specify how much memory * the procedure can allocate and use without inducing excessive page swapping.

```
-REALMEMSIZE=n | nK | nM | nG | hexX | MIN | MAX
```

- The REALMEMSIZE= option should never be set beyond the amount of real memory.
  - In UNIX and z/OS, REALMEMSIZE= specifies real (physical) memory. In Windows, REALMEMSIZE= specifies virtual memory.

3. Pokročilé programování v SAS – indexy, …

3.1 Creating an Index

3.2 Using an Index

3.3 Creating a Sample Data Set (Self-Study)

Objectives

- Define indexes.
- List the uses of indexes.
- Use the DATA step to create indexes.
- Use PROC DATASETS to create and maintain indexes.
- Use PROC SQL to create and maintain indexes.

Simplified Index File

The index file consists of entries that are organized in a tree structure and connected by pointers.

Partial Listing of
orion.sales_history

Customer_ID	Employee_ID	. . .
14958	121031	. . .
14844	121042	. . .
14864	99999999	. . .
14909	120436	. . .
14862	120481	. . .
14853	120454	. . .
14838	121039	. . .
14842	121051	. . .
14815	99999999	. . .
14797	120604	. . .
.	.	.
.	.	.
.	.	.

Simplified Index

Customer_ID Key Value	Record Identifier (RID) Page(obs, obs, ...)
4006	17(85)
4021	17(89)
4059	17(90)
4063	17(80, 86)
. . .	
14958	1(1, 24)
14972	1(14)
. . .	

The Purpose of Indexes

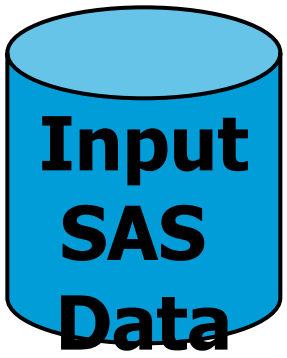
- Indexes can provide direct access to observations in SAS data sets to accomplish the following:
  - yield faster access to small subsets (WHERE)
  - return observations in sorted order (BY)
  - perform table lookup operations (SET with KEY=)
  - join observations (PROC SQL)
  - modify observations (MODIFY with KEY=)

Why Use an Index?

- How is data processed if the input data is not indexed?

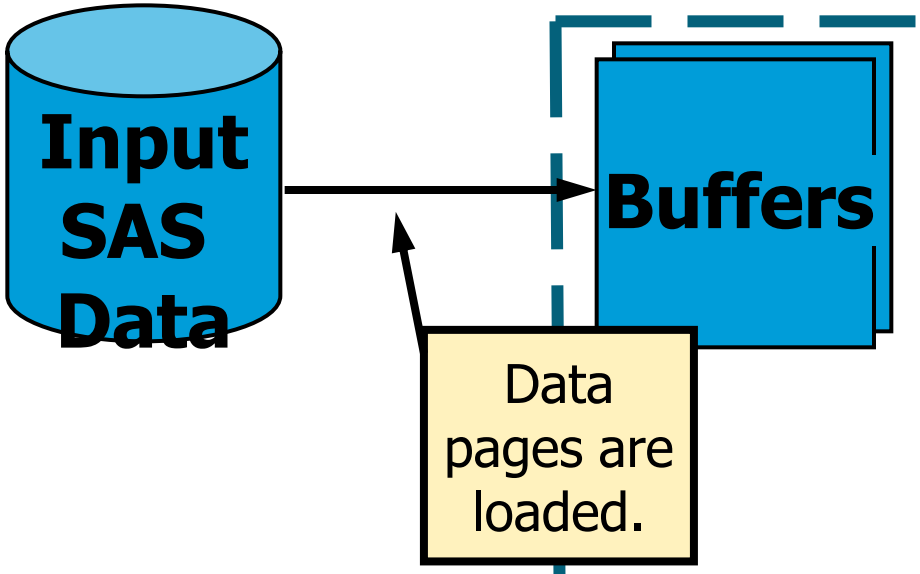
```
data customer14958;
   set orion.sales_history;
   where Customer_ID=14958;
run;
```

Reading SAS Data Sets *without* an Index



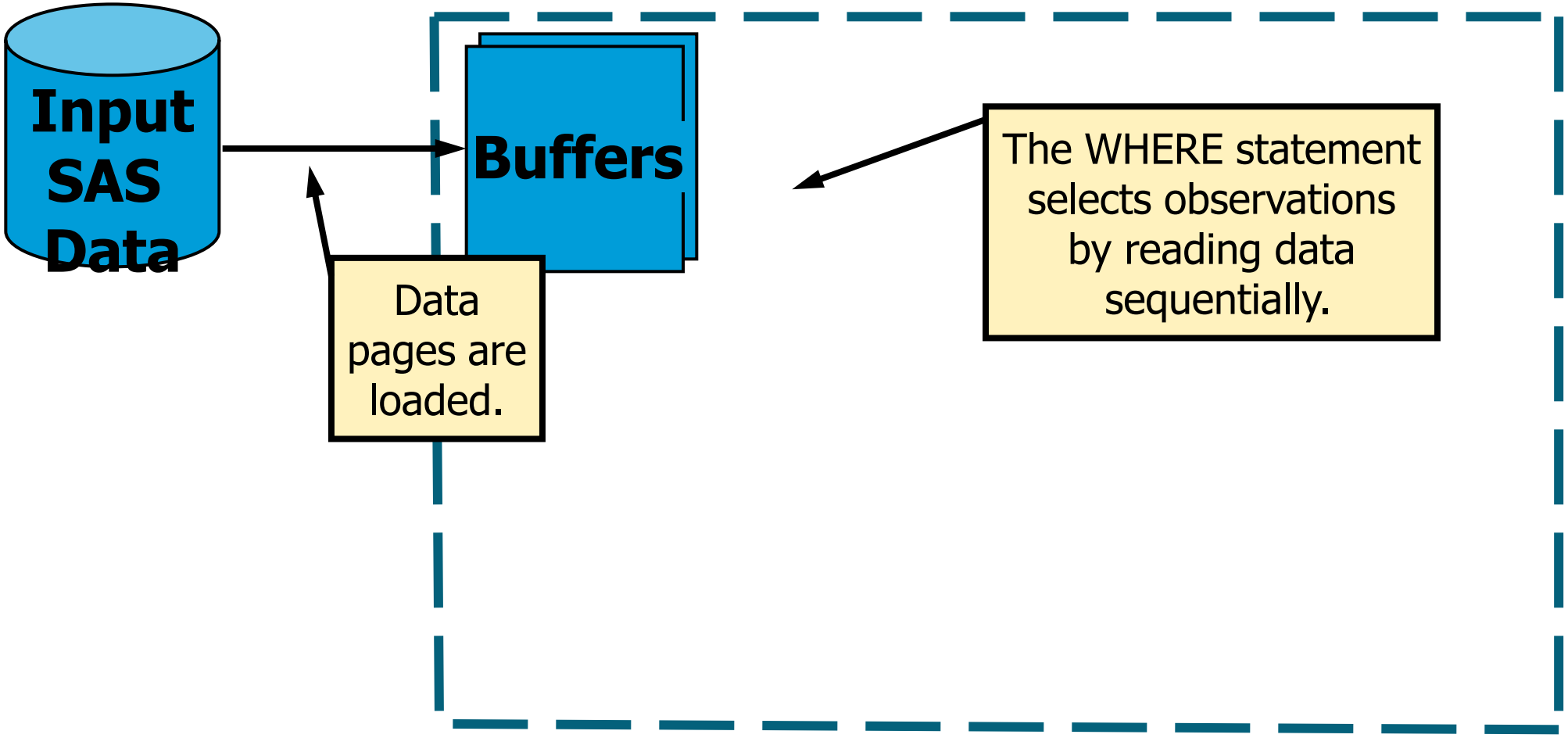
...

Reading SAS Data Sets *without* an Index

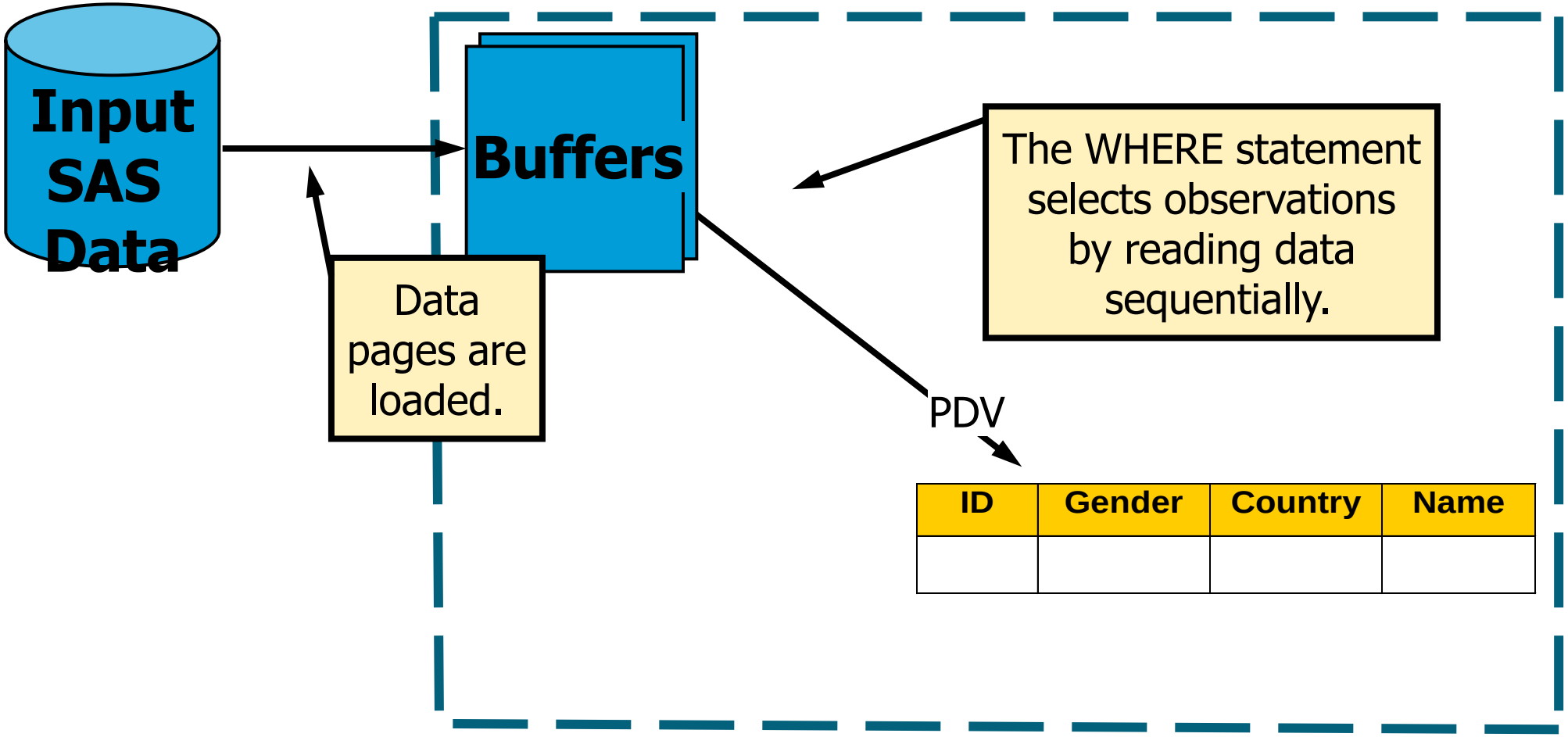


- **The number of buffers available affects the I/O.**

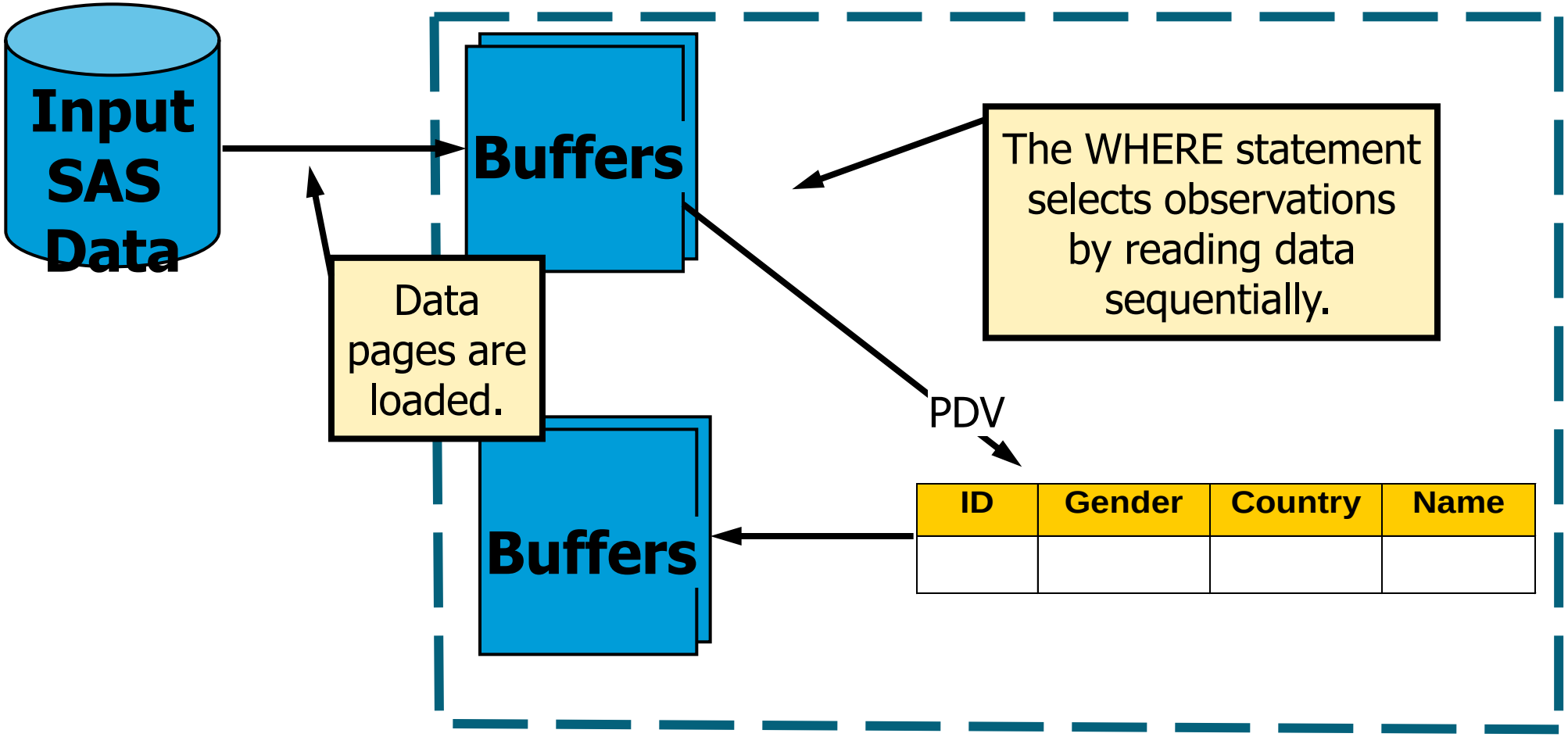
Reading SAS Data Sets *without* an Index



Reading SAS Data Sets *without* an Index

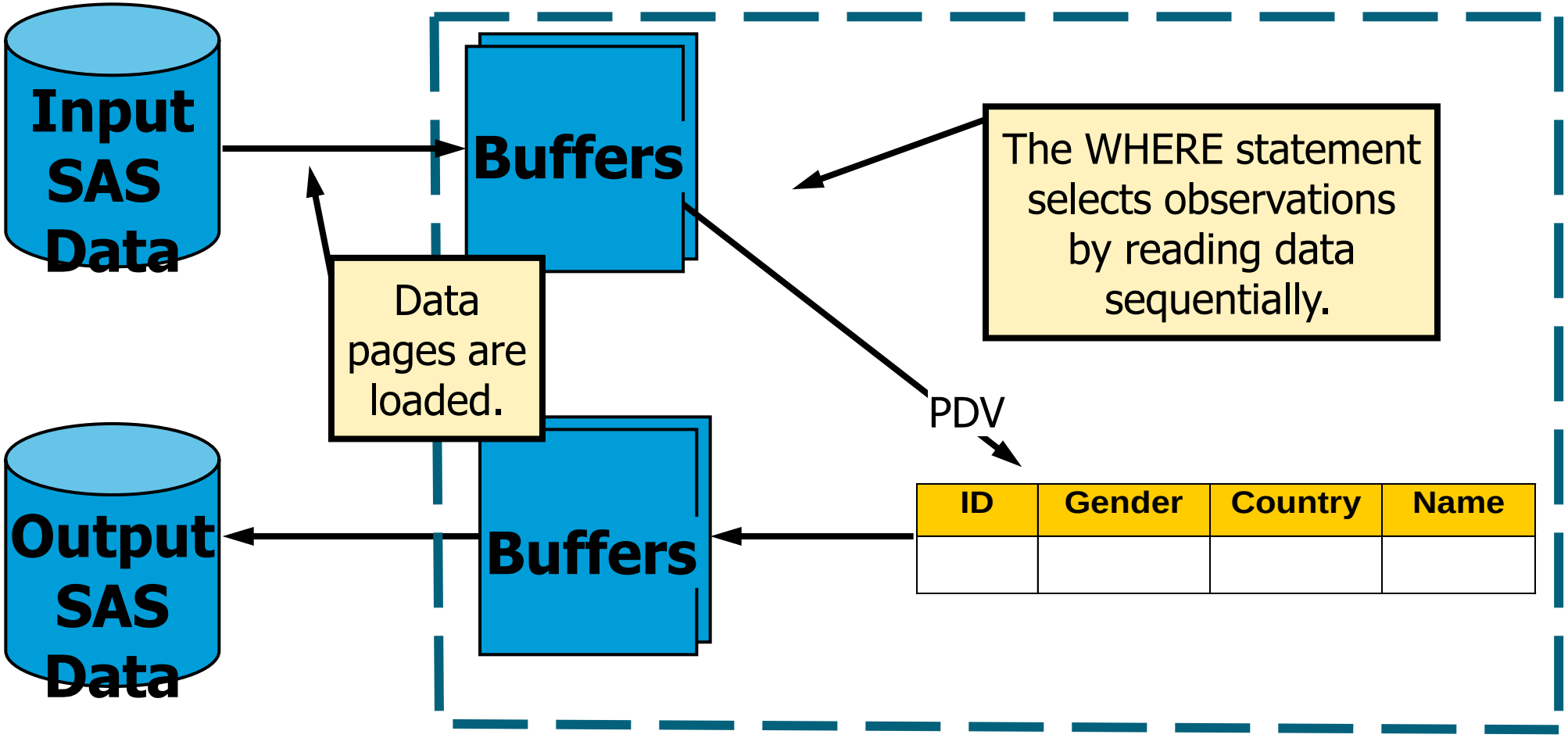


Reading SAS Data Sets *without* an Index



...

Reading SAS Data Sets *without* an Index

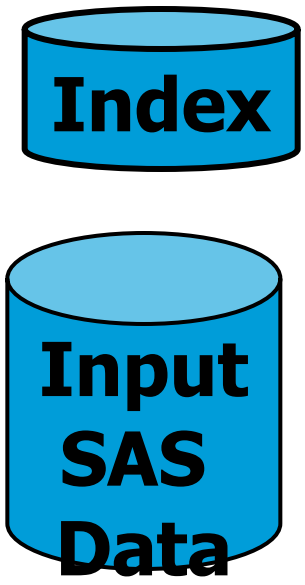


Why Use an Index?

- How is data processed if the input data is indexed?

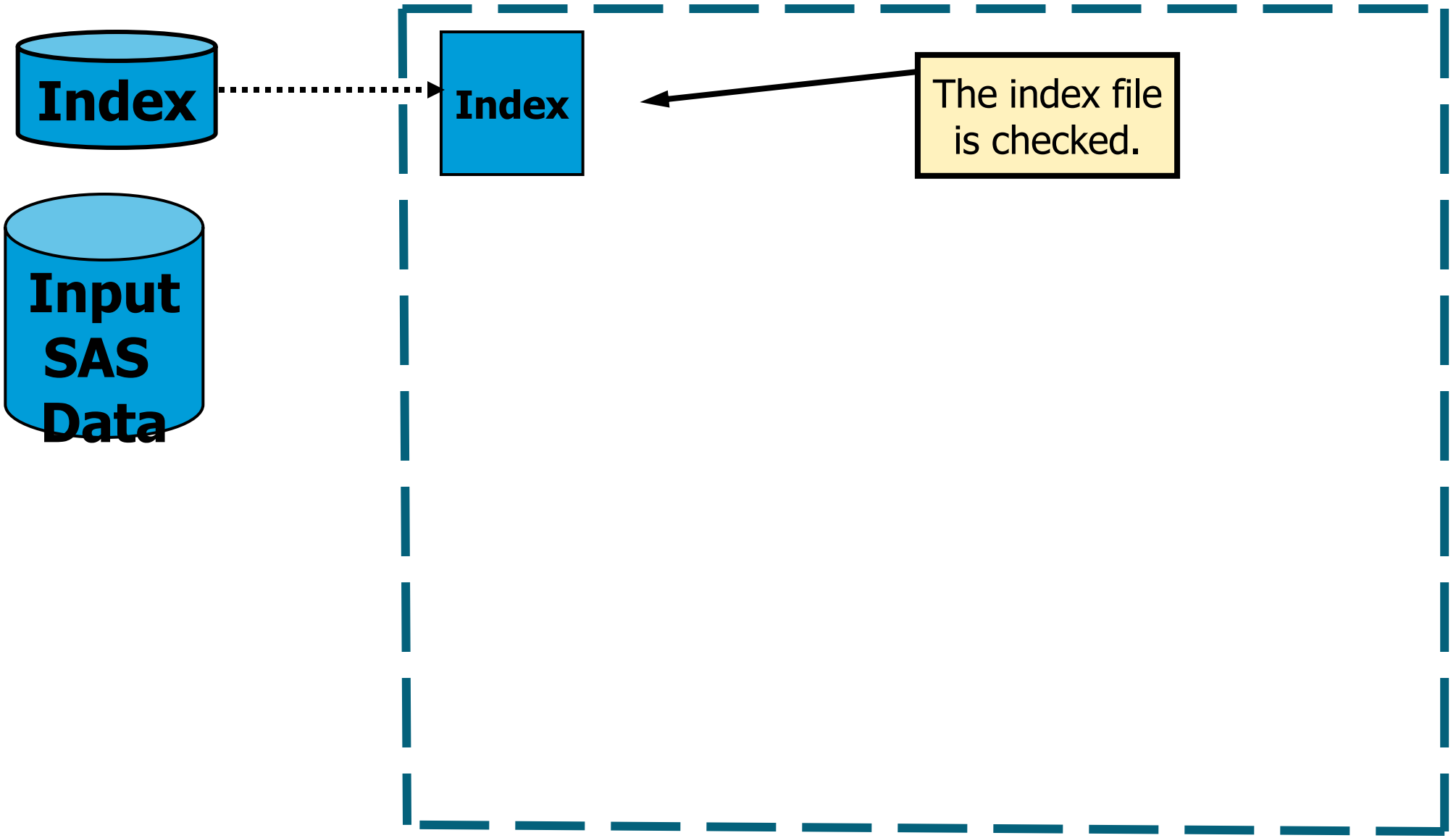
```
data customer14958;
   set orion.sales_history;
   where Customer_ID=14958;
run;
```

Reading SAS Data Sets *with* an Index



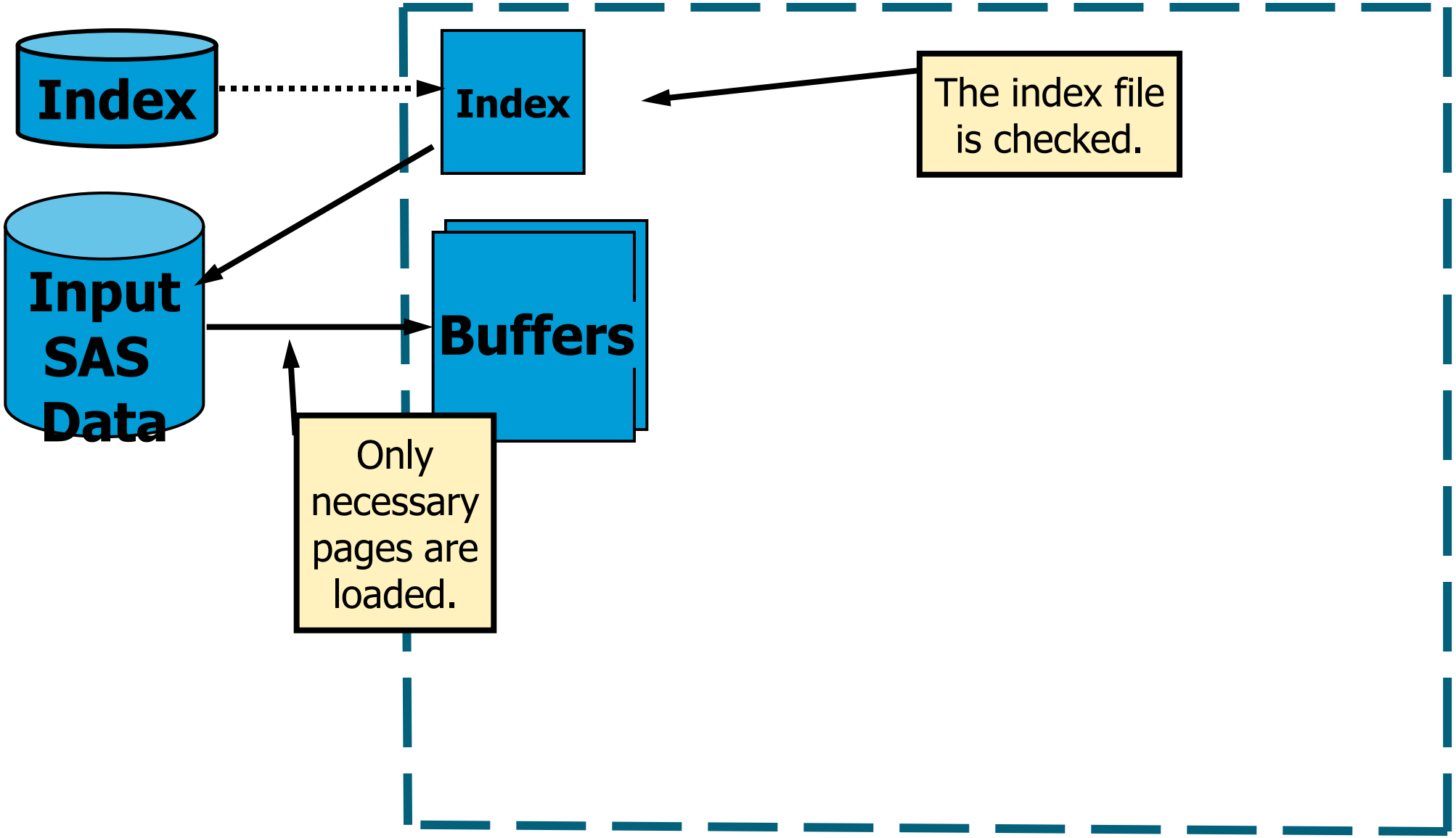
...

Reading SAS Data Sets *with* an Index



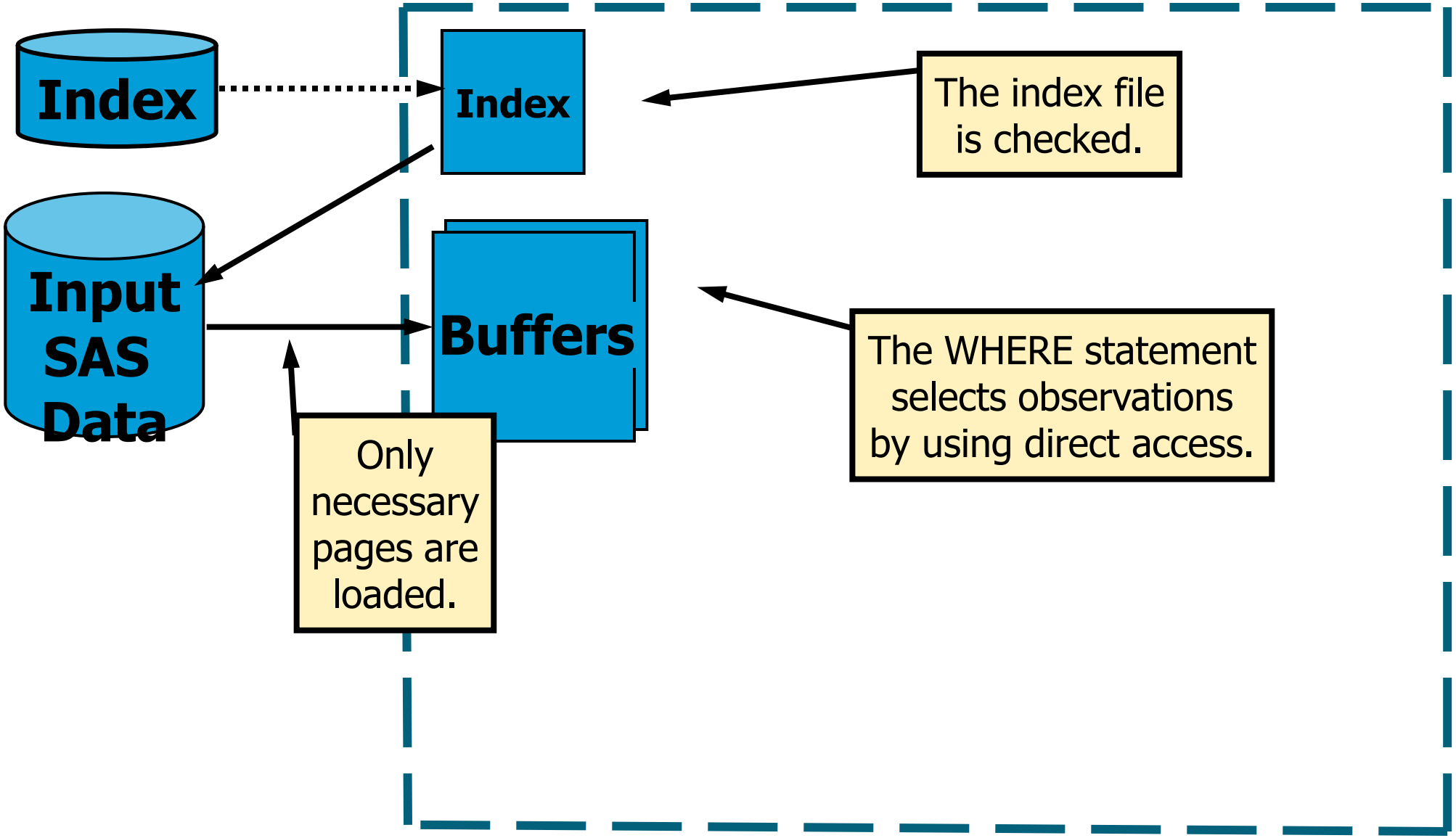
...

Reading SAS Data Sets *with* an Index



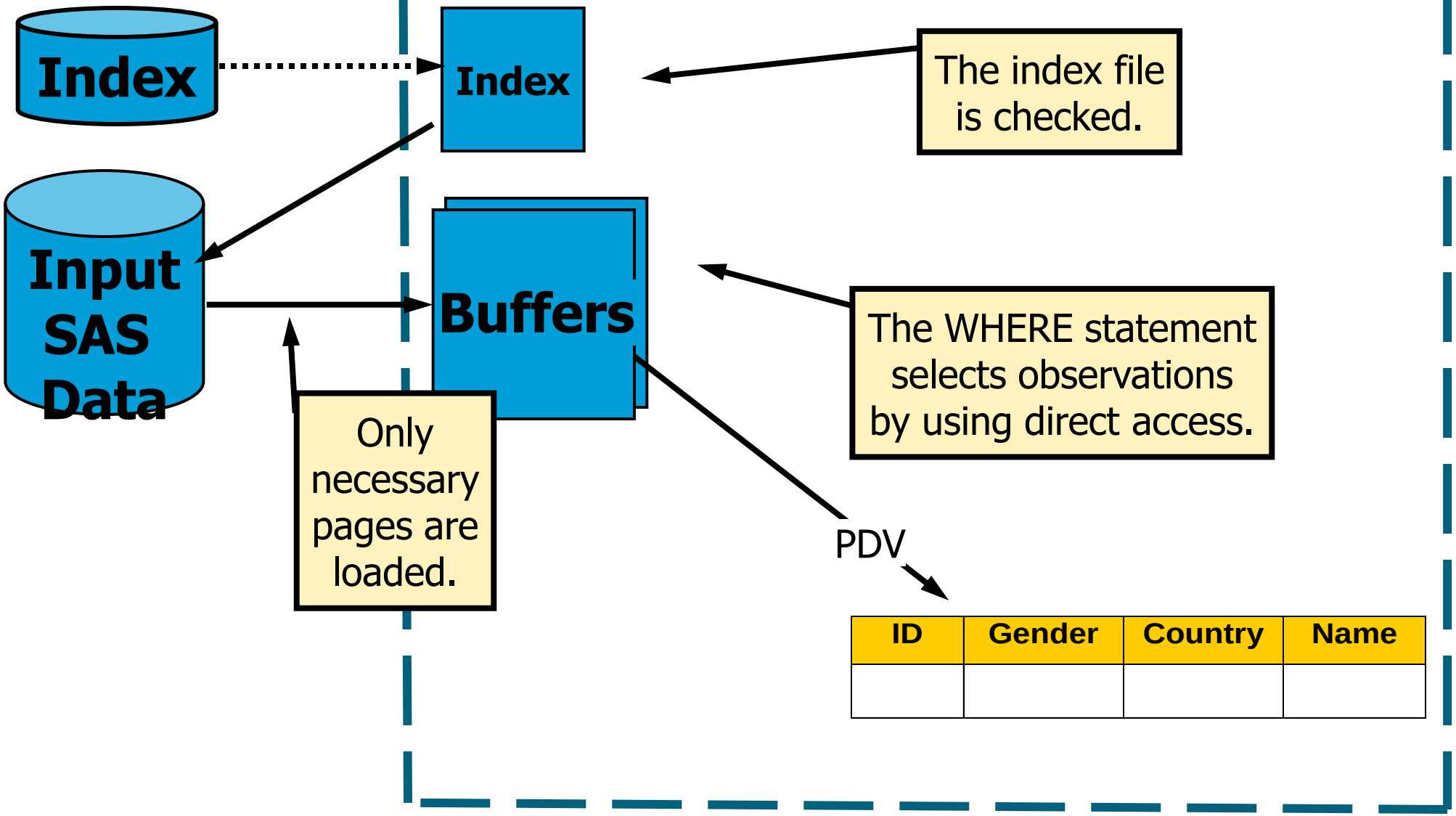
...

Reading SAS Data Sets *with* an Index



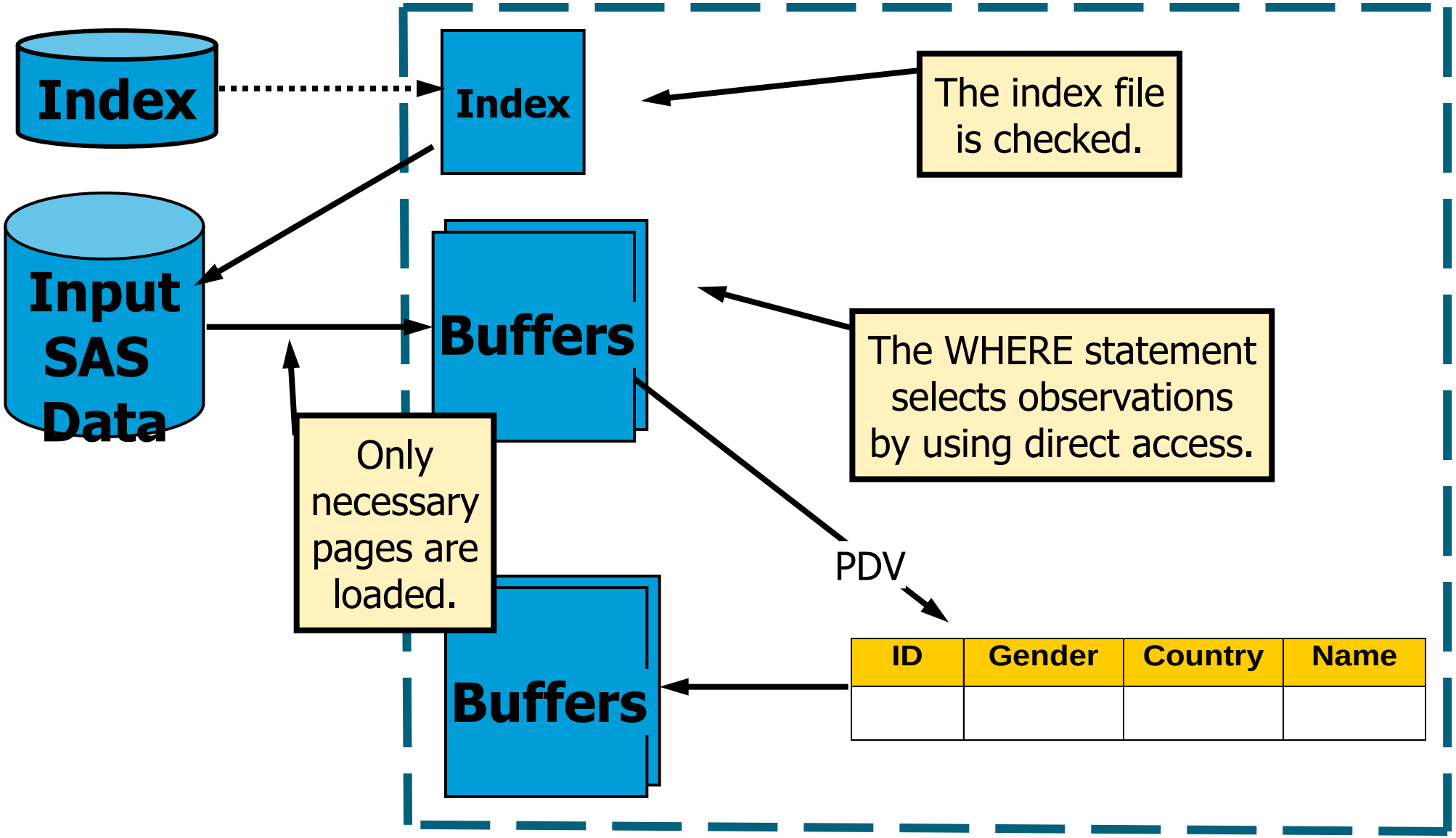
...

Reading SAS Data Sets *with* an Index



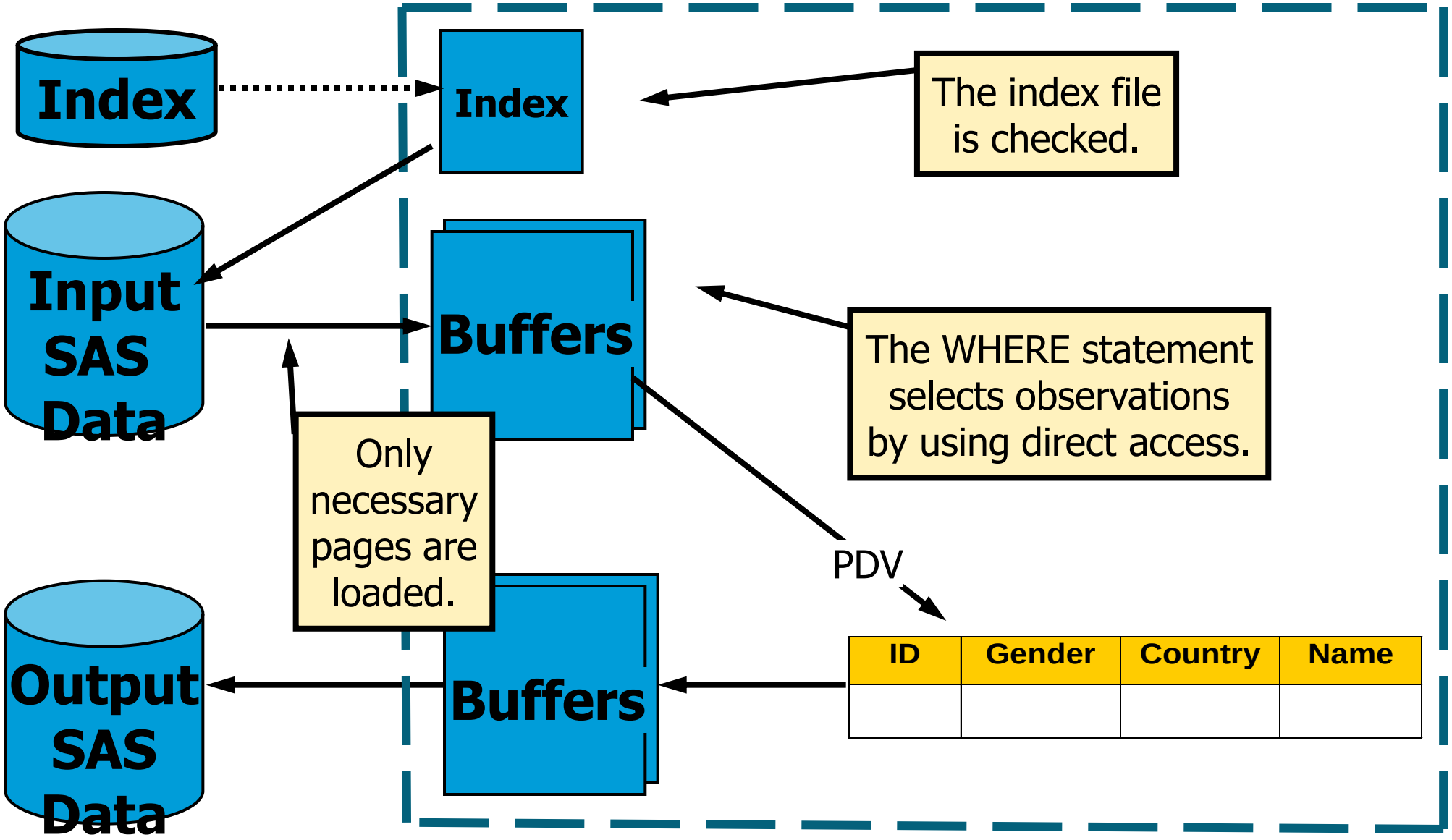
...

Reading SAS Data Sets *with* an Index

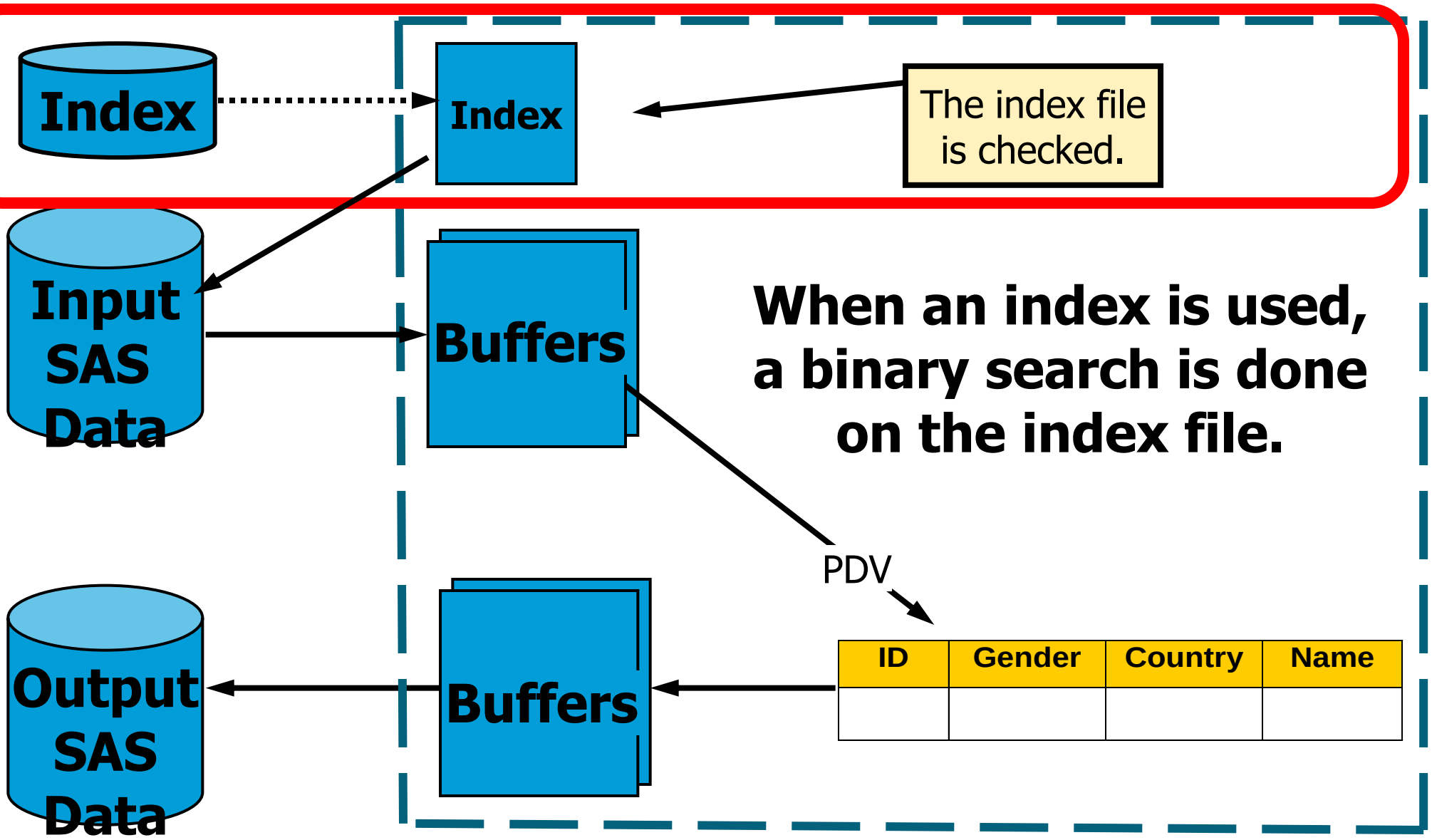


...

Reading SAS Data Sets *with* an Index



How Is the Index File Checked?



Using a Binary Search

Partial Listing of
orion.sales_history

RID	Customer_ID	Employee_ID	. . .
1	14958	121031	. . .
2	14844	121042	. . .
3	14864	99999999	. . .
4	14909	120436	. . .
.	.	.	
23	14844	121042	. . .
24	14958	121031	. . .
25	14821	120918	. . .
.	.	.	
.	.	.	. . .
.	.	.	

```
where Customer_ID=14958;
```

Simplified Index File

Customer_ID Key Value	Record Identifier (RID) Page(obs, obs, ...)
4006	17(85)
4021	17(89)
4059	17(90)
4063	17(80, 86)
. . .	
14958	1(1, 24)
14972	1(14)
. . .	

...

Using a Binary Search

Partial Listing of
orion.sales_history

RID	Customer_I	Empl	.
1			
2			
3			.
4		6	. . .
.	.	.	
23	14844	121042	. . .
24	14958	121031	. . .
25	14821	120918	. . .
.	.	.	
.	.	.	. . .
.	.	.	

Is 14958 in the top half or the bottom half?

```
where Customer_ID=14958;
```

Simplified Index File

Customer_ ID Key Value	Record Identifier (RID) Page(obs, obs, ...)
4006	17(85)
4021	17(89)
4059	17(90)
4063	17(80, 86)
. . .	
14958	1(1, 24)
14972	1(14)
. . .	

...

Using a Binary Search

Partial Listing of
orion.sales_history

RID	Customer_I	Empl	.
1			
2			
3			.
4		6	. . .
.	.	.	
23	14844	121042	. . .
24	14958	121031	. . .
25	14821	120918	. . .
.	.	.	
.	.	.	. . .
.	.	.	

Is 14958 in the top half or the bottom half?

```
where Customer_ID=14958;
```

Simplified Index File

Customer_ ID Key Value	Record Identifier (RID) Page(obs, obs, ...)
4006	17(85)
4021	17(89)
4059	17(90)
4063	17(80, 86)
. . .	
14958	1(1, 24)
14972	1(14)
. . .	

...

Using a Binary Search

Partial Listing of
orion.sales_history

RID	Customer_ID	Employee_ID	. . .
1	14958	121031	. . .
2	14844	121042	. . .
3	14864	99999999	. . .
4	14909	120436	. . .
.	.	.	
23	14844	121042	. . .
24	14958	121031	. . .
25	14821	120918	. . .
.	.	.	
.	.	.	. . .
.	.	.	

```
where Customer_ID=14958;
```

Simplified Index File

Customer_ID Key Value	Record Identifier (RID) Page(obs, obs, ...)
4006	17(85)
4021	17(89)
4059	17(90)
4063	17(80, 86)
. . .	
14958	1(1, 24)
14972	1(14)
. . .	

...

Using a Binary Search

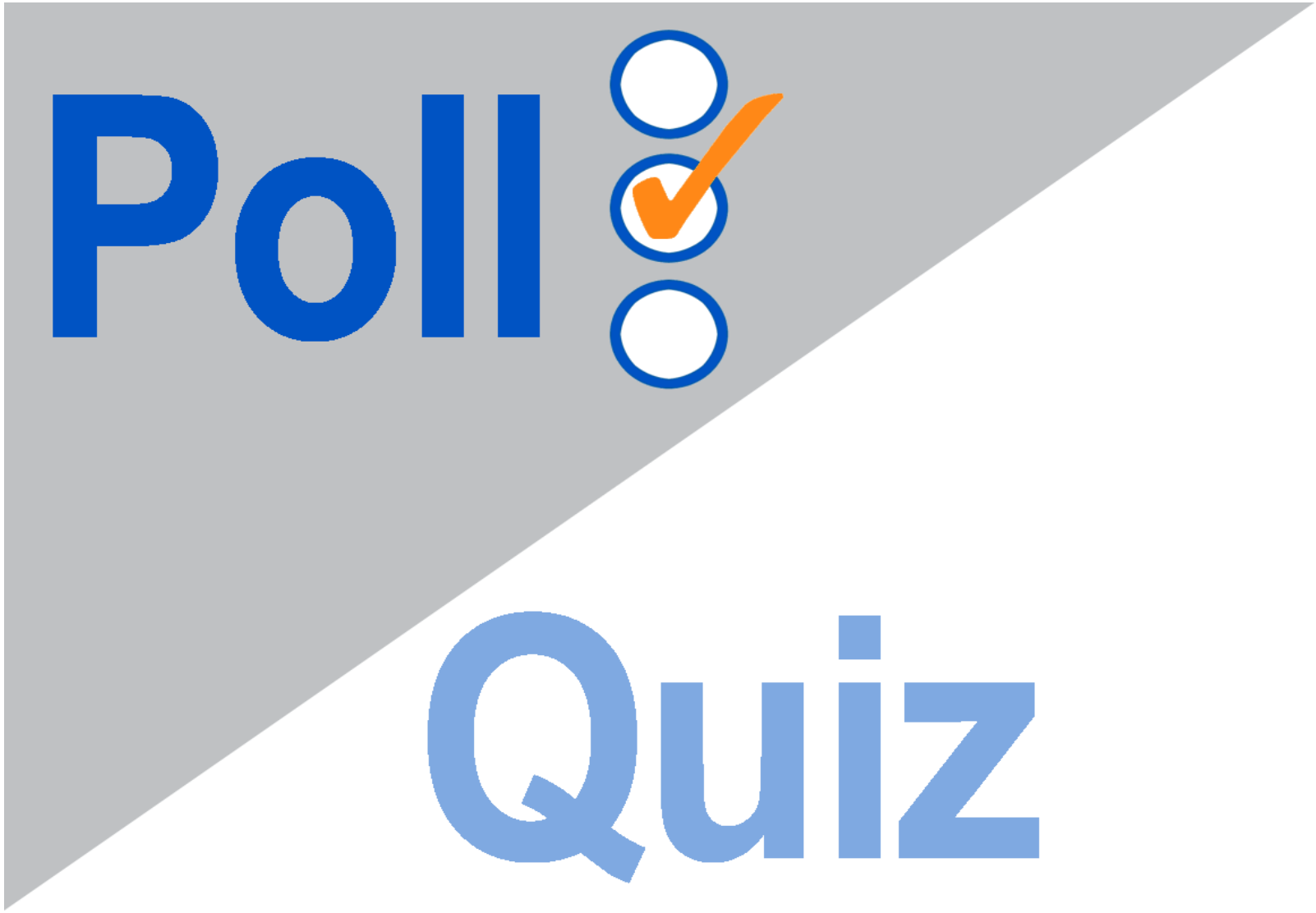
Partial Listing of
orion.sales_history

RID	Customer_ID	Employee_ID	. . .
1	14958	121031	. . .
2	14844	121042	. . .
3	14864	99999999	. . .
4	14909	120436	. . .
.	.	.	
23	14844	121042	. . .
24	14958	121031	. . .
25	14821	120918	. . .
.	.	.	
.	.	.	. . .
.	.	.	

```
where Customer_ID=14958;
```

Simplified Index File

Customer_ID Key Value	Record Identifier (RID) Page(obs, obs, ...)
4006	17(85)
4021	17(89)
4059	17(90)
4063	17(80, 86)
. . .	
14958	1(1, 24)
14972	1(14)
. . .	


Poll
Quiz

3.02 Multiple Choice Poll

- If a WHERE statement uses an index to retrieve a small subset of data, which of these resources is conserved?

a. I/O

b. Disk space

c. Memory

d. Programmer time

3.02 Multiple Choice Poll – Correct Answer

- If a WHERE statement uses an index to retrieve a small subset of data, which of these resources is conserved?

a. I/O

b. Disk space

c. Memory

d. Programmer time

Business Scenario

- The SAS data set **orion.sales_history** is often queried with a WHERE statement.

Partial Listing of **orion.sales_history**

Customer_ ID	. . .	Order_ ID	Order_ Type	Product_ID	. . .	Product_ Group	. . .
14958	. . .	1230016296	1	210200600078	. . .	N.D. Gear, Kids	. . .
14844	. . .	1230096476	1	220100100354	. . .	Eclipse Clothing	. . .
14864	. . .	1230028104	2	240600100115	. . .	Bathing Suits	. . .
14909	. . .	1230044374	1	240100200001	. . .	Darts	. . .
14862	. . .	1230021668	1	240500200056	. . .	Running Clothes	. . .
14853	. . .	1230021653	1	220200200085	. . .	Shoes	. . .
14838	. . .	1230140184	1	220100300042	. . .	Knitwear	. . .
14842	. . .	1230025285	1	240200100053	. . .	Golf	. . .
14815	. . .	1230109468	3	230100700004	. . .	Tents	. . .
14797	. . .	1230168587	1	220101000004	. . .	Shorts	. . .

Business Scenario

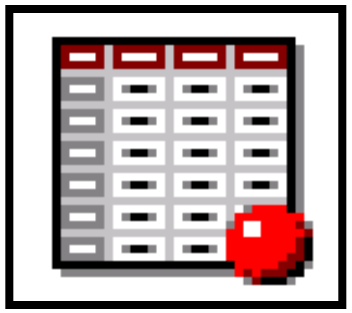
- You need to create three indexes on the most frequently used subsetting columns.

Index Name	Index Variables
Customer_ID	Customer_ID
Product_Group	Product_Group
SaleID	Order_ID Product_ID

Partial Listing of **orion.sales_history**

Customer_ ID	. . .	Order_ ID	Order_ Type	Product_ID	. . .	Product_ Group	. . .
14958	. . .	1230016296	1	210200600078	. . .	N.D. Gear, Kids	. . .
14844	. . .	1230096476	1	220100100354	. . .	Eclipse Clothing	. . .

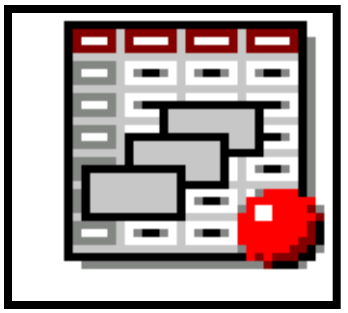
Creating an Index



Customer_ID
Order_ID
Product_ID
Product_Group

Key variables in **orion.sales_history**

`sales_history.sas7bdat`



Customer_ID
Product_Group
SaleID

Indexes in the index file for **orion.sales_history**

`sales_history.sas7bndx`

Directory-based Index File Naming Conventions

Index Name	Index Variables	Index Type
Customer_ID	Customer_ID	Simple
Product_Group	Product_Group	Simple
SaleID	Order_ID Product_ID	Composite

Index Terminology

- There are two types of indexes.

Type	Based On	Name	Example
Simple	the value of only one variable	automatically given the same name as its key variable	`Customer_ID` `Product_Group`
Composite	the values of more than one variable concatenated to form a single value	must be given a name that is not the same as any variable or existing index	`SaleID= (Order_ID Product_ID)`

Index Terminology

- Index options include the following:

UNIQUE	Values of the key variable(s) must be unique. This option prevents an observation with a duplicate value for the key variable(s) from being added to the data set.

Partial Listing of **orion.sales_history**

Customer_ID	Employee_ID	. . .	Order_ID	Order_Type	Product_ID	Quantity	. . .
14958	121031	. . .	1230016296	1	210200600078	1	. . .
14844	121042	. . .	1230096476	1	220100100354	1	. . .
14864	99999999	. . .	1230028104	2	240600100115	1	. . .
14909	120436	. . .	1230044374	1	240100200001	1	. . .
14862	120481	. . .	1230021668	1	240500200056	1	. . .
14853	120454	. . .	1230021653	1	220200200085	3	. . .
14838	121039	. . .	1230140184	1	220100300042	4	. . .

- The concatenation of the values for **Order_ID** and **Product_ID** forms a unique identifier for a row of data.

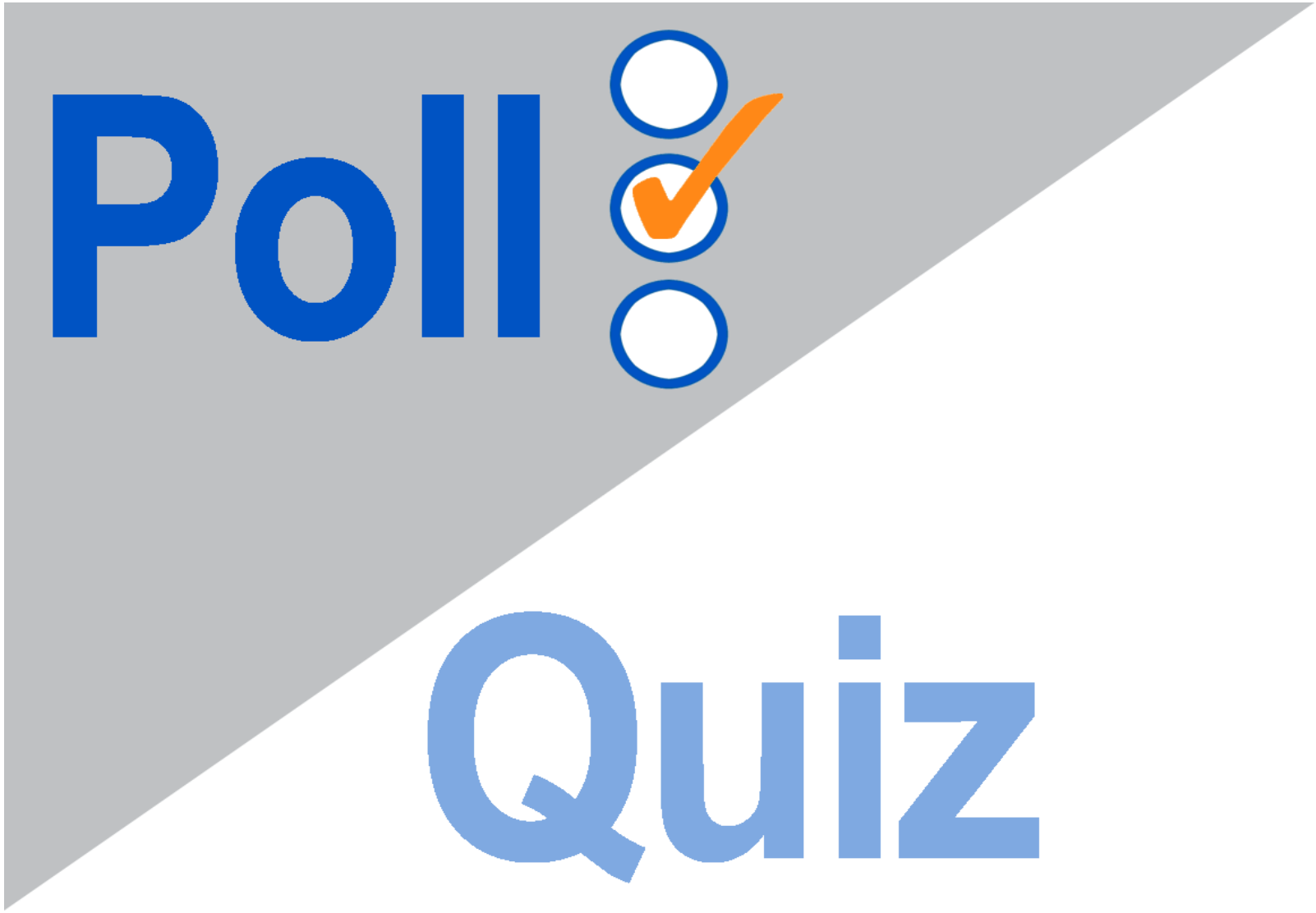
Index Terminology

- Index options include the following:

NOMISS	excludes all observations with missing values from the index. Observations with missing values can still be read from the data set, but not using the index.

  - If there is a large number of missing values for the key variable(s), the NOMISS option can create a smaller index file.
  - An index created with the NOMISS option is not used for the following processing:
    - a BY statement
    - a WHERE expression satisfied by missing values

NOMISS cannot be used when you create indexes using PROC SQL.


Poll
Quiz

3.03 Multiple Answer Poll

- On which of these indexed variables can you assign the UNIQUE option?

a. **Customer_ID** in an **orders** data set where a customer can place multiple orders
b. **Order_Date** in an **orders** data set
c. **Employee_ID** in a data set containing each individual employee and the family members' names stored in variables **Dependent1 – Dependent10**
d. **Product_ID** in a data set containing the product identifier and the product description

3.03 Multiple Answer Poll – Correct Answers

- On which of these indexed variables can you assign the UNIQUE option?

a. **Customer_ID** in an **orders** data set where a customer can place multiple orders

b. **Order_Date** in an **orders** data set

c. **Employee_ID** in a data set containing each individual employee and the family members' names stored in variables **Dependent1 – Dependent10**

d. **Product_ID** in a data set containing the product identifier and the product description

Creating Indexes

- To create indexes at the same time that you create a data set, use the INDEX= data set option on the output data set.
- To create or delete indexes on existing data sets, use one of the following:
  - DATASETS procedure
  - SQL procedure

Creating Indexes

- When you create the index, do the following:
  - designate the key variable(s)
  - specify the UNIQUE and/or the NOMISS index option if appropriate
  - select a valid SAS name for the index (composite index only)
- A data set can have these index features:
  - multiple simple and composite indexes
  - character and numeric key variables

Creating an Index with the INDEX= Data Set Option

```
options msglevel=i;
data orion.sales_history(index=
                (Customer_ID Product_Group
                 SaleID=(Order_ID Product_ID)/unique));
   set orion.history;
   Value_Cost=CostPrice_Per_Unit*Quantity;
   Year_Month=mdy(Month_Num, 15, input(Year_ID,4.));
   format Value_Cost dollar12.
          Year_Month monyy7.;
   label Value_Cost="Value Cost"
         Year_Month="Month/Year";
run;
```

- The following code would delete the indexes:

```
data  orion.sales_history;
   set orion.sales_history;
run;
```

p303d01

Creating an Index with the INDEX= Data Set Option

- General form of the INDEX= data set option:

SAS-data-file-name **(INDEX =**
(*index-specification-1</option> </option>*
*…<index-specification-n</option> </option> >***));**

-  For increased efficiency, use the INDEX= option to create indexes when you initially create a SAS data set.

Viewing Information about Indexes

- To display information in the log concerning index creation or index usage, change the value of the MSGLEVEL= system option from its default value of N to I.
- General form of the MSGLEVEL= system option:

OPTIONS MSGLEVEL=N | I;

```
11   options msglevel=i;
12   data orion.sales_history(index=
13         (Customer_ID Product_Group
14          SaleID=(Order_ID
15                  Product_ID)/unique));
16   set orion.sales_history;
17   run;

NOTE: There were 1500 observations read from the data set ORION.SALES_HISTORY.
NOTE: The data set ORION.SALES_HISTORY has 1500 observations and 22 variables.
NOTE: Composite index SaleID has been defined.
NOTE: Simple index Product_Group has been defined.
NOTE: Simple index Customer_ID has been defined.
```

Creating an Index with the INDEX= Data Set Option

Advantages	Disadvantages
You can create the data set and the index in one step.	To create an additional index, you must re-create the existing indexes.
SAS only reads the data once.	You need to know in advance that indexes are needed.

Managing Indexes with PROC DATASETS

```
options msglevel=n;
proc datasets library=orion nolist;
   modify sales_history;
      index create Customer_ID;
      index create Product_Group;
      index create SaleID=(Order_ID
                   Product_ID)/unique;
quit;
```

- The following code would delete the indexes:

```
proc datasets library=orion nolist;
   modify sales_history;
      index delete Customer_ID
                   Product_Group SaleID;
quit;
```

p303d02

Managing Indexes with PROC DATASETS

- You can use the DATASETS procedure on existing data sets to create or delete indexes.
- General form of the PROC DATASETS step to delete or create indexes:

```
PROC DATASETS LIBRARY=libref NOLIST;
      MODIFY SAS-data-set-name;
            INDEX DELETE index-name;
            INDEX CREATE index-specification
                                    < / options>;
QUIT;
```

Poll

Quiz

3.04 Quiz

- Open and submit the program **p303a01**.
- What error messages are in the log?

```
options msglevel=n;
proc datasets library=orion nolist;
   modify sales_history;
       index create Customer_ID;
       index create Product_Group;
       index create SaleID=(Order_ID
                            Product_ID)/unique;
quit;
```

p303a01

3.04 Quiz – Correct Answer

```
1     options msglevel=n;
2     proc datasets library=orion nolist;
3        modify sales_history;
4           index create Customer_ID;
ERROR: An index named Customer_ID with the same definition already exists for
       file ORION.SALES_HISTORY.DATA.
5           index create Product_Group;
6           index create SaleID=(Order_ID
7                        Product_ID)/unique;
8     quit;

NOTE: Statements not processed because of errors noted above.
NOTE: The SAS System stopped processing this step because of errors.
NOTE: PROCEDURE DATASETS used (Total process time):
      real time           0.48 seconds
      cpu time            0.09 seconds
```

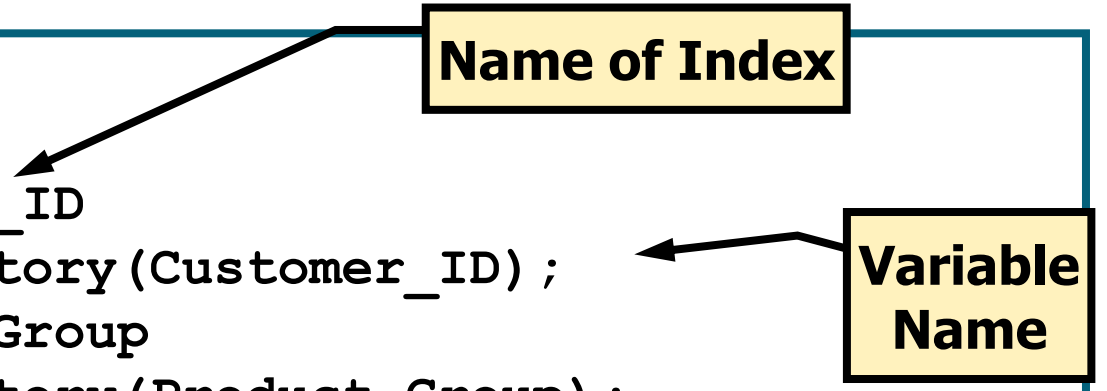
If an index exists, it must be deleted before it can be re-created.

Managing Indexes with PROC DATASETS

Advantages	Disadvantages
Additional indexes can be created without re-creating the original indexes.	You can only create indexes on existing SAS data sets and existing variables.
One or more indexes can be deleted without deleting all of the indexes on the data set.	PROC DATASETS cannot perform data manipulation.
	If an index exists, it must be deleted before it can be re-created.

Managing Indexes with PROC SQL

```
options msglevel=n;
proc sql;
   create index Customer_ID
      on orion.sales_history(Customer_ID);
   create index Product_Group
      on orion.sales_history(Product_Group);
   create unique index SaleID
      on orion.sales_history(Order_ID, Product_ID);
quit;
```



- The following code would delete the indexes:

```
proc sql;
   drop index Customer_ID, Product_Group, SaleID
      from orion.sales_history;
quit;
```

Managing Indexes with PROC SQL

- You can use PROC SQL on existing data sets to create or delete indexes.
- General form of the PROC SQL step to create or delete indexes:

```
PROC SQL;
        DROP INDEX index-name
              FROM table-name;
       CREATE <option> INDEX  index-name
              ON table-name(column-name-1,...
                                column-name-n);
QUIT;
```

Managing Indexes with PROC SQL

Advantages	Disadvantages
Additional indexes can be created without re-creating the original indexes.	You can only create indexes on existing SAS data sets and existing variables.
One or more indexes can be deleted without deleting all of the indexes on the data set.	The CREATE INDEX statement cannot perform data manipulation.
	If an index exists, it must be deleted before it can be re-created.

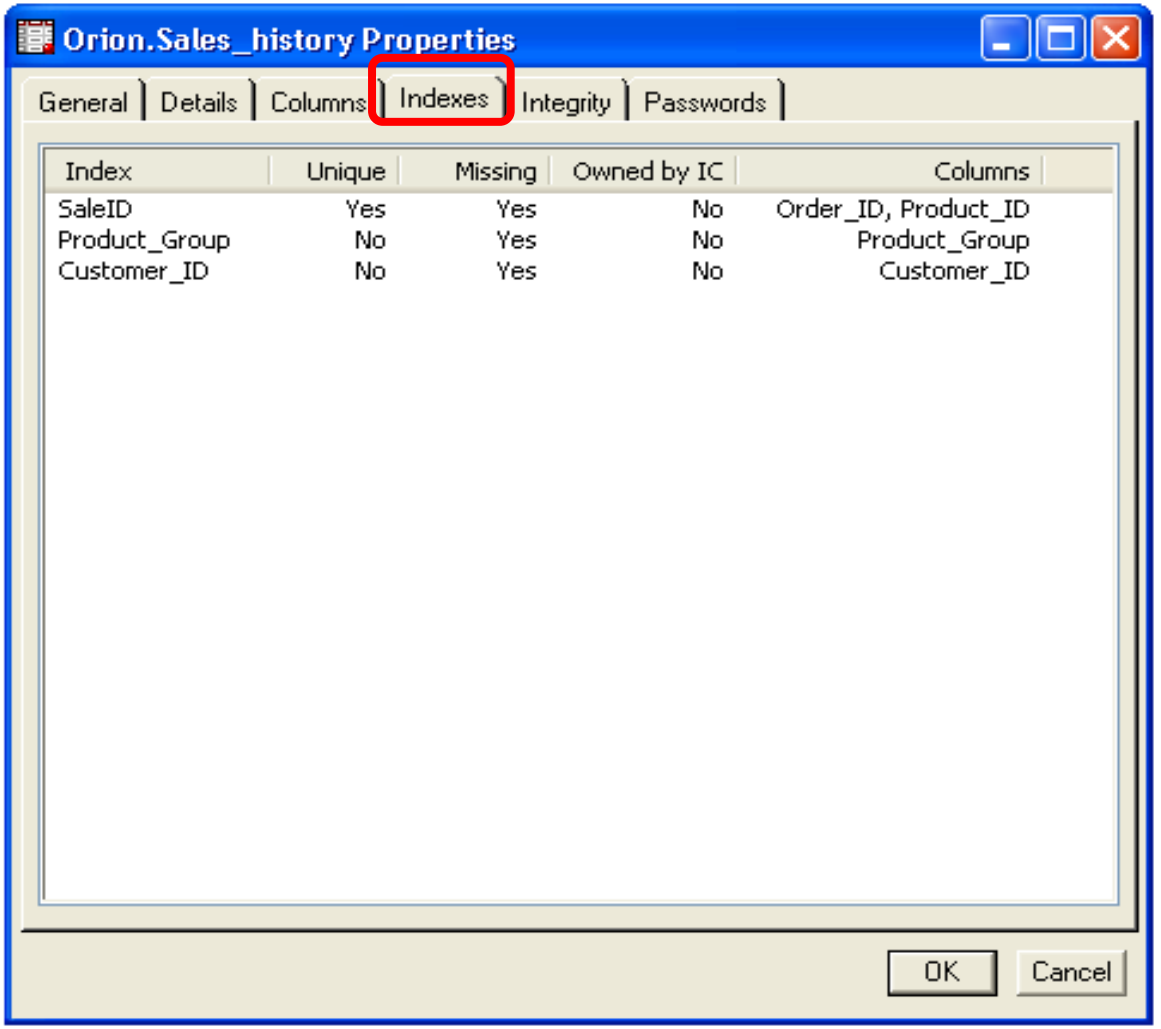
Comparing Techniques for Index Creation

INDEX= Data Set Option	PROC DATASETS	PROC SQL
You can create the SAS data set at the same time that the index is created.	You can only create indexes on existing SAS data sets and existing variables.	You can only create indexes on existing SAS data sets and existing variables.
To create an additional index, you must re-create the existing indexes.	Additional indexes can be created without re-creating the original indexes.	Additional indexes can be created without re-creating the original indexes.
The DATA step can perform data manipulation at the same time that the index is created.	PROC DATASETS cannot perform data manipulation.	The CREATE INDEX statement cannot perform data manipulation.
To delete one or more indexes, you must re-create the other required indexes.	One or more indexes can be deleted without deleting all of the indexes on the data set.	One or more indexes can be deleted without deleting all of the indexes on the data set.
An existing index can be re-created without first deleting it.	If an index exists, it must be deleted before it can be re-created.	If an index exists, it must be deleted before it can be re-created.

Documenting Indexes

- The following can be used to document indexes:
  - SAS Explorer
  - PROC CONTENTS
  - PROC DATASETS
  - SAS Management Console

Properties Window in SAS Explorer



Index Documentation

```
proc contents data=orion.sales_history;
run;
```

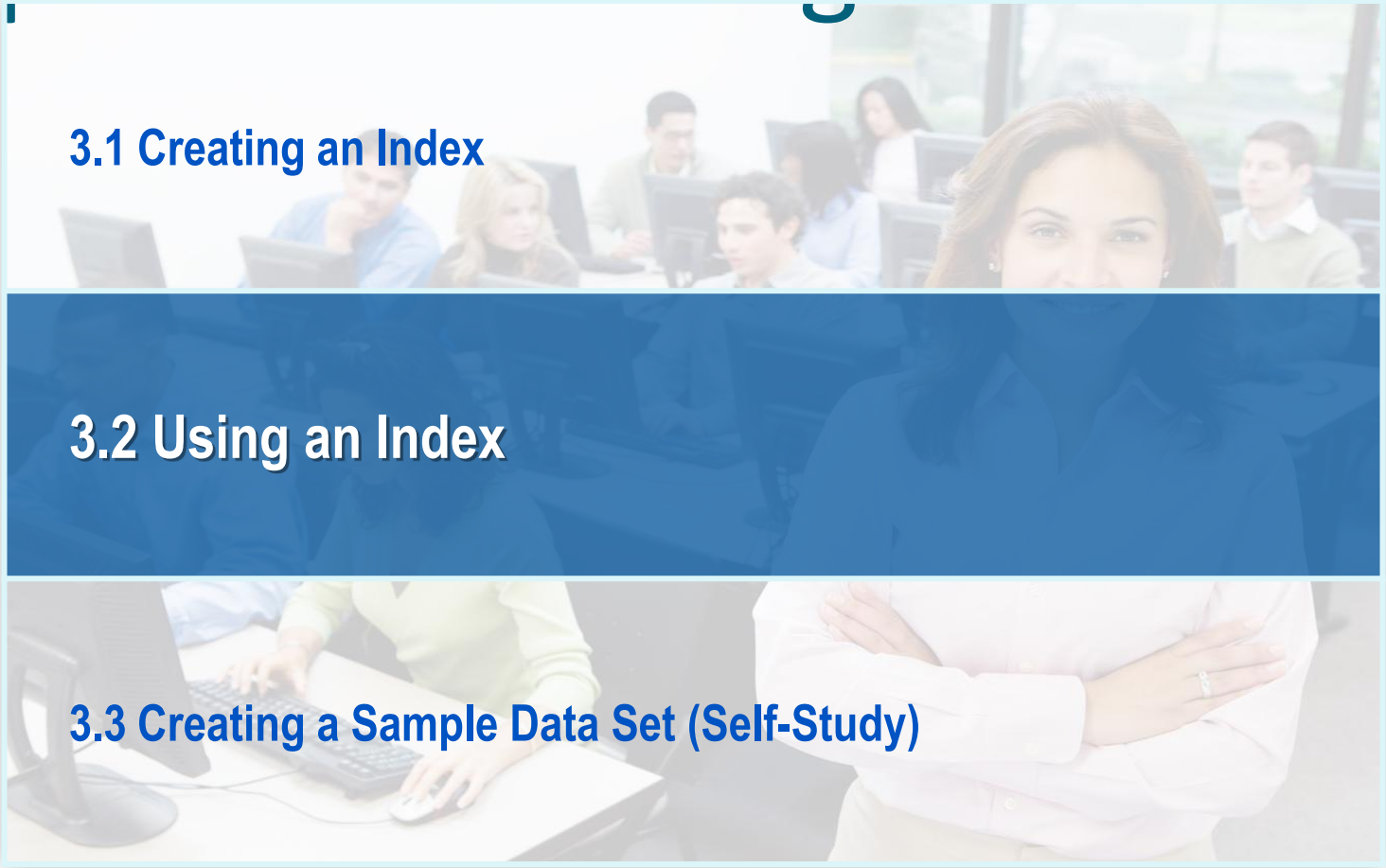
```
proc datasets lib=orion nolist;
   contents data=sales_history;
quit;
```

These two steps produce identical output.

Partial PROC DATASETS Output

```
            Alphabetic List of Indexes and Attributes

                                    # of
                          Unique    Unique
#     Index               Option    Values    Variables

1     Customer_ID                     1046
2     Product_Group                     56
3     SaleID              YES         1500    Order_ID Product_ID
```

Chapter 3: Accessing Observations



3.1 Creating an Index

3.2 Using an Index

3.3 Creating a Sample Data Set (Self-Study)

Objectives

- Describe when an index is used for WHERE statement processing.
- Describe when an index is ***not*** used for WHERE statement processing.

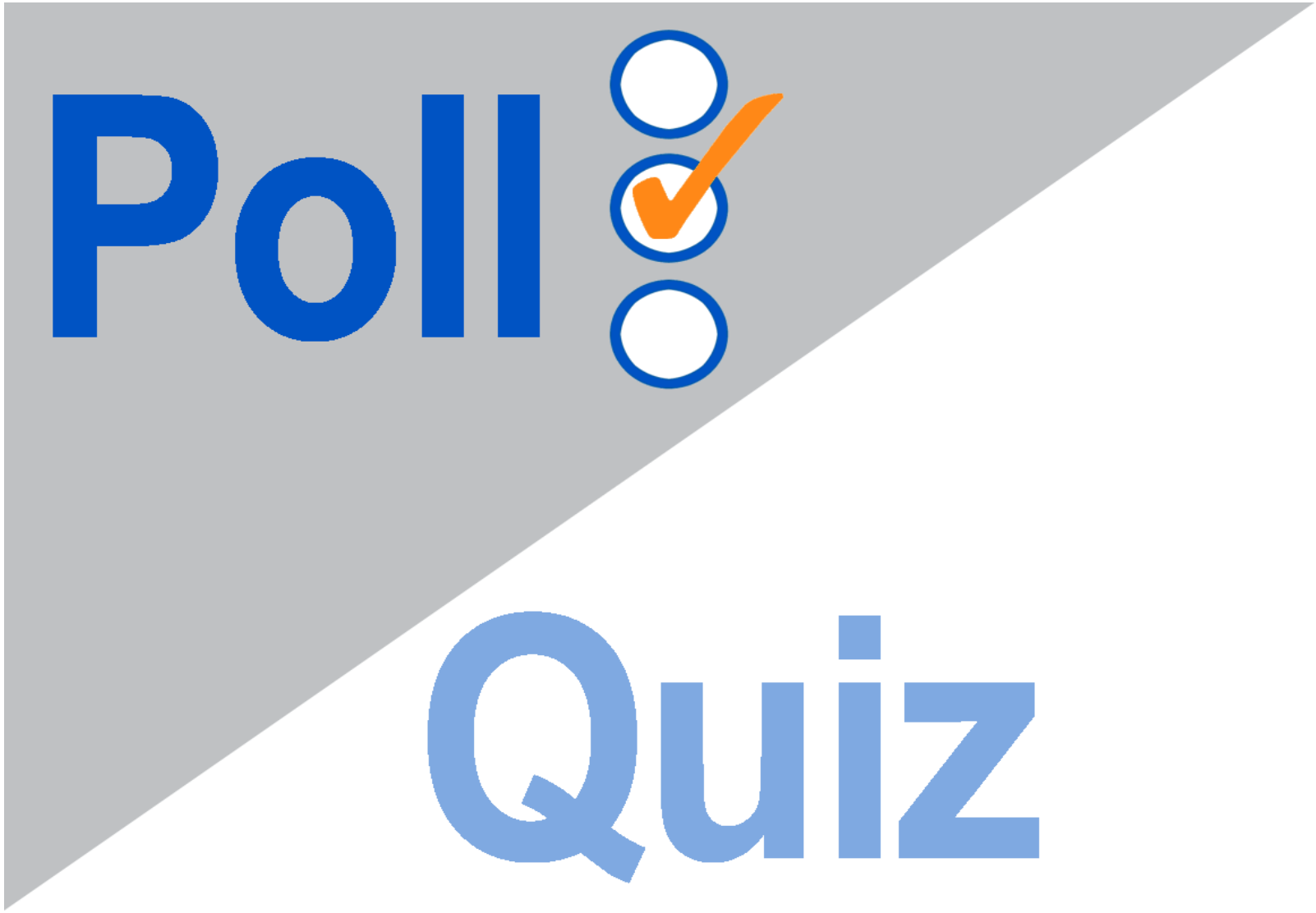
Index Usage Possible

- An index ***might*** be used when a WHERE expression references one of the following:
  - a simple index key variable
  - the primary key variable of a composite index

Although a WHERE expression can consist of multiple conditions that specify different variables, SAS uses only one index to process the WHERE expression.

Index Usage Possible

- A WHERE condition might possibly use an index, provided the condition contains any one of the following:
  - a comparison operator or the IN operator
  - the NOT operator
  - the special WHERE operators (CONTAINS, LIKE, IS NULL|IS MISSING, and BETWEEN…AND)
  - the TRIM or SUBSTR functions (***if*** the second argument of the SUBSTR function is 1)


Poll
Quiz

Setup for the Poll

The following indexes were created on the **orion.sales_history** data set.

Partial PROC DATASETS Output

Alphabetic List of Indexes and Attributes

#	Index	Unique Option	# of Unique Values	Variables
1	Customer_ID		1046	
2	Product_Group		56	
3	SaleID	YES	1500	Order_ID Product_ID

3.05 Multiple Answer Poll

- Which of the following WHERE conditions could possibly use an index?

a. `where Product_ID=220100300042;`

b. `where Customer_ID ne 3245;`

c. `where Customer_ID=15020 or Customer_ID=14853;`

d. `where Order_ID=1230036183;`

e. `where Customer_ID='3245';`

3.05 Multiple Answer Poll – Correct Answers

- Which of the following WHERE conditions could possibly use an index?

a. `where Product_ID=220100300042;`

b. `where Customer_ID ne 3245;`

c. `where Customer_ID=15020 or Customer_ID=14853;`

d. `where Order_ID=1230036183;`

e. `where Customer_ID='3245';`

When Is an Index Not Used?

- An index is ***not*** used in the following circumstances:
  - with a subsetting IF statement in a DATA step
  - with particular WHERE expressions
  - if SAS determines that all observations will satisfy the WHERE expression
  - if SAS determines that it is more efficient to read the data sequentially

No Index Usage

- SAS does not use an index when a WHERE expression references an indexed variable if the following conditions exist:
  - No single index can supply all required observations.
  - Any function other than TRIM or SUBSTR appears in the WHERE expression.
  - The SUBSTR function does not search a string beginning at the first position.
  - The SOUNDS-LIKE operator (=*) is used.

Compound Optimization

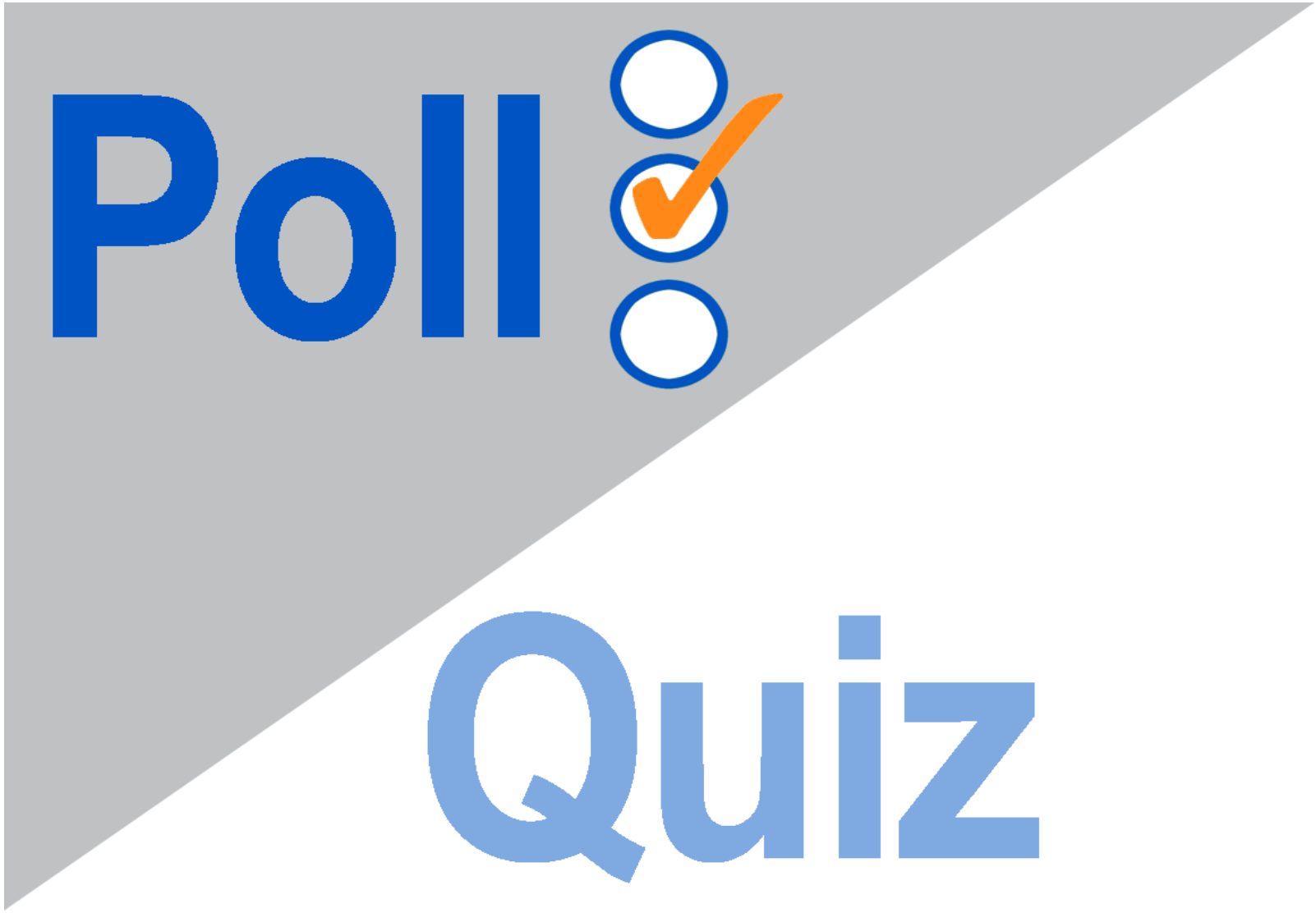
- A WHERE expression that references multiple variables can take advantage of a composite index.

compound optimization	use of a composite index to optimize some WHERE expressions that involve multiple variables

```
where Order_ID=240200100038 and
        Product_ID=1230151326;
```

Compound Optimization

- For compound optimization to occur, ***all*** of the following must be true:
  - At least the first two key variables in the composite index must be used in the WHERE conditions.
  - The conditions must be connected using the AND operator.
  - At least one condition must use the EQ, equal sign (=), or IN operator.


Poll
Quiz

3.07 Multiple Choice Poll

- Which of the following WHERE statements can use the composite index **SaleID** for compound optimization?

a. `where Order_ID=240200100038 or Product_ID=1230151326;`

b. `where Order_ID=. and Product_ID=1230151326;`

c. `where int(Order_ID/1000000000)=240 and Product_ID=1230151326;`

d. `where Order_ID>240000000000 and Product_ID<1240000000;`

3.07 Multiple Choice Poll – Correct Answer

- Which of the following WHERE statements can use the composite index **SaleID** for compound optimization?

a. `where Order_ID=240200100038 or Product_ID=1230151326;`

b. `where Order_ID=. and Product_ID=1230151326;`

c. `where int(Order_ID/1000000000)=240 and Product_ID=1230151326;`

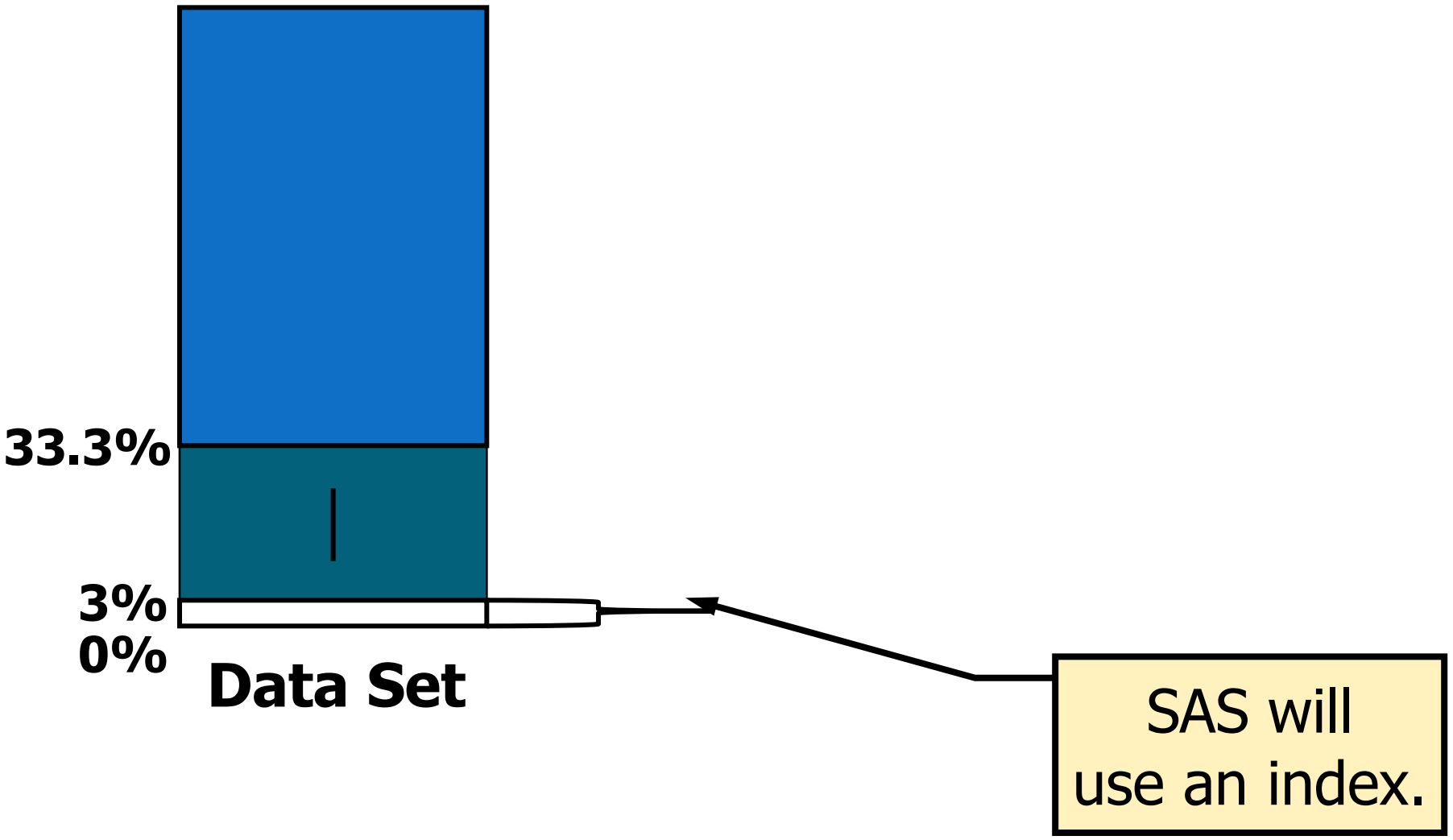
d. `where Order_ID>240000000000 and Product_ID<1240000000;`

WHERE Expression Index Usage

- SAS uses the following steps to decide whether to evaluate a WHERE expression using a sequential read or using an index:
  - Determine whether the WHERE expression can be satisfied by an existing index.
  - Select the best index, if several indexes are available.
  - Estimate the number of observations that qualify.
  - Compare the probable resource usage for both methods.

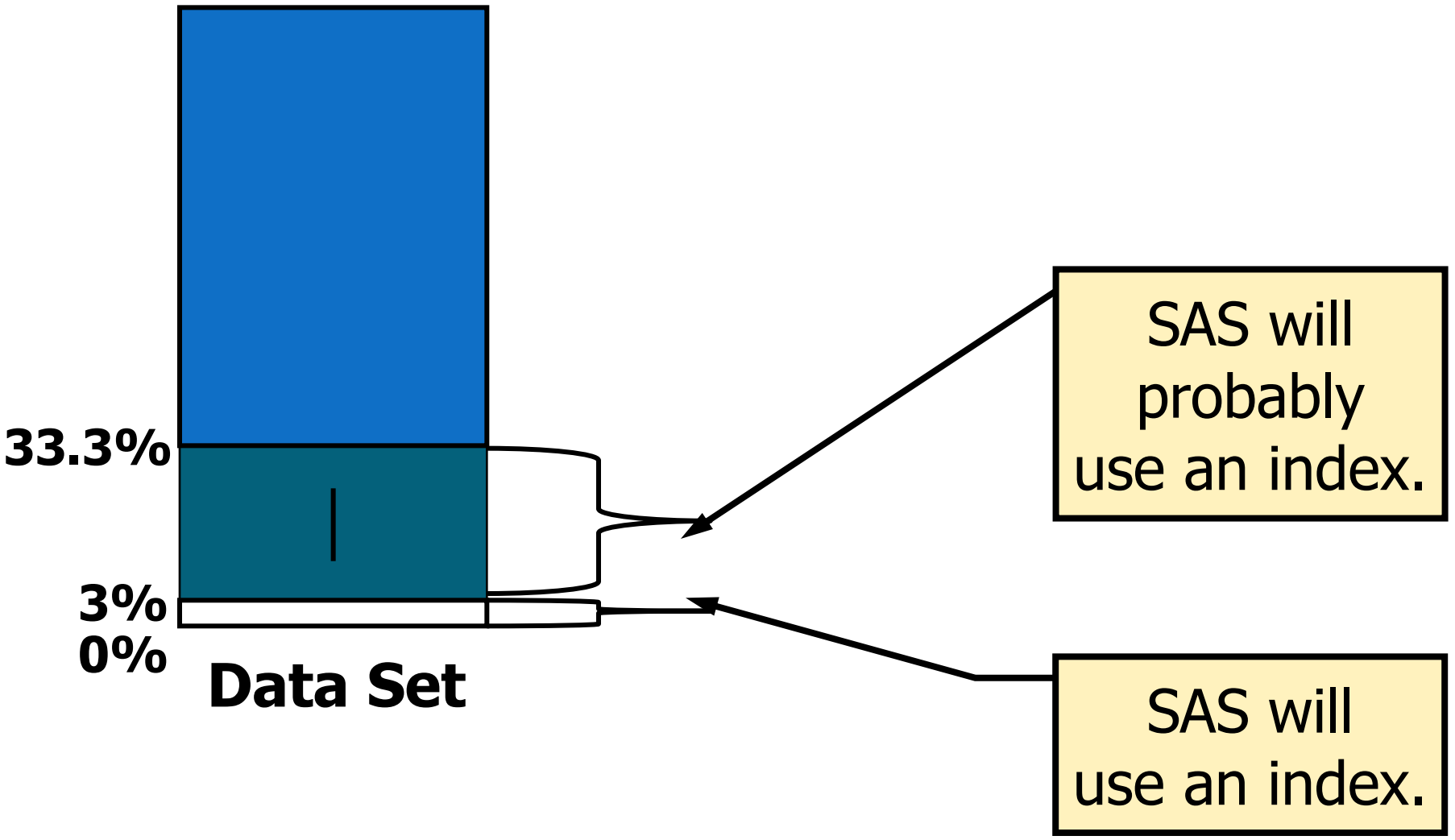
SAS estimates the I/O operations for indexed access based on the subset size and sort order.

Subset Size



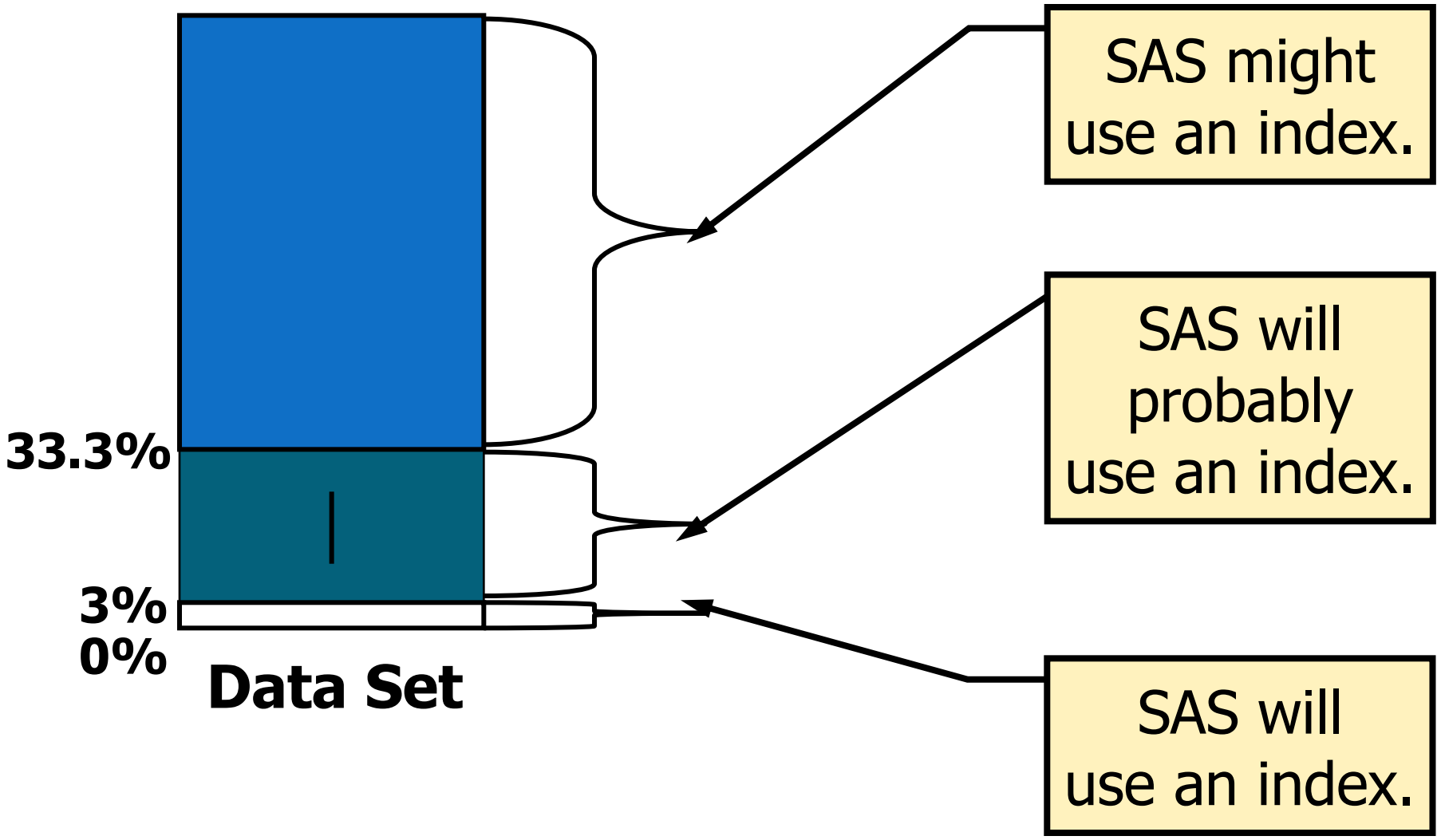
...

Subset Size


33.3%
3%
0%
Data Set
SAS will probably use an index.
SAS will use an index.

...

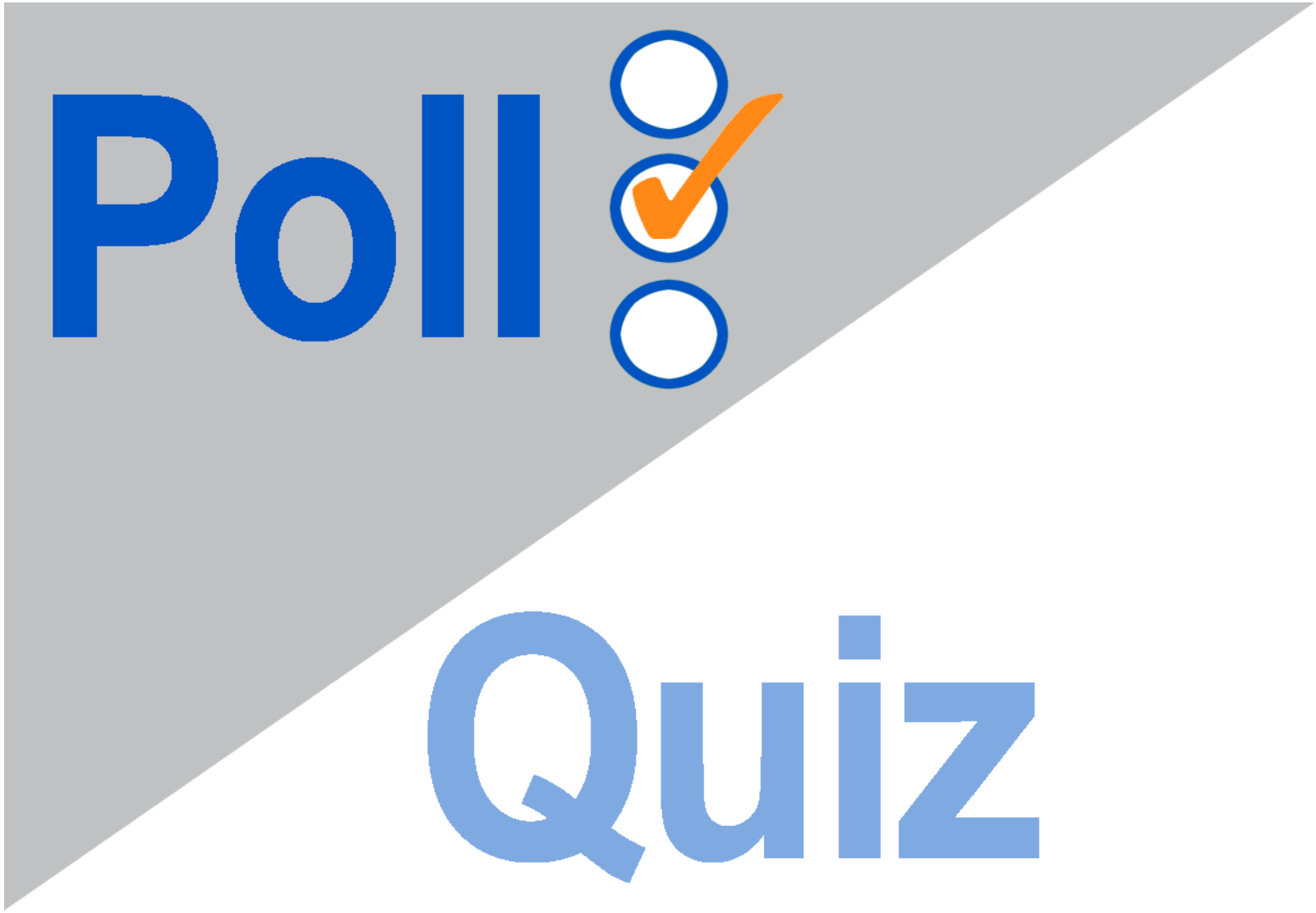
Subset Size


33.3%
3%
0%
Data Set
SAS might use an index.
SAS will probably use an index.
SAS will use an index.

Subset Size

- The SAS index includes cumulative percentiles or centiles. By default, SAS stores 21 centiles or every

centiles	provide information about the distribution of values in an index.


Poll
Quiz

3.08 Multiple Choice Poll

- Which of the following is used to determine the I/O to read a SAS data set sequentially?

a. the page size of the input data set and the number of buffers available
b. the number of observations and the number of variables
c. the page size of the output data set and the number of output buffers available

3.08 Multiple Choice Poll – Correct Answer

- Which of the following is used to determine the I/O to read a SAS data set sequentially?

a. the page size of the input data set and the number of buffers available

b. the number of observations and the number of variables

c. the page size of the output data set and the number of output buffers available

Review of Factors Affecting I/O

- The following factors affect I/O:
  - size of the subset relative to the size of the data file
  - order of data with regard to the chosen index
  - page size of the data file
  - number of buffers allocated
  - cost to uncompress a compressed file for a sequential read

Data Order

Obs	Customer_ID
.	
.	
.	
8939	56487
8940	70175
8941	74667
.	
.	
.	
.	
.	
.	
32548	89619
32549	70187
32550	76278
.	
.	
.	
.	
.	
.	
45775	84989
45776	70201
45777	20209
.	
.	
.	

Unsorted data

For data that is sorted and indexed on the same variable(s), retrieval time through the index is much faster than either sorted or indexed data alone.

```
where Customer_ID in
 (70201, 70187, 70175);
```

Fewer pages are copied into memory if the data is sorted.

Obs	Customer_ID
.	
.	
.	
51050	70175
51051	70177
51052	70180
51053	70181
51054	70183
51055	70184
51056	70186
51057	70187
51058	70188
51059	70191
51060	70192
51061	70193
51062	70194
51063	70195
51064	70197
51065	70199
51066	70200
51067	70201
.	
.	
.	

Sorted data

Controlling WHERE Processing Index Usage

- You can control index usage for WHERE processing with these data set options:

IDXWHERE=YES	tells SAS to choose the best index to optimize a WHERE expression and to disregard the possibility that a sequential search of the data set might be more resource efficient.
IDXWHERE=NO	tells SAS to ignore all indexes and satisfy the conditions of a WHERE expression with a sequential search of the data set.
IDXNAME=*index-name*	directs SAS to use a specific index.

Use the IDXWHERE=NO option when you know an available index will not optimize WHERE clause processing.

Using the IDXWHERE= Option

- To ensure that SAS uses an index when printing the data for **Customer_ID in (14844,4983,5862,10032) and Product_Group contains 'Shoes'**, use the following code:

```
options msglevel=i;
proc print data=orion.sales_history(idxwhere=yes);
   where Customer_ID in (14844,4983,5862,10032)
         and Product_Group contains 'Shoes';
   var Customer_ID Product_ID Product_Group ;
   title 'With an Index';
run;
```

Using the IDXWHERE= Option

Partial SAS Log

```
1669  options msglevel=i;
1670  proc print data=orion.sales_history(idxwhere=yes);
1671     where Customer_ID in (14844,4983,5862,10032)
1672           and Product_Group contains 'Shoes';
INFO: Data set option (IDXWHERE=YES)forced an index to be used rather
      than a sequential pass for where-clause processing.
INFO: Index Customer_ID selected for WHERE clause optimization.
1673     var Customer_ID Product_ID Product_Group ;
1674     title 'With an Index';
1675  run;
```

Using the IDXNAME= Option

- Because using the index on **Customer_ID** returns a smaller subset than would the index on **Product_Group**, the IDXNAME= data set option can be used.

```
options msglevel=i;
proc print data=orion.sales_history(idxname=Customer_ID) ;
   where Customer_ID in (14844,4983,5862,10032)
         and Product_Group contains 'Shoes';
   var Customer_ID Product_ID Product_Group ;
   title 'With an Index';
run;
```

Use the IDXNAME= option when you know the better index so SAS does not need to do the evaluation.

Using the IDXNAME= Option

Partial SAS Log

```
92  options msglevel=i;
193  proc print data=orion.sales_history(idxname=Customer_ID);
194     where Customer_ID in (14844,4983,5862,10032)
195           and Product_Group contains 'Shoes';
INFO: Index Customer_ID selected for WHERE clause optimization.
196     var Customer_ID Product_ID Product_Group ;
197     title 'With an Index';
198  run;

NOTE: There were 3 observations read from the data set
ORION.SALES_HISTORY.
      WHERE Customer_ID in (4983, 5862, 10032, 14844) and
            Product_Group contains 'Shoes';
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.15 seconds
      cpu time            0.01 seconds
```

Maintaining Indexes

Data Management Tasks	Index Action Taken
Copy the data set with the COPY procedure or the DATASETS procedure	Index file constructed for new data file
Move the data set with the MOVE option in the COPY procedure	Index file deleted from IN= library; rebuilt in OUT= library
Copy the data set with a drag-and-drop action in SAS Explorer	Index file constructed for new file

continued...

Maintaining Indexes

Data Management Tasks	Index Action Taken
Rename the data set	Index file renamed
Rename the variable	Variable renamed to new name in index file
Add observations	Value/Identifier pairs added
Delete observations	Value/Identifier pairs deleted; space recovered for re-use
Update observations	Value/Identifier pairs updated if values change

- The APPEND procedure and the INSERT INTO statement in the SQL procedure update the index file after all the data is appended or inserted.

continued...

Maintaining Indexes

Data Management Tasks	Index Action Taken

Delete a data set.
`proc datasets lib=work;`
`  delete a;`
`run;`
	Index file deleted

Rebuild a data set with a DATA step or the SQL procedure.
`data a;`
`  set a;`
`run;`
`proc sql;`
`  create table a as`
`  select * from a;`
`quit;`
	Index file deleted

Sort the data set in place with the FORCE option in the SORT procedure.
`proc sort data=a force;`
`  by var;`
`run;`
	Index file deleted

Guidelines for Indexing

- Suggested guidelines for creating indexes:
  - Create an index when you intend to retrieve a small subset of observations from a large data file.
  - Do not create an index if the data file page count is less than three pages. It is faster to access the data sequentially.
  - Create indexes on variables that are discriminating. These variables precisely identify observations that satisfy WHERE expressions.
  - When you create a composite index, make the first key variable the most discriminating.
  - Consider the cost of maintaining an index for a data file that is frequently changed.

continued...

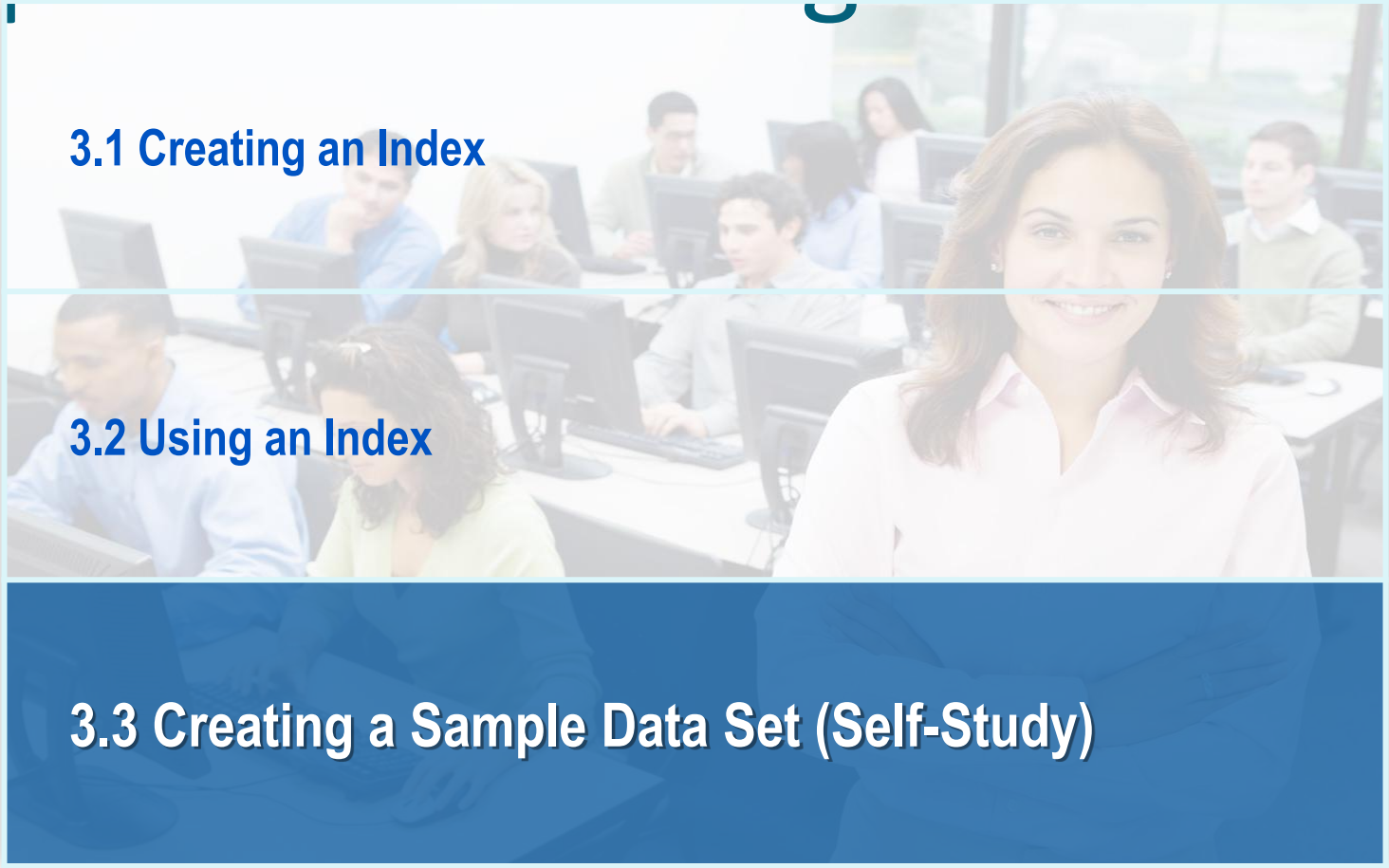
Guidelines for Indexing

- To minimize I/O for indexed access, sort the data by the key variable(s) before creating the index. Maintain the data file in sorted order by the key variable to improve performance.
- Minimize the number of indexes to reduce disk storage and update costs. Create indexes only on variables that are often used in queries or BY-group processing (when the data cannot be sorted).
- Consider how often your applications use an index.
  An index must be used often in order to compensate for the resources used in creating and maintaining it.
- When you create an index to process a WHERE expression, do not try to create one index that might be used to satisfy every conceivable query.

Index Trade-offs

Advantages	Disadvantages
fast access to a small subset of observations	extra CPU cycles and I/O operations to create and maintain an index
values returned in sorted order	increased CPU to read the data
can enforce uniqueness	extra disk space to store the index file
	extra memory to load the index pages and the compiled SAS C code to use the index

Chapter 3: Accessing Observations



3.1 Creating an Index

3.2 Using an Index

3.3 Creating a Sample Data Set (Self-Study)

Objectives

- Create a systematic sample.
- Create a random sample with replacement.
- Create a random sample without replacement.

Business Scenario

- The Marketing Department wants to send customer satisfaction questionnaires to a sample of the customers in the **orion.order_fact** SAS data set.

Partial Listing of **orion.order_fact**

Customer_ID	Employee_ID	Street_ID	Order_Date	Delivery_Date	Order_ID	. . .
63	121039	9260125492	11JAN2003	11JAN2003	1230058123	...
5	99999999	9260114570	15JAN2003	19JAN2003	1230080101	...
45	99999999	9260104847	20JAN2003	22JAN2003	1230106883	...
41	120174	1600101527	28JAN2003	28JAN2003	1230147441	...
183	120134	1600100760	27FEB2003	27FEB2003	1230315085	...
.	.	.	.	.	.	
.	.	.	.	.	.	
.	.	.	.	.	.	

Business Scenario

- Select a subset by reading every 50th observation from observation number 1 to the end of the SAS data set.

```
data subset;
 ③ do PickIt=1 to TotObs by 50; ②
       set
orion.order_fact(keep=Customer_ID
                 Employee_ID Street_ID
Order_ID)
            point=PickIt
            nobs=TotObs; ①
       output; ④
   end;
   stop; ⑤
run;
```

Poll

Quiz

3.09 Quiz

- Are POINT= and NOBS= individual statements or part of the SET statement?

```
data subset;
   do PickIt=1 to TotObs by 50;
      set orion.order_fact(keep=Customer_ID
                Employee_ID Street_ID Order_ID)
          point=PickIt
          nobs=TotObs;
        output;
   end;
   stop;
run;
```

3.09 Quiz – Correct Answer

```
data subset;
   do PickIt=1 to TotObs by 50;
      set orion.order_fact(keep=Customer_ID
             Employee_ID Street_ID Order_ID)
          point=PickIt
          nobs=TotObs;
        output;
   end;
   stop;
run;
```

POINT= and NOBS= are part of the SET statement.

Using the POINT= Option

- To create a sample, use the POINT= option in the SET statement.
- General form of the POINT= option:

SET *data-set-name* POINT=*point-variable*;

- The *point-variable* has the following attributes:
  - names a temporary numeric variable that contains the number of the observation to read
  - must be given a value before the execution of the SET statement
  - must be a variable (for example, X) and not a constant value (for example, 12)
  - must be a valid observation number

Using the Number of Observations

- You can use the NOBS= option in the SET statement to determine how many observations there are in a SAS data set.
- General form of the SET statement:

SET *SAS-data-set* NOBS= *variable*;

- The NOBS= option creates a temporary variable whose value has the following characteristics:
  - is the number of observations in the input data set(s)
  - is assigned during compilation
  - is retained
  - should not be modified during execution

Using the STOP Statement

- The POINT= option has the following features:
  - uses direct-access read mode
  - does not detect the end-of-file marker
- To prevent the DATA step from looping continuously, use the STOP statement.
- General form of the STOP statement:

```
STOP;
```

Compilation

```
data subset;
   do PickIt=1 to TotObs by 50;
      set orion.order_fact
          (keep=Customer_ID
                Employee_ID
                Street_ID
                Order_ID)
         point=PickIt
         nobs=TotObs;
        output;
   end;
   stop;
run;
```

PDV

D PickIt	D Tot Obs	Customer_ID	Employee_ID	Street_ ID	Order_ID	D _N_
.	617	.	.	.	.	.

p303d07

Execution

Partial Listing of
orion.order_fact

obs	Customer_ID	Employee_ID	. . .
1	63	121039	. . .
2	5	99999999	. . .
	. . .	. . .	. . .
50	17023	99999999	. . .
51	17023	99999999	. . .
	. . .	. . .	. . .

```
data subset;
   do PickIt=1 to TotObs by 50;
      set orion.order_fact
          (keep=Customer_ID
                Employee_ID
                Street_ID
                Order_ID)
          point=PickIt
          nobs=TotObs;
       output;
   end;
   stop;
run;
```

PDV

D PickIt	D Tot Obs	Customer_ID	Employee_ID	Street_ ID	Order_ID	D _N_
1	617	.	.	.	.	1

...

Execution

Partial Listing of
orion.order_fact

obs	Customer_ID	Employee_ID	. . .
1	63	121039	. . .
2	5	99999999	. . .
	. . .	. . .	. . .
50	17023	99999999	. . .
51	17023	99999999	. . .
	. . .	. . .	. . .

```
data subset;
   do PickIt=1 to TotObs by 50;
      set orion.order_fact
          (keep=Customer_ID
                Employee_ID
                Street_ID
                Order_ID)
         point=PickIt
         nobs=TotObs;
        output;
   end;
   stop;
run;
```

PDV

D PickIt	D Tot Obs	Customer_ID	Employee_ID	Street_ ID	Order_ID	D _N_
1	617	63	121039	9260125492	1230058123	1

...

Execution

Partial Listing of
orion.order_fact

obs	Customer_ID	Employee_ID	. . .
1	63	121039	. . .
2	5	99999999	. . .
	. . .	. . .	. . .
50	17023	99999999	. . .
51	17023	99999999	. . .
	. . .	. . .	. . .

```
data subset;
   do PickIt=1 to TotObs by 50;
      set orion.order_fact
          (keep=Customer_ID
                Employee_ID
                Street_ID
                Order_ID)
          point=PickIt
          nobs=TotObs;
       output;
   end;
   stop;
run;
```

Output current observation.

PDV

D PickIt	D Tot Obs	Customer_ID	Employee_ID	Street_ ID	Order_ID	D _N_
1	617	63	121039	9260125492	1230058123	1

...

Execution

Partial Listing of
orion.order_fact

obs	Customer_ID	Employee_ID	. . .
1	63	121039	. . .
2	5	99999999	. . .
	. . .	. . .	. . .
50	17023	99999999	. . .
51	17023	99999999	. . .
	. . .	. . .	. . .

```
data subset;
   do PickIt=1 to TotObs by 50;
      set orion.order_fact
          (keep=Customer_ID
                Employee_ID
                Street_ID
                Order_ID)
          point=PickIt
          nobs=TotObs;
       output;
   end;
   stop;
run;
```

PDV

D PickIt	D Tot Obs	Customer_ID	Employee_ID	Street_ ID	Order_ID	D _N_
51	617	63	121039	9260125492	1230058123	1

...

Execution

Partial Listing of
orion.order_fact

obs	Customer_ID	Employee_ID	. . .
1	63	121039	. . .
2	5	99999999	. . .
	. . .	. . .	. . .
50	17023	99999999	. . .
51	17023	99999999	. . .
	. . .	. . .	. . .

```
data subset;
   do PickIt=1 to TotObs by 50;
      set orion.order_fact
          (keep=Customer_ID
                Employee_ID
                Street_ID
                Order_ID)
          point=PickIt
          nobs=TotObs;
       output;
   end;
   stop;
run;
```

PDV

D PickIt	D Tot Obs	Customer_ID	Employee_ID	Street_ ID	Order_ID	D _N_
51	617	63	121039	9260125492	1230058123	1

...

Execution

Partial Listing of
orion.order_fact

obs	Customer_ID	Employee_ID	. . .
1	63	121039	. . .
2	5	99999999	. . .
	. . .	. . .	. . .
50	17023	99999999	. . .
51	17023	99999999	. . .
	. . .	. . .	. . .

```
data subset;
   do PickIt=1 to TotObs by 50;
      set orion.order_fact
          (keep=Customer_ID
                Employee_ID
                Street_ID
                Order_ID)
         point=PickIt
         nobs=TotObs;
        output;
   end;
   stop;
run;
```

PDV

D PickIt	D Tot Obs	Customer_ID	Employee_ID	Street_ ID	Order_ID	D _N_
51	617	17023	99999999	2600100021	1230931366	1

...

Execution

Partial Listing of **orion.order_fact**

obs	Customer_ID	Employee_ID	. . .
1	63	121039	. . .
2	5	99999999	. . .
	. . .	. . .	. . .
50	17023	99999999	. . .
51	17023	99999999	. . .
	. . .	. . .	. . .

```
data subset;
   do PickIt=1 to TotObs by 50;
      set orion.order_fact
          (keep=Customer_ID
                Employee_ID
                Street_ID
                Order_ID)
          point=PickIt
          nobs=TotObs;
       output;
   end;
   stop;
run;
```

Output current observation.

PDV

D PickIt	D Tot Obs	Customer_ID	Employee_ID	Street_ ID	Order_ID	D _N_
51	617	17023	99999999	2600100021	1230931366	1

...

Execution

Partial Listing of
orion.order_fact

obs	Customer_ID	Employee_ID	. . .
1	63	121039	. . .
2	5	99999999	. . .
	. . .	. . .	. . .
50	17023	99999999	. . .
51	17023	99999999	. . .
	. . .	. . .	. . .

```
data subset;
   do PickIt=1 to TotObs by 50;
      set orion.order_fact
          (keep=Customer_ID
                Employee_ID
                Street_ID
                Order_ID)
          point=PickIt
          nobs=TotObs;
       output;
   end;
   stop;
run;
```

PDV

D PickIt	D Tot Obs	Customer_ID	Employee_ID	Street_ ID	Order_ID	D _N_
101	617	17023	99999999	2600100021	1230931366	1

...

Execution

Partial Listing of
orion.order_fact

obs	Customer_ID	Employee_ID	. . .
1	63	121039	. . .
2	5	99999999	. . .
	. . .	. . .	. . .
50	17023	99999999	. . .
51	17023	99999999	. . .
	. . .	. . .	. . .

PickIt > TotObs

```
data subset;
   do PickIt=1 to TotObs by 50;
      set orion.order_fact
          (keep=Customer_ID
                Employee_ID
                Street_ID
                Order_ID)
          point=PickIt
          nobs=TotObs;
       output;
   end;
   stop;
run;
```

PDV

D PickIt	D Tot Obs	Customer_ID	Employee_ID	Street_ ID	Order_ID	D _N_
651	617	215	120175	1600102721	1243963366	1

...

Execution

Partial Listing of
orion.order_fact

obs	Customer_ID	Employee_ID	. . .
1	63	121039	. . .
2	5	99999999	. . .
	. . .	. . .	. . .
50	17023	99999999	. . .
51	17023	99999999	. . .
	. . .	. . .	. . .

```
data subset;
   do PickIt=1 to TotObs by 50;
      set orion.order_fact
          (keep=Customer_ID
                Employee_ID
                Street_ID
                Order_ID)
          point=PickIt
          nobs=TotObs;
         output;
   end;
   stop;
run;
```

Execution stops.

PDV

D PickIt	D Tot Obs	Customer_ID	Employee_ID	Street_ ID	Order_ID	D _N_
651	617	215	120175	1600102721	1243963366	1

...

Resulting Data Set

Systematic Sample

Obs	Customer_ID	Employee_ID	Street_ID	Order_ID
1	63	121039	9260125492	1230058123
2	17023	99999999	2600100021	1230931366
3	17	121037	9260123306	1231757107
4	195	120160	1600101663	1232590052
5	41	120134	1600101527	1233545775
6	11171	99999999	2600100032	1235176942
7	10	121043	9260129395	1237327705
8	53	120121	1600103258	1238674844
9	90	121040	9260111614	1239543223
10	89	121061	9260116551	1240549230
11	27	99999999	9260105670	1241930625
12	41	120195	1600101527	1242838815
13	215	120175	1600102721	1243963366

Creating a Random Sample

- Instead of creating a systematic sample, create a random sample where each observation has an equal chance of being selected.
- There are two types of random samples:
  - *with replacement*, where an observation might be selected more than one time
  - *without replacement*, where an observation cannot be selected more than once
- You can use the RANUNI function to generate random numbers from a uniform distribution.
- General form of the RANUNI function:

RANUNI(*seed*)

Using the RANUNI Function

- The RANUNI function returns a rational number between 0 and 1 (non-inclusive) generated from a uniform distribution.

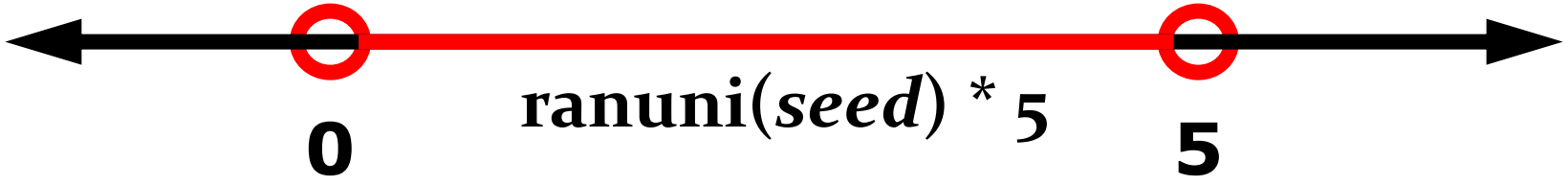
Examples:

Random number
.01253689
.95196500

...

Using the RANUNI Function

- If you want a ***number*** between 0 and 5 (non-inclusive), multiply the number returned from the RANUNI function by 5.



Examples:

Random number* 5

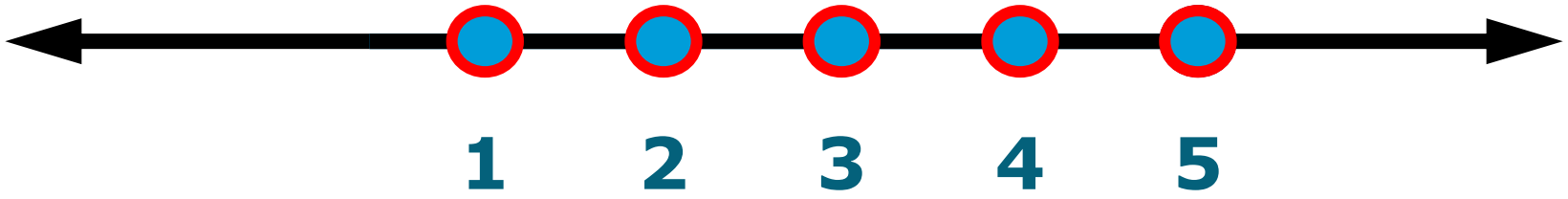
.01253689 → 0.06268445

.95196500 → 4.75982500

...

Using the RANUNI and CEIL Functions

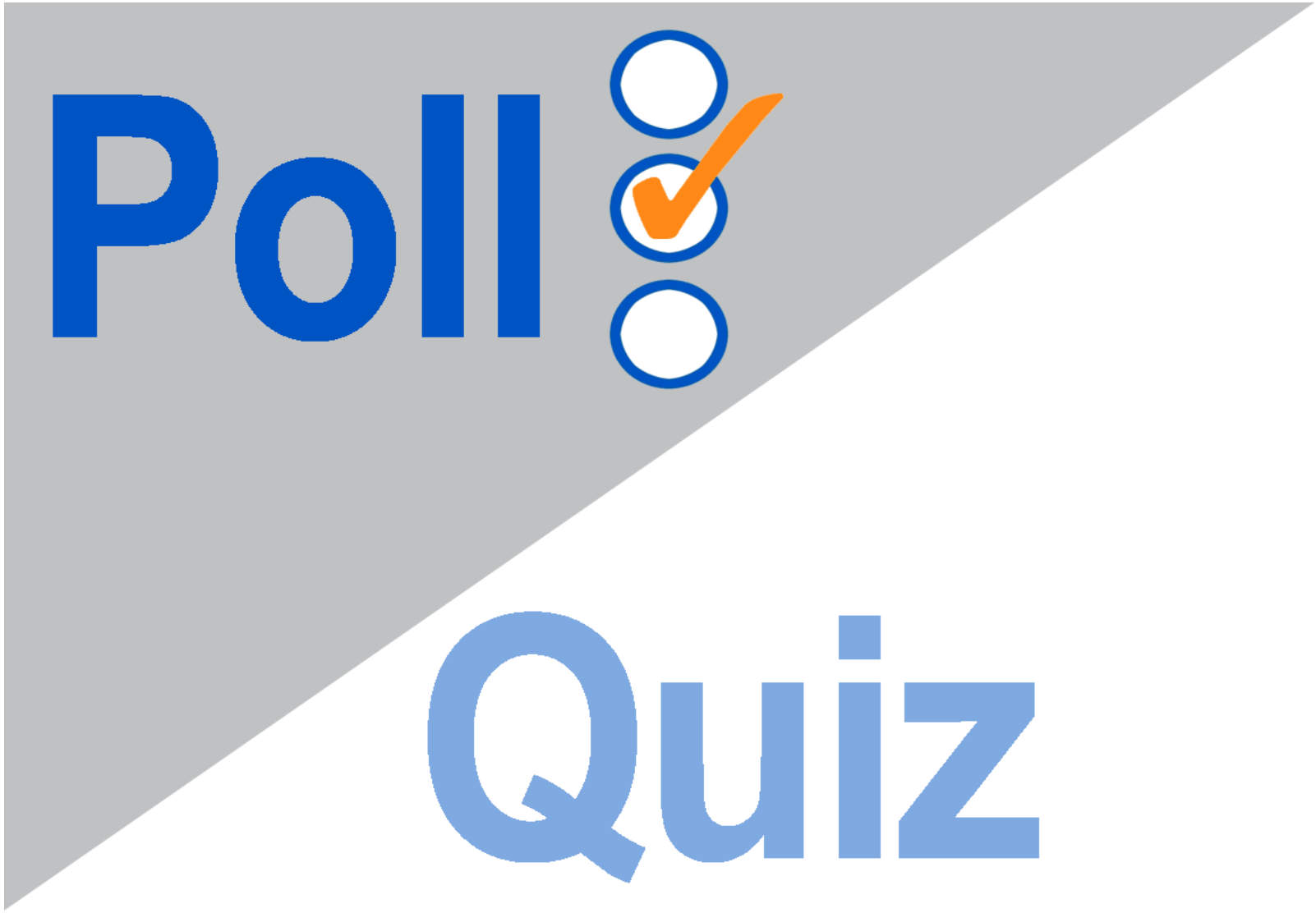
- If you want an ***integer*** between 1 and 5 (inclusive), use the CEIL function on the number returned by multiplying the random number by 5.



ceil(ranuni(*seed*) * 5)

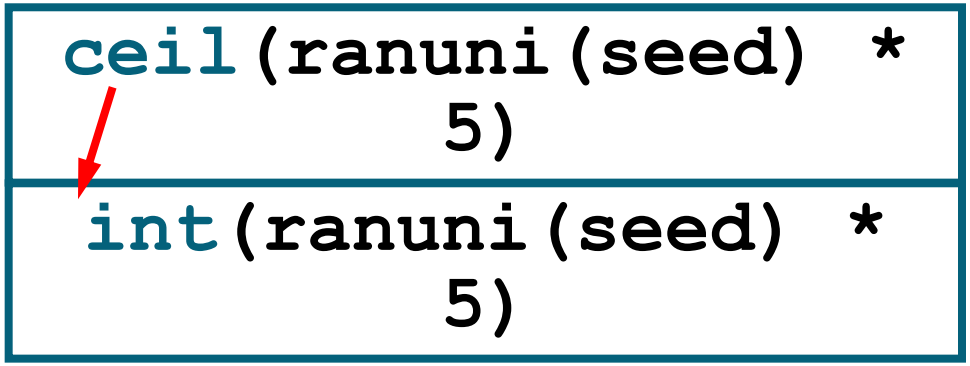
Examples:

Random number* 5	CEIL( )
.01253689	0.06268445 → 1
.95196500	4.75982500 → 5


Poll
Quiz

Setup for the Poll

- Instead of the CEIL function, would the INT function return the same results?



3.10 Poll

Instead of the CEIL function, would the INT function return the same results?

- Yes
- No

3.10 Poll – Correct Answer

Instead of the CEIL function, would the INT function return the same results?

- ○ Yes
- ○ No

The INT function returns the integer portion of its argument, which could possibly be 0 and never be 5.

Using the SURVEYSELECT Procedure

- The SURVEYSELECT procedure has the following attributes:
  - provides a variety of methods for selecting probability-based random samples
  - can select a simple random sample or can sample according to a complex multistage sample design that includes stratification, clustering, and unequal probabilities of selection
  - is part of SAS/STAT

Using the SURVEYSELECT Procedure

- This program creates a SAS data set, **ordersample**, that contains 10 observations randomly selected, without replacement, from the **orion.order_fact** SAS data set.

```
proc surveyselect data=orion.order_fact
                  (keep=Customer_ID
Employee_ID
                                Street_ID Order_ID)
                   out=ordersample
                   method=srs n=10;
run;
```

Using the SURVEYSELECT Procedure

- General form of the SURVEYSELECT procedure:

```
PROC SURVEYSELECT
options;
      STRATA variables;
      CONTROL variables;
      SIZE variable;
      ID variables;
RUN;
```

Using the SURVEYSELECT Procedure

- The PROC SURVEYSELECT statement performs the following tasks:
  - invokes the procedure
  - can, if you choose, identify input and output data sets
  - specifies the sample selection method, the sample size, and other sample design parameters
- The PROC SURVEYSELECT statement is the only statement required to create a simple random sample.

Options for the SURVEYSELECT Procedure

- The following options can be specified in the PROC SURVEYSELECT statement:

To do this:	Use this option:
Specify the input data set	DATA=
Specify the output data set	OUT=
Suppress displayed output	NOPRINT
Specify selection method	METHOD=
Specify sample size	SAMPSIZE= N=
Specify random number seed	SEED=

Methods Used by the SURVEYSELECT Procedure

- Selected values for the METHOD= option are as follows:

SYS	This method of systematic random sampling selects units at a fixed interval throughout the sampling frame or stratum after a random start.
URS	This method of unrestricted random sampling selects units with equal probability and with replacement. Because units are selected with replacement, a unit can be selected for the sample more than once.
SRS	This method of simple random sampling selects units with equal probability and without replacement. The selection probability for each individual unit equals n/N.

Using the SURVEYSELECT Procedure

- This program creates a SAS data set, **ordersample**, that contains 10 observations randomly selected, without replacement, from the **orion.order_fact** SAS data set.

```
proc surveyselect data=orion.order_fact
                  (keep=Customer_ID
Employee_ID
                                Street_ID Order_ID)
                   out=ordersample
                   method=srs n=10;
run;
```

p303d10

Using the SURVEYSELECT Procedure

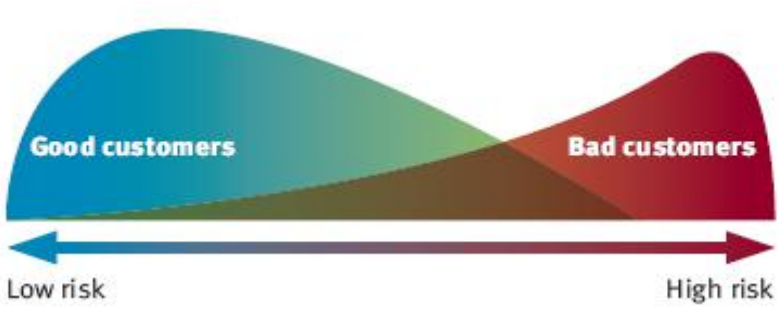
- In addition to creating the SAS data set, **ordersample**, PROC SURVEYSELECT provides the following information in the Output window:

```
The SURVEYSELECT Procedure

Selection Method     Simple Random Sampling

Input Data Set                ORDER_FACT
Random Number Seed             525990001      1
Sample Size                           10
Selection Probability          0.016207      2
Sampling Weight                    61.7      3
Output Data Set              ORDERSAMPLE
```

Comparison of the DATA Step and the SURVEYSELECT Procedure

DATA Step	PROC SURVEYSELECT
full power of DATA step processing	less coding
can create multiple output data sets	one output data set with additional statistics
part of Base SAS	part of SAS/STAT

4. Credit scoring- historie, základní pojmy



Úvod

- Credit Scoring je soubor prediktivních modelů a jejich základních technik, které slouží jako podpora finančním institucím při poskytování úvěrů.

- Tyto techniky rozhodují, kdo dostane úvěr, jaká má být výše úvěru a jaké další strategie zvýší ziskovost dlužníků vůči věřitelům.

- Credit Scoringové techniky kvantifikují a posuzují rizika při poskytování úvěrů konkrétnímu spotřebiteli.

Úvod

- Nerozeznají a nestanovují "dobré" nebo "špatné" (očekává se negativní chování, tj. např. default) žádosti o úvěr na individuální bázi, nýbrž poskytují statistické šance, nebo pravděpodobnosti, že žadatel s daným skóre se stane "dobrým" nebo "špatným".

- Tyto pravděpodobnosti nebo skóre, spolu s dalšími obchodními úvahami jako jsou předpokládaná míra schvalování, zisk nebo ztráty, jsou pak použity jako základ pro rozhodování o poskytnutí/neposkytnutí úvěru.

Why do we need score?

- “HISTORICAL EVOLUTION”:

Money lender

- lend only to people which he knows

Operators

- they make decision based on client's information and their experience

Automatic scoring

- make decision on statistical base

PAST EXPERIENCE -> ESTIMATION FOR FUTURE

Why score?

ADVANTAGES:

- Automatization of approval proces
- Cost – effective
- Less fraud possibilities

DISADVANTAGES

- Statistical based, not take in account client like individual

Úvod

- Zatímco historie úvěru sahá 4000 let nazpět (první zaznamenaná zmínka o úvěru pochází ze starověkého Babylonu - 2000 let před n.l.), historie credit scoringu je pouze 50-70 let stará.

- První přístup k řešení problému identifikace skupin v populaci představil ve statistice Fisher (1936). V roce 1941, Durand jako první rozpoznal, že tyto techniky mohou být použity k rozlišování mezi dobrými a špatnými úvěry.

Úvod

- Významným milníkem při posuzování úvěrů byla druhá světová válka.
- Do té doby bylo standardem individuální posuzování žadatele o úvěr. Dále bylo standardem, že ve finanční sféře byli zaměstnáni (téměř) výhradně muži.
- Odchod značné části mužské populace do služeb armády měl za následek potřebu předat zkušenosti dosavadních posuzovatelů žádostí o úvěr novým pracovníkům.
- Díky tomu vznikla jakási rozhodovací pravidla a došlo k „automatizaci“ posuzování žádostí o úvěr.

Úvod

- Příchod kreditních karet ke konci šedesátých let minulého století a růst výpočetního výkonu způsobil obrovský rozvoj a využití credit scoringových technik. Událost, která zajistila plnou akceptaci credit scoringu, bylo přijětí zákonů „Equal Credit Opportunity Acts" (o rovné příležitosti přístupu k úvěrům) a jeho pozdějších znění přijatých v USA v roce 1975 a 1976. Tyto stanovily za nezákonné diskriminace v poskytování úvěru, vyjma situace, pokud tato diskriminace „byla empiricky odvozená a statisticky validní".

Úvod

- V osmdesátých letech minulého století začala být využívána logistická regrese, dodnes v mnoha oblastech považovaná za průmyslový standard, a lineární programování. O něco později se objevily na scéně metody umělé inteligence, např. neuronové sítě. Mezi další používané techniky lze zařadit metody nejbližšího souseda, splajny, waveletové vyhlazování, jádrové vyhlazování, Bayesovské metody, regresní a klasifikační stromy, support vector machines, asociační pravidla, klastrová analýza a genetické algoritmy.

Historie -detail

Date	Event
2000 BC	First use of credit in Assyria, Babylon, and Egypt.
1100s	First pawnshops in Europe established by charitable institutions, and by 1350 they were being run as commercial concerns.
1536	Charging of interest deemed acceptable by the Protestant church.
1730	First advertisement for credit placed by Christopher Thornton of Southwark, London who offered furniture that could be paid off weekly.
1780s	First use of cheques in England.
1803	First consumer reports by Mutual Communications Society in London.
1832	First publication of the *American Railroad Journal*.
1841	Mercantile Agency is first American credit reporting agency.
1849	Harrod's established as one of the world's first department stores.
1851	First use of credit ratings for trade creditors by John M. Bradstreet.
1856	Singer Sewing Machines offers consumer credit.
1862	Poor's Publishing publishes *Manual of the Railroads of the United States*.
1869	First American consumer bureau is Retailers Commercial Agency (RCA) in Brooklyn.
1886	Sears established, and launches its catalogue in 1893.

Zdroj: Anderson

pawnshop = zastavárna
deemed acceptable = považován za přijatelný
Advertisement for credit = reklama na úvěr
Mercantile agency = obchodní agentura

Historie -detail

Date	Event
1906	National Association of Retail Credit Agencies formed in the USA.
1909	John M. Moody publishes first credit rating grades for publicly traded bonds.
1913	Henry Ford uses production lines to produce affordable automobiles.
1927	Establishment of Schufa Holdings AG, first credit bureau in Germany.
1934	First public credit registry (PCR) established in Germany.
1936	R.A. Fisher's use of statistical techniques to discriminate between iris species.
1941	David Durand writes report, suggesting statistics can assist credit decisions.
194?	Henry Wells uses credit scoring at Spiegel Inc.
1950	Diners Club and American Express launch first charge cards.
1950s	Sears uses propensity scorecards for catalogue mailings.
1956	FI consultancy established in California, USA.
1958	First use of application scoring by American Investments.
1960s	Widespread adoption of credit scoring by credit card companies.
1966	Credit Data Corp. becomes first automated credit bureau.
1970	Fair Credit Reporting Act governs credit bureaux.
1974	Equal Credit Opportunity Act causes widespread adoption of credit scoring.
1975	FI implements first behavioural scoring system for Wells Fargo.
1978	Stannic implements first vehicle finance scorecards in South Africa.
1982	CCN offers Credit Account Information Sharing (CAIS), its consumer credit bureau service.
1984	FI develops first bureau scores used for pre-screening.
1987	MDS develops first bureau scores used for bankruptcy prediction.
1995	Mortgage securitisers Freddy Mac and Fannie Mae adopt credit scoring.
2000	Moody's KMV introduces RiskCalc for financial ratio scoring (FRS).
2000s	Basel II implemented by many banks.

Zdroj: Anderson

affordable = dostupný
iris species = druhy kosatců
Charge card = kreditní karta
Propensity scorecard = scoringová karta pro modelování náchylnosti (k nákupu)
FI = společnost Fair, Isaac…dnes FICO
Mortgage = hypotéka

Historie -detail

Table 2.4. Genealogies and milestones—credit cards

Date	Event
1914	Western Union introduces embossed metal plate first charge card in the United States.
1920s	Introduction of 'shopper's plates', early version of modern store cards.
1950	Diners Club and American Express launch first charge cards.
1951	Diners Club launches first credit card in New York city.
1960	Bank Americard established, later to become Visa.
1966	Master Charge established, later to become MasterCard.
1966	Barclaycard established in the United Kingdom.

Table 2.5. Genealogies and milestones—credit scoring consultancies

Name	Year	Notes
Fair Isaac (FI)		
FI	1956	Founded San Francisco CA, by Bill Fair and Earl Isaac
	1958	First scorecard development, for American Investments
	1984	Develops first bureau score for pre-screening
	1995	First use of scoring by mortgage securitisers
Experian-Scorex		
Management Decision Systems (MDS)	1974	Founded by John Coffman and Gary Chandler
	1982	MDS purchased by CCN
Scorex	1984	Founded in Monaco by Jean-Michel Trousse
MDS	1987	MDS develops first monthly bureau score, for bankruptcy
Experian-Scorex	2003	Created as subsidiary of Experian, after purchase of Scorex

Zdroj: Anderson

Historie -detail

Table 2.7. Genealogies and milestones—credit bureaux

Name	Year	Notes
Dun & Bradstreet		
Mercantile Agency	1841	Founded, New York NY, by Lewis Tappan.
	1849	Benjamin Douglass takes over, and expands.
John M. Bradstreet Co.	1849	Founded, Cincinnati OH.
	1851	First use of credit rating grades.
R.G. Dun & Co.	1859	Robert G. Dun incorporates Mercantile Agency.
Dun & Bradstreet	1933	Merger orchestrated by Arthur Whiteside.
Experian		
Manchester Guardian Society	1827	Founded, Manchester, UK.
Chilton Corp.	1897	Founded, Dallas TX. Publishes 'Red Book'.
Michigan Merchants	1932	Founded, later to become Credit Data Corp.
TRW	1968	Purchases Credit Data Corp., and changes name to TRW-Credit Data.
TRW	1976	Information Systems and Services (IS&S) division produces first business credit report.
CCN	1980	Founded, when Great Universal Stores (GUS) spins off information services division
	1884	Purchases Manchester Guardian Society
TRW	1989	Purchases Chilton Corp.
Experian	1996	Founded, through TRW divestiture of TRW-CD & IS&S. Purchased by GUS, who merges it with CCN.
Equifax		
London Assn. for the Protection of Trade	1842	Founded, London, UK
RCA	1869	Founded, Brooklyn, NY
RCC	1899	Founded, Atlanta, GA
	1934	Purchases RCA
United Assn. for the Protection of Trade	1965	LAPT renamed
Equifax	1975	RCC renamed to Equifax
	1994	Purchases UAPT-Infolink and Canadian Bonded Credits
TransUnion		
TransUnion	1968	Founded, as holding company for Union Tank Car Company (UTCC)
	1969	Purchases the Credit Bureau of Cook County

Zdroj: Anderson

Historie -detail

Table 2.8. Genealogies and milestones—credit rating agencies

Name	Year	Notes
Standard & Poor's (S&P)		
Poor's Publishing Co.	1862	Founded, by Henry Varnum Poor
S&P	1941	Poor's Publishing and Standard Statistics merge
Moody's Investor Services (MIS)		
John Moody & Co.	1900	Founded, by John Moody, but fails in 1907
John Moody	1909	First use of rating grades for bonds
Moody's Investor Services	1914	Incorporation of MIS
	1962	MIS purchased by D&B
Moody's KMV	2002	Created as MIS subsidiary after merger of Risk Management Services and KMV
Fitch IBCA		
Fitch Publishing Co.	1913	Founded, by John Knowles Fitch
IBCA	1978	Founded
Fitch IBCA	1997	Merger of Fitch Publishing and IBCA

Zdroj: Anderson

Historie –další zajímavé čtení

http://www.fundinguniverse.com/company-histories/Fair-Isaac-and-Company-Company-History.html

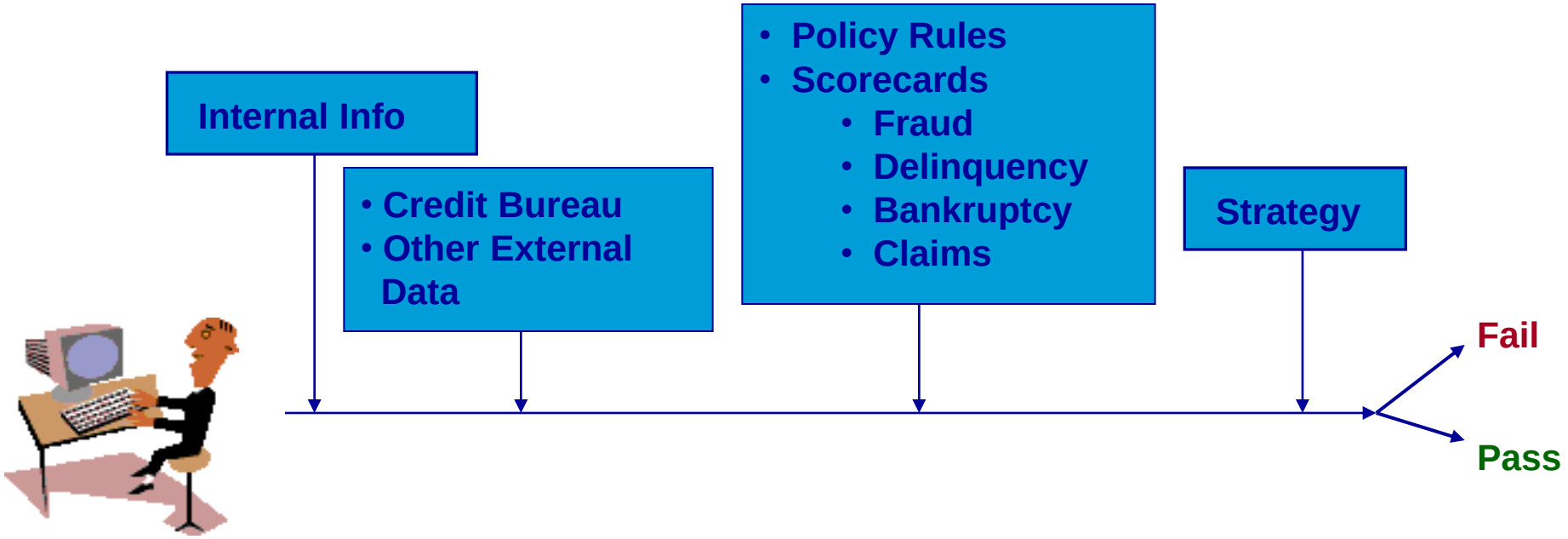
http://www.fico.com/en/Company/News/Pages/03-10-2009.aspx

http://www.directlendingsolutions.com/history_credit_scoring.htm

http://www.pbs.org/wgbh/pages/frontline/shows/credit/more/scores.html

http://en.wikipedia.org/wiki/Credit_score

Risk Management – Acquisition



Data Acquisition

Risk Management – Customer

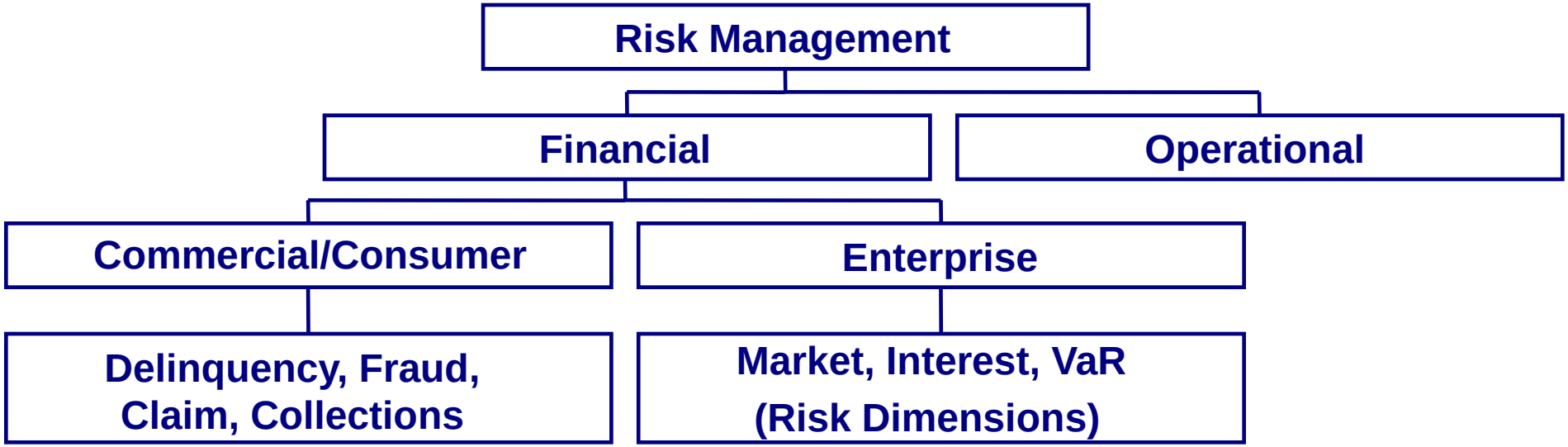
- Credit Line Management
- Usage Monitoring
- Transaction Fraud
- Transaction Approval
- Renewal/Reissue

- Collections
- Claims

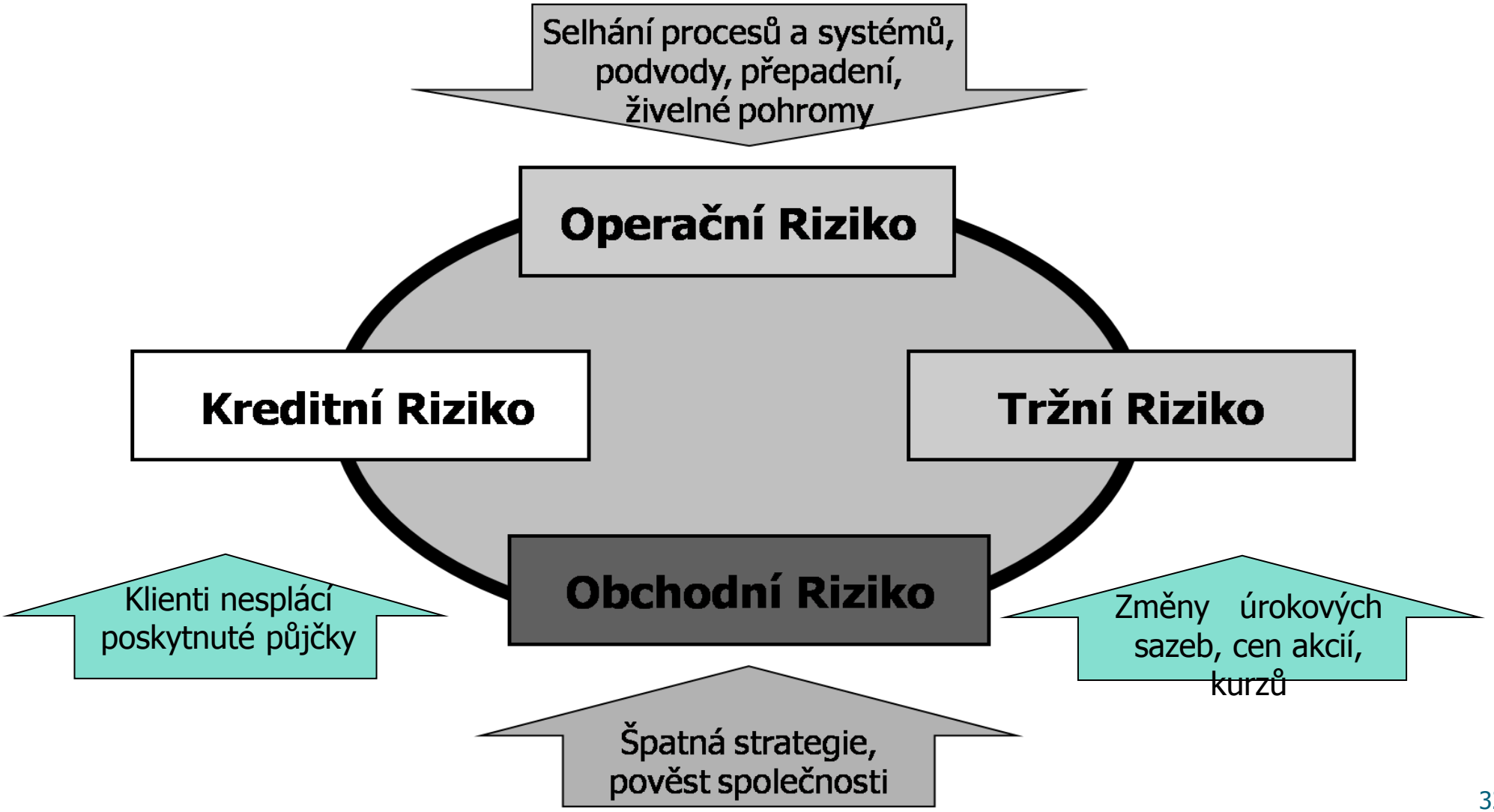
√ Scorecards
√ Policy Rules
√ Strategies

.. Lots of analysis

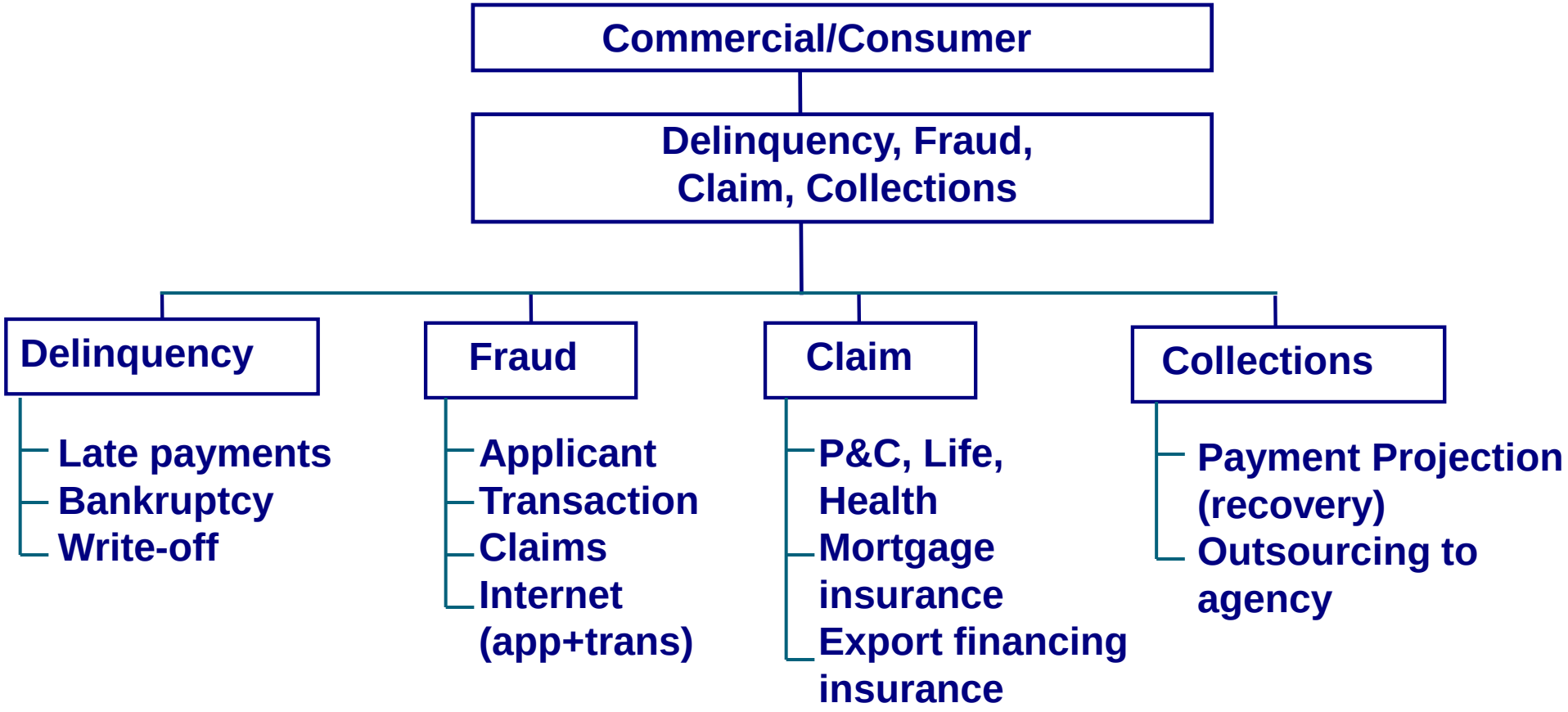
Risk Management



Risk Management a druhy rizik



Risk Management



P&C: Property & Casualty Insurance (majetkové a úrazové pojištění)

Why Manage Risk?

- Reduce exposure to high-risk accounts.
- Decrease bad debt and claims payouts.
- Ensure better pricing to reflect risk.
- Detect fraud early-on.
- Increase approval rates (the "right kind" – potentially increasing revenue).
- Handle most approvals/declines quickly (customer service).
- Analysts/investigators only focus on difficult accounts.
- Ensure consistent, equal and objective treatment of each applicant across the organization.
- Offer more efficient marketing initiatives.

Users of Risk Management

- Banks
  - Citibank, Royal Bank, CIBC, BankOne
- Finance Companies
  - GE Capital, HFC, GMAC
- Insurance
  - Life, Property and Casualty, Health
- Government
  - Ministries/Departments of Health (Medicare), Ministries of Finance (IRS), Workers Compensation.

Users of Risk Management

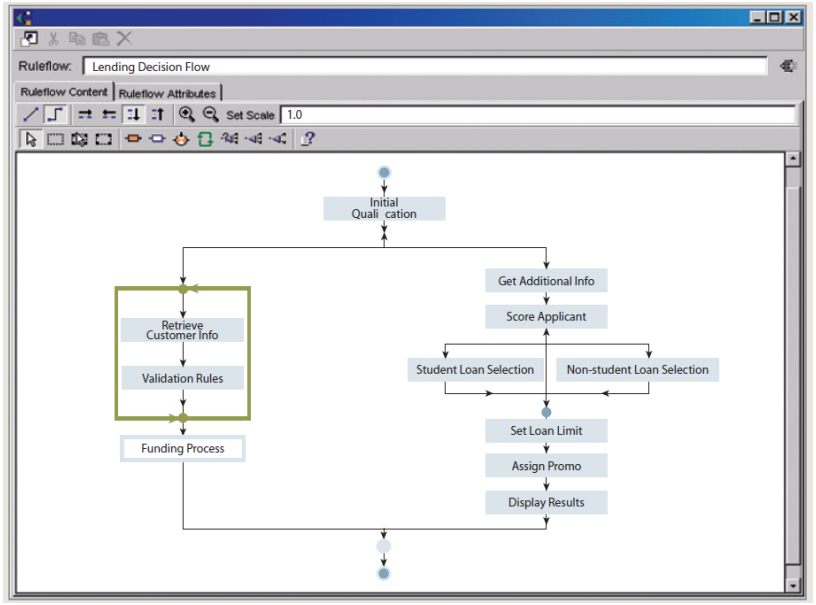
- Utilities
  - Hydro/Power/Energy, Water
- Communications
  - Bell, Sprint, AT&T (land lines and cellular)
- Retail
  - JC Penneys, Sears, Hudsons Bay Company, Target
- Manufacturers/Industrials
  - Those who give credit to small businesses.

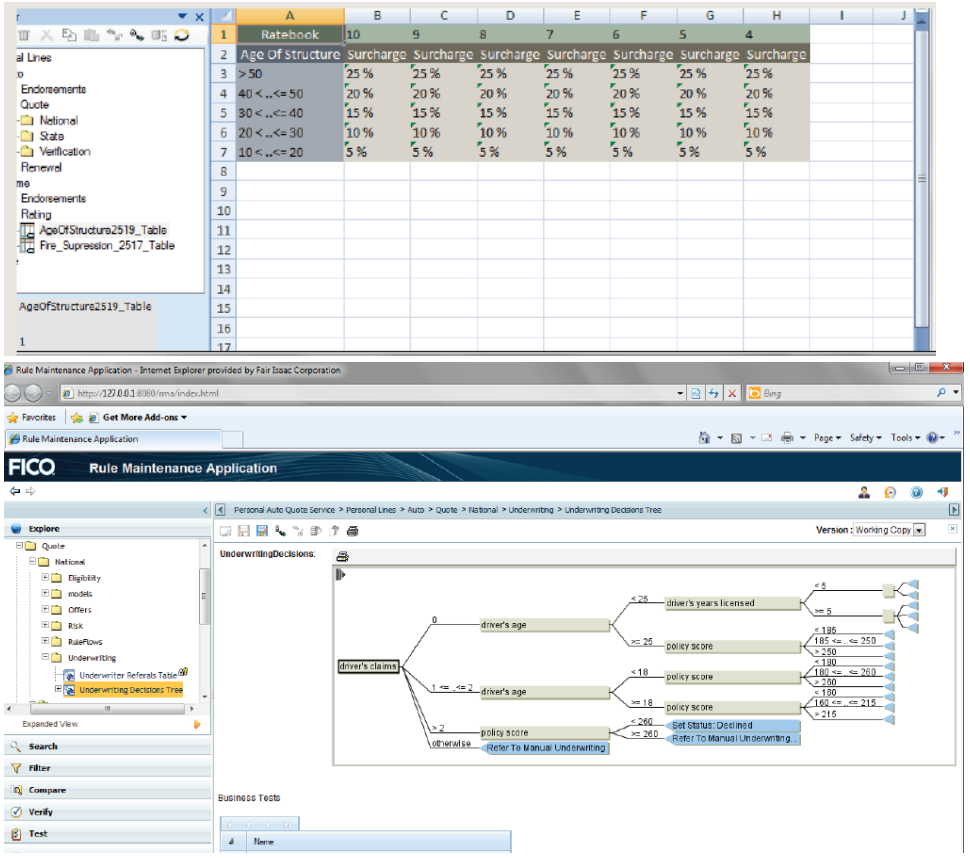
Risk Management "Toolbox"

- Risk Data Mart/Data Warehouse
- Risk prediction models (scorecards)
- Reporting
- Analysis tools

- Operational/strategy implementation software (for example, FICO™ Blaze Advisor®, FICO® TRIAD® Customer Manager, Experian Probe SM, Experian NBSM, Cardpac, VisionPlus, Pro-Logic Ovation).

FICO™ Blaze Advisor®



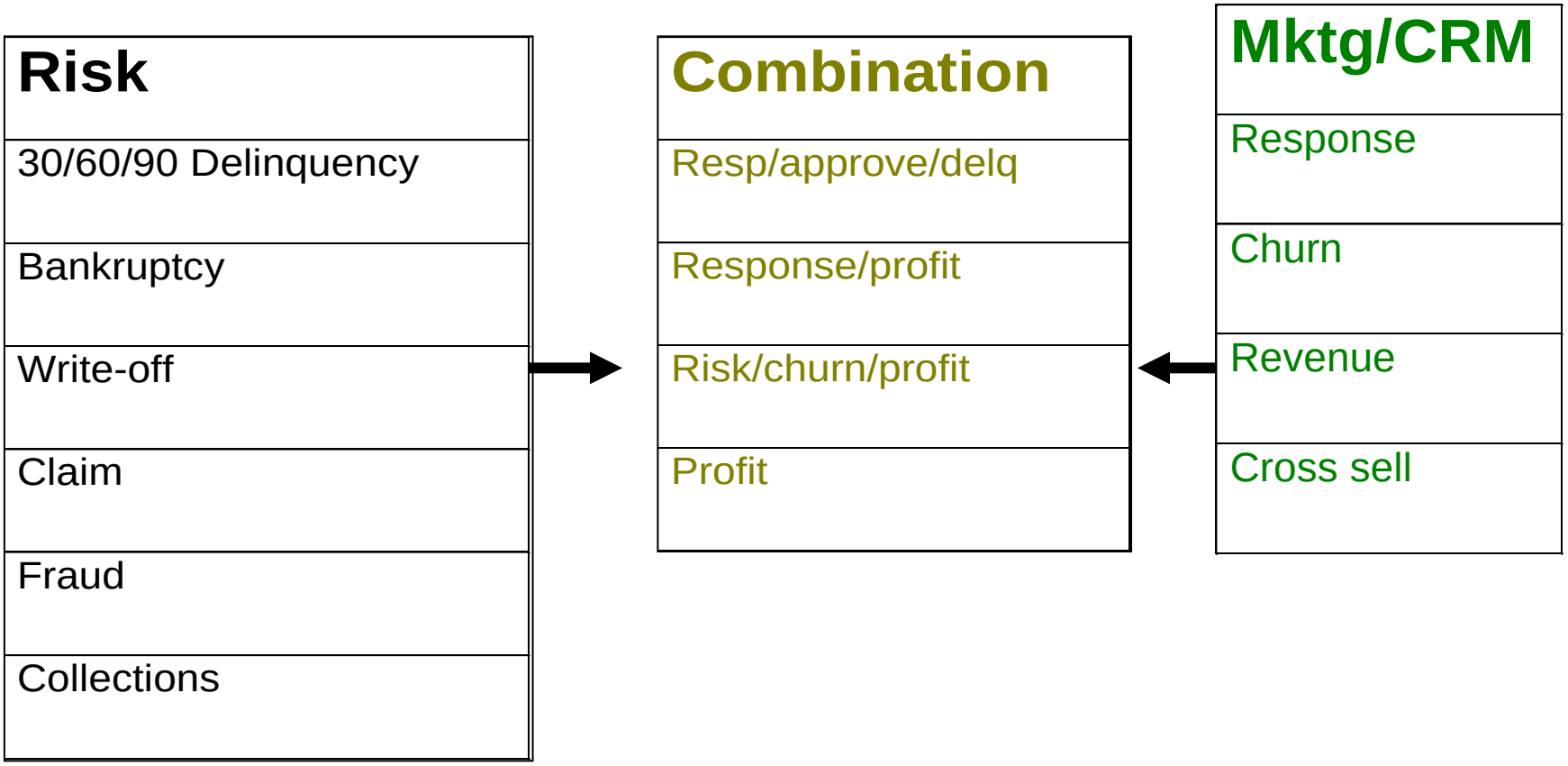


Zdroj: http://www.fico.com/account/resourcelookup.aspx?theID=430

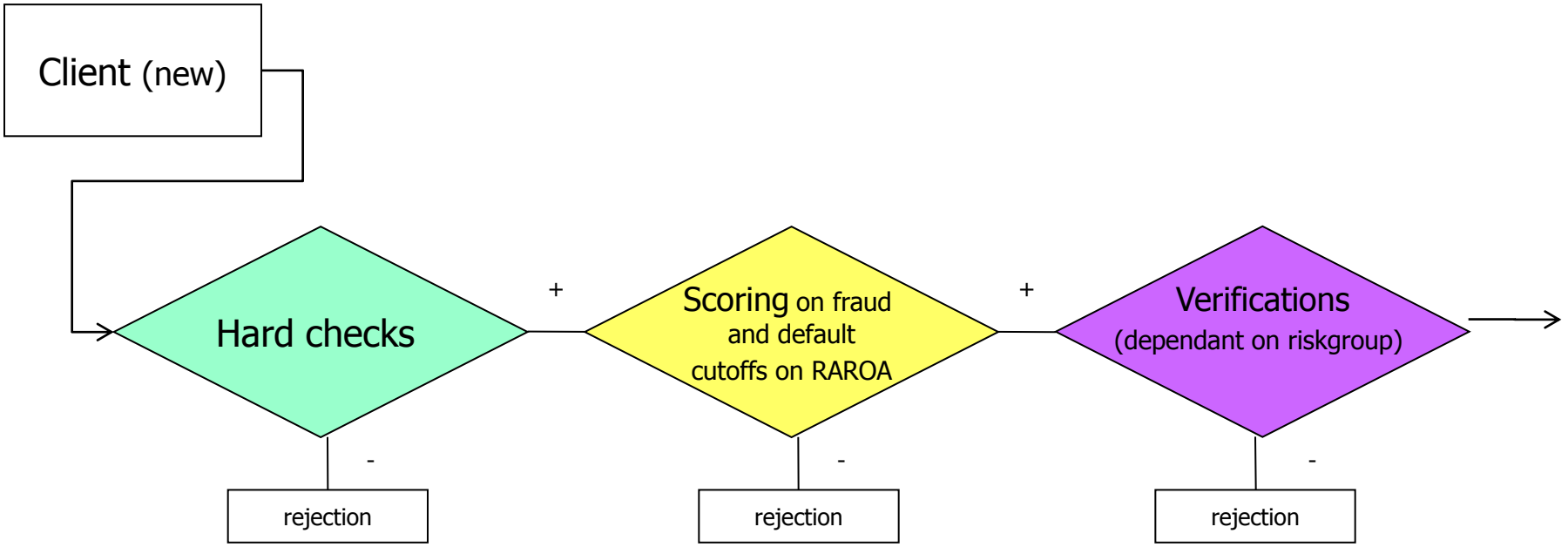
Scorecards

- Predict the probability of a negative event.
  - Custom – based on clients own data
  - Generic – based on pooled industry or bureau data (Beacon, Empirica)
  - Application – new applicants
  - Behavioral – current customers

Scorecard Types



Scoring in approval process



Policy declines – low age, unsufficient length of employment, "terorrist" etc.

What is the probability that client will pay?

Will the contract be profitable?

Is the number of client's phone valid? Etc.

Fraud Risk

- Fraud risk is one of the fastest growing areas in risk management.
- Examples include bank/retail card fraud, insurance fraud, health care fraud, welfare fraud, franchise fraud, internet fraud, mortgage fraud, investment fraud, tax fraud, merchant fraud.
- E-commerce presents opportunities.
- The F.B.I. estimates that between 10–15% of loan applications contain material misrepresentations.

Reporting and Analysis

- Scorecard and portfolio performance
- Approval rates, applicant profile, loss rates, high risk segments
- Behavior tracking to develop better strategies
- Capturing fraud, approval/decline, pricing, credit line management, collections, cross sells qualification, claims.

Risk Applications

- Retail/banking (consumer and commercial)
  - Application and behavior scorecards for all credit products.
  - Strategy design for credit limit setting, authorizations and collections/reissue/suspension.
  - Fraud application and transaction detection
  - Pricing/down payment
  - ATM limits, check holds
  - Pre-qualifying direct marketing lists.
- Automotive/finance
  - Loans and leasing
    - Application, behavioral, fraud, collection scorecards
  - Pricing/down payment.

Risk Applications

- Government
  - Fraud detection (for example, Welfare, health insurance)
  - Entitlement/claims assessment (for example, Workers compensation)
- Communications
  - Security deposit
  - International call access
  - Contract/"pay as you go"
  - Telephone fraud
  - "Shadow limit" setting
  - Suspension of service
  - Collections.

Risk Applications

- Insurance
  - Rate setting
  - Fraud detection
  - Claims management
  - Risk control for CRM initiatives.
- Utilities
  - Security deposit
  - Collections.

Risk Applications

- Manufacturers/pharmaceuticals/industrials
  - Assessing credit risk of business clients
  - Credit risk assessment of franchisees (for example, gas stations)
  - Payment terms
  - Collections
  - Merchant fraud.

Risk Applications

- Optimizing work flow in adjudication departments
- Evaluating/pricing portfolios
- Securitization
- Setting economic/regulatory capital allocation
- Reducing turnaround time (automated scoring)
- Comparing quality of business from different channels/regions/suppliers.

Resources

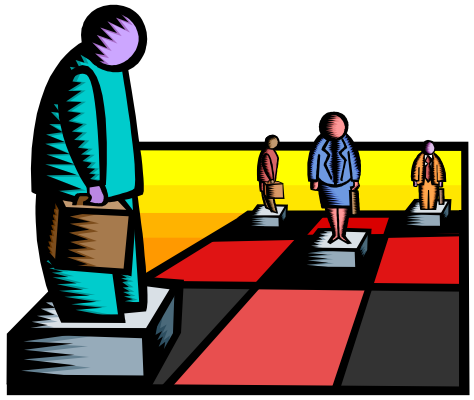
- www.ftc.gov/bcp/conline/pubs/credit/scoring.htm
- www.creditscoring.com
- www.my-credit-score.com
- www.fairisaac.com, www.myfico.com
- www.experian.com
- www.creditinfocenter.com
- www.consumersunion.org/finance/scorewc200.htm
- www.phil.frb.org/files/br/brs097lm.pdf
- www.nacm.org
- www.rmahq.org
- www.riskmail.org
- www.occ.treas.gov

Resources

- **Credit Scoring & Its Applications**
  by Lyn Thomas, Jonathan Crook, David Edelman
- **Credit Risk Modeling: Design and Application**
  by Elizabeth Mays (Editor)
- **Internal Credit Risk Models: Capital Allocation and Performance Measurement**
  by Michael K Ong
- **Handbook of Credit Scoring**
  by Elizabeth Mays
- **Applications of Performance Scoring to Accounts Receivables Management in Consumer Credit**
  by John Y. Coffman
- **Introduction to Credit Scoring,**
  by E.M. Lewis

Scorecard Development roles-objectives

- Understand the critical resources needed to successfully complete a scorecard development and implementation project.
- Understand some of the operational considerations that go into scorecard design.



Major Roles

- Scorecard Developer
  - Data miner, data issues
- Credit Scoring Manager/Risk Manager
  - Strategic view, corporate policies, implementation
- Product Manager
  - Client base, target market, marketing direction.

Major Roles

- Operational Managers
  - Customer Service, Adjudication, Collections
  - Strategy execution, impact on customers
- IT/IS Managers
  - external/internal data, implementation platforms.

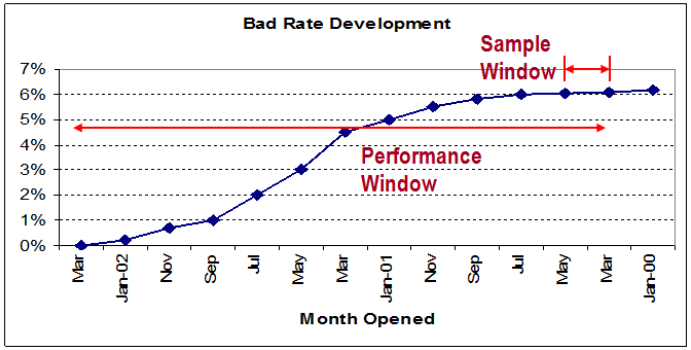
Minor Roles

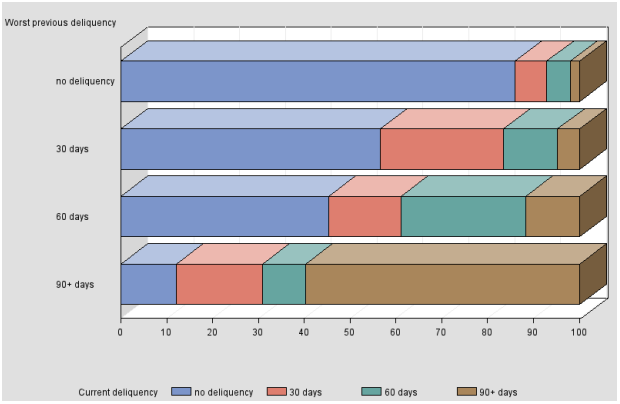
- Project Manager
  - Coordination, time lines
- Corporate Risk staff
  - Corporate policies, capital allocation
- Legal.

Why All of These Roles?

- Can I use this variable?
  - Legal, technical (derived variables, implementation platform), future application form design
- Segmentation
  - Marketing, application form design, systems
- What is the impact on this segment?
  - Operational, marketing, risk manager, corporate risk.

5. Metodologie vývoje scoringových funkcí





Objectives

- Understand how scorecards to predict credit risk are developed.
- Understand the analyses and issues for implementation of scorecards.

Main Stages – Development

- Stage 1: Preliminaries and Planning
  - Create Business Plan
    - Identify organizational objectives
    - Internal versus External development, and scorecard type
  - Create Project Plan
    - Identify project risks
    - Identify project team.

Main Stages – Development

- Stage 2: Data Review and Project Parameters
  - Data availability and quality
  - Data gathering for definition of project parameters
  - Definition of project parameters
    - Performance window and sample window
    - Performance categories definition (target)
    - Exclusions
  - Segmentation
  - Methodology
  - Review of implementation plan.

Main Stages – Development

- Stage 3: Development Database Creation
  - Development sample specification
  - Sampling
  - Development data collection and construction
  - Adjusting for prior probabilities.

Main Stages – Development

- Stage 4: Scorecard Development
  - Missing values and outliers
  - Initial characteristic analysis
  - Preliminary scorecard
  - Reject inference
  - Final scorecard production
    - Scaling
    - Points allocation
    - Misclassification
  - Scorecard strength
  - Validation.

Main Stages – Development

- Stage 5: Scorecard Management Reports
  - Gains tables and charts
  - Characteristic reports.

Main Stages – Implementation

- Stage 1: Pre-Implementation Validation

- Stage 2: Strategy Development
  - Scoring strategy
  - Setting cutoffs
  - Strategy considerations
  - Policy rules
  - Overrides.

Main Stages – Post Implementation

- Post-Implementation
  - Scorecard and Portfolio Monitoring Reports
  - Review.

Development

Stage 1: Preliminaries and Planning

Objectives

- Create a business plan to ensure a viable and smooth project.

- ***"All Models are wrong. Some are useful"***

George Box

Create Business Plan

- Identify organizational objectives.
  - Reasons for model development
    - Profit, revenue, loss, automation, operational efficiency
  - Role of scorecards in decision making
    - sole arbiter or decision support tool?

Create Business Plan

- Internal/External Development and Scorecard Type
  - Capability and resources
    - Staff, tools, expertise, data
    - Market segment
  - Custom, generic, judgmental
    - segment, data, time.

Create Project Plan

- Scope and timelines
- Deliverables (scorecard format and documentation,…)
- Implementation strategy
  - Testing, coding
  - Strategy development
  - FYI list.

- Seamless process from planning to development and implementation.

Create Project Plan

- Identify Project Risks
  - Data risks
    - Availability, quality, quantity
    - Weak data
  - Operational risks
    - Organizational priority
    - Implementation delays
    - System interpretation of data.

Create Project Plan

- Identify Project Team
  - Roles clearly defined
    - Signoff, executor, advisor, FYI
    - Critical path.

Development

Stage 2: Data Review and Project Parameters

Objectives

- Identify data requirements.
- Perform pre-modeling analysis.
  - Understand the business
  - Exclusions
  - What is a “bad”? – target definition
  - Sample Window/ Performance Window.

Data Availability and Quality

- Number of “goods”, “bads” and “rejects”
  - Initial idea at this stage, estimated from performance reports
- Internal data
  - Reliable, accessible
- External data
  - Accessible, format
  - Retro pull.

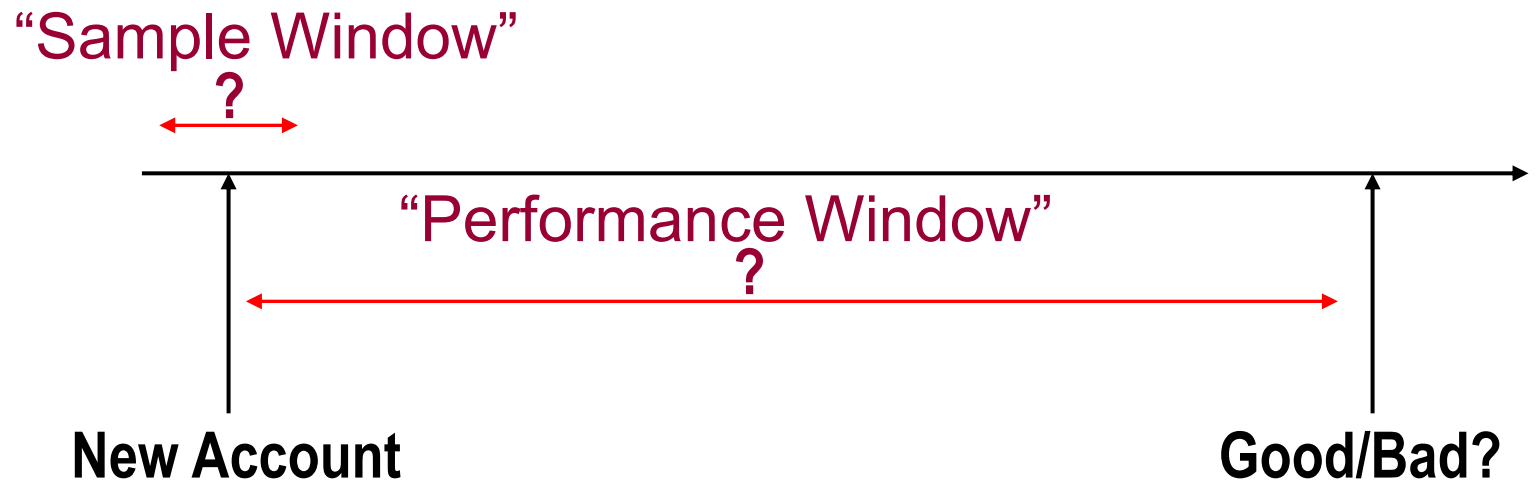
Data Gathering

- To determine “bad” definition and exclusions:
  - All applications over the last 2–5 years (or a large sample)
  - account/ID number
  - Date opened/applied
  - Accept/reject indicator
  - Arrears/payment history
  - Product/channel and other identifiers
  - Account status
  - Other items to understand the business.

Exclusions

- “Include those whom you would score during normal day to day operations”
  - VIP
  - Staff
  - Fraud
  - Pre-approved
  - Underage
  - Cancelled (sometimes).

Performance



Parameters

- Performance Window
  - How far back do I go to get my sample?
- Sample Window
  - Time frame from which sample will be taken.
- Definition of “bad”
- Bad and approval rates (when oversampling).

Parameters

- Seasonality
  - Plot approval rate/applications across time
  - Establish any ‘abnormal’ zones (for example, talk to marketing).
- Sample used in development must be from a normal business period, to get as accurate a picture as possible of the target population.

Parameters – "Bad"

- Plot "bad" rate by "month opened" (cohort)
- For different definitions of bad
  - 30/60/90 days past due
  - Charge off/write-off
  - Bankrupt
  - Claim
  - Profit based
  - Less than *x*% owed collected
- "Ever" versus "Current" bad
  - Ever bad should be used where possible
  - Considered "bad" if you reach status anytime during performance window.

Cohort Analysis – Example

Bad = 90 days					
Open Date	**1 Qtr**	**2 Qtr**	**3 Qtr**	**4 Qtr**	**5 Qtr**
Jan-99	0.00%	0.44%	0.87%	1.40%	2.40%
Feb-99	0.00%	0.37%	0.88%	1.70%	2.30%
Mar-99	0.00%	0.42%	0.92%	**1.86%**	**2.80%**
Apr-99	0.00%	**0.65%**	**1.20%**	**1.90%**	
May-99	0.00%	0.10%	0.80%	1.20%	
Jun-99	0.00%	0.14%	0.79%	1.50%	
Jul-99	0.00%	0.23%	0.88%		
Aug-99	0.00%	0.16%	0.73%		
Sep-99	0.00%	0.13%	0.64%		
Oct-99	0.20%	**0.54%**			
Nov-99	0.00%	**0.46%**			
Dec-99	0.00%	0.38%			
Jan-00	0.30%				
Feb-00	0.00%				
Mar-00	0.00%				

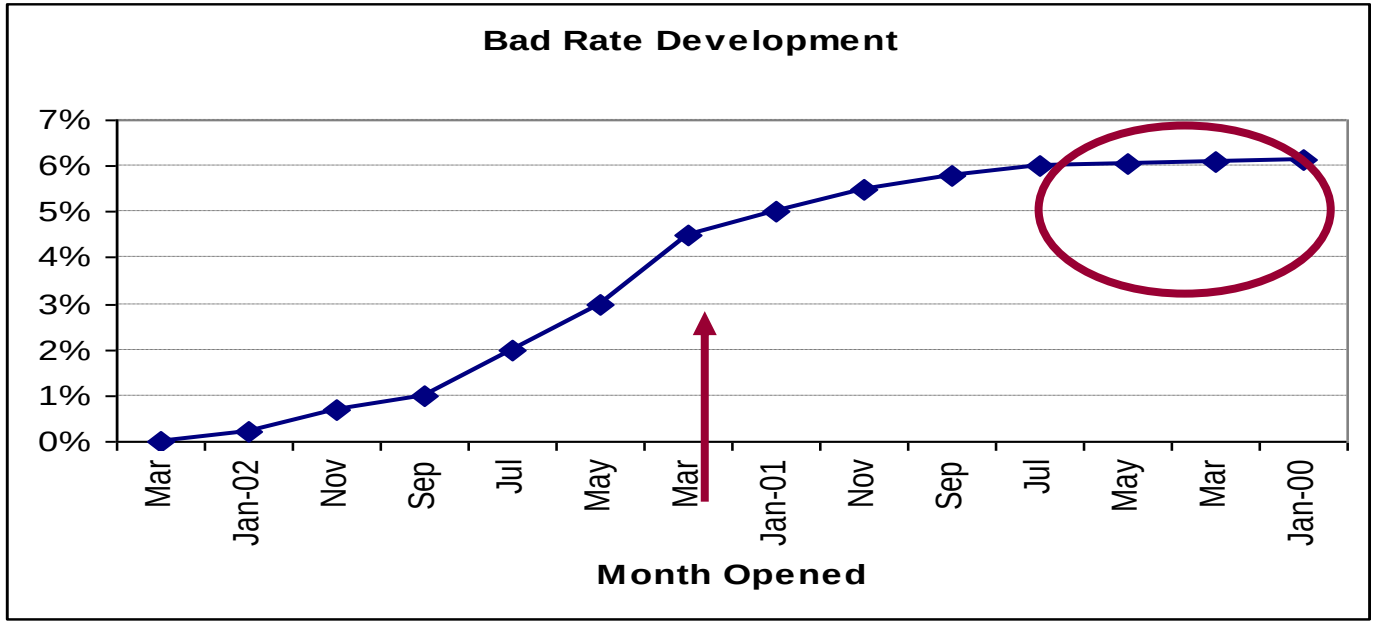
Current versus Ever – Example

- Current bad definition: No Delinquency
- Ever bad definition: 3 months delinquent.

Month	1	2	3	4	5	6	7	8	9	10	11	12
Delq	0	0	1	1	0	0	0	1	2	3	0	0

Month	13	14	15	16	17	18	19	20	21	22	23	24
Delq	0	0	1	2	0	0	0	1	0	1	0	0

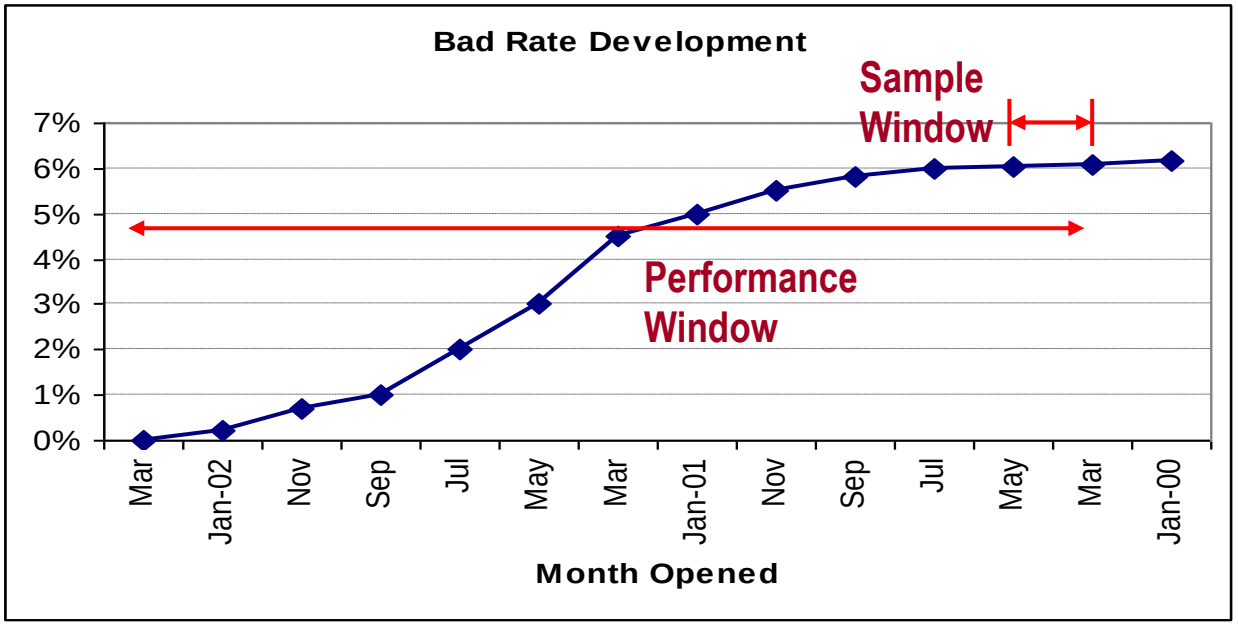
Determining Parameters



- mth opened from earliest to latest, and "bad rate" as of this month. For simplicity, this is straight delinquency .. No profit.
- notice at one point the bad rate levels off - this means everyone who was going to go bad has gone bad I.e. they have been given enough time. This is telling us that for this bad defn, accts from jan-march are mature enough.
-lesson 1: need sample that is mature enough, so that you wont be defining a "bad" as a good just because you haven't given them enough time.
-if you take accts from the middle (enter), some of the accts haven't matured yet so your bad rate is understated.

-Example: response scoring .. How long do you wait for the responses to come in. the period of measurement is 'perf window'.

Determining Parameters



So for each definition of "bad" you'll get a sample window of mature accounts, and a performance window indicating the time taken for the bad rate to mature. Also the approval rate for this sample window.

Couple of notes on this "maturing" process.

- 30 day definition will mature quicker than 90 day. Cause it takes ppl less time to go 30 day than 90 day. Chargeoff even more.
- for the same bad defn, credit card quicker than mortgage (18-24 mths vs. 3-5 yrs) .

- Why are we doing all this for the different definition?
- because each one will produce different counts and based on reasons on the next slide, we'll determine the best set of parameters.

Determining Parameters – Bad

- Organizational objectives/purpose
- Tighter definition – more precise, low counts
- Looser definition – differentiation sub-optimal
- Interpretable and trackable
- Consistency
- Reality – the best definition under the circumstances (lack of data, history).

Lets look at the considerations.
- objectives: this may seem obvious, but it is not to a lot of ppl. If you're building a scorecard to predict profit, then use profit. Some orgs want a delinquency based defn, but also include profit. E.g. if acct is chronically 2 mths late, but still profitable.. You can't set 2 mths as a "bad" - whereas in a pure delq scorecard this may be possible.
- tighter/looser: tighter means 90 day, 120 day, writeoff .. Better differentiation, but low count. Remember 2000 bads.
- looser means more count, but sub-opt diff.
- interpretable e.g. bad is 2 times 60 days, 3 times 30 days or 1 times 90 days. Sounds good, but hell to track and interpret. Keep it simple.
- consistency across other cards, products. Also if accounting writes off acct at 7 mths, then keep it consistent with that.
- typically most delq cards are 90 days.

- Reality: you take what you got. Lack of history allows only a 30 day definition .. Take it. Can't measure real bad rate .. Use proxy. (example LOC like an account)

Sample Definitions – Bad

- Ever 90 days delinquent
- Bankrupt
- Claim over $1000
- 3 x 30 days, or 2 x 60 days, or 1 x 90 days
- Negative NPV
- Not profitable
- 50% recovered within 3 months
- Fraud over $500
- *Closed within 6 months*.

Confirming “Bad” Definition

- Analytical
  - “Roll rate” analysis
  - Current versus worst delinquency comparison
  - Profitability analysis

- Consensus.

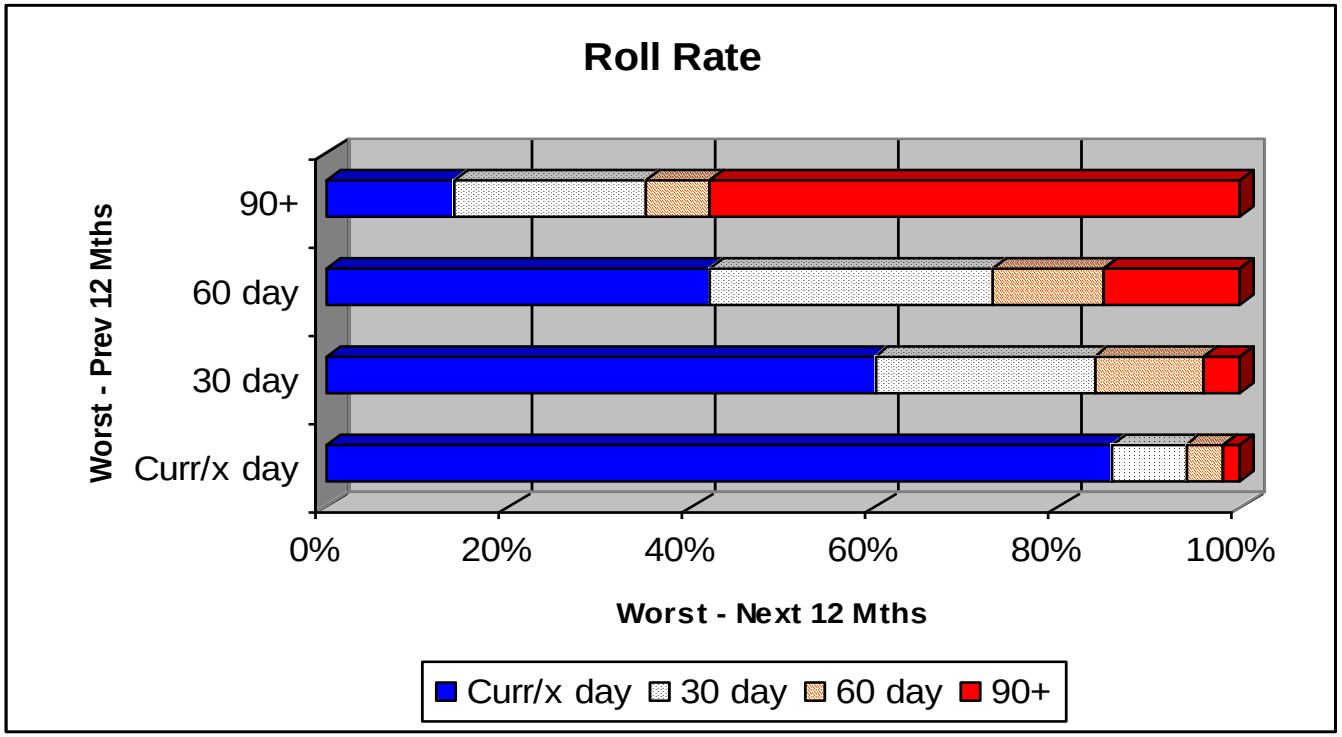
Roll Rate Analysis

- Compare Worst delinquency
  - for example, Previous 12 months versus Next 12 Months

Month	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
Arrears	0	0	1	2	0	0	0	1	2	3	0	0

Month	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*
Arrears	1	2	3	3	3	4	3	0	0	0	0	0

Roll Rate Analysis



You find out which 'bad defn' is truly bad' - also known as POINT OF NO RETURN.

Lets look at 30 day: out of everyone who had worst 30 day, majority became current, only a few became worse - this is not a good bad defn.
- out of those 60 days, some went over .. Most went back I.e. became better

-but those who were 90 day .. **Majority did not become better. This confirms our definition.**

-**In general .. Once you hit 90 days, you're not coming back. That's a true bad. Rem: this is based on 'bad' objective. If other, perhaps there is a different point in time..**

Roll Rate Analysis

- Look for ‘point of no return’.
- Consider objectives.
- Consider sample counts.
- Typically for delinquency, after 90 days most accounts do not cure.

Current versus Worst Comparison

				Worst Delinquency			
		Current	30 days	60 days	90 days	120 days	writeoff
Current	Current	100%	68%	34%	15%	4%	
Delinquency	30 days		16%	22%	8%	5%	
	60 days		8%	19%	17%	8%	
	90 days		4%	14%	32%	11%	
	120 days		2%	8%	18%	54%	
	writeoff		2%	3%	10%	18%	100%

32% 56% 40% 18% 44% 60% 72%

Parameters – Goods/Indeterminates

- Good
  - Never delinquent
  - Ever *x*- days delinquent
  - No claims
  - Profitable, positive NPV
  - No fraud
  - No bankruptcy
  - Recovery > 75%, $ value
  - Must be good throughout performance window
- Indeterminate
  - Mild delinquency, roll rate not conclusive either way
  - Inactive
  - Offer declined
  - Voluntary cancellations*
  - High balance < $50

Default – definice cílové prom. (good/bad)

- Obvykle je tato definice založena na klientově počtu dnů po splatnosti (Days Past Due, DPD) a částce po splatnosti. S částkou po splatnosti je spojena potřeba stanovení jisté míry tolerance, tedy stanovení co je považováno za významný dluh a co nikoli. Např. nemusí dávat smysl považovat za dluh částky menší než 100 Kč.
- Dále je třeba stanovit časový horizont (performance window), na kterém jsou dva zmíněné parametry sledovány.
- Za dobrého klienta lze např. označit klienta, který:
  - je po splatnosti méně než 60 dnů(s tolerancí 100 Kč) v prvních 6-ti měsících od první splátky,
  - je po splatnosti méně než 90 dnů (s tolerancí 30 Kč) v průběhu celé své platební historie (ever).

Default – definice cílové prom.

- Volba těchto parametrů závisí do značné míry na typu finančního produktu (jistě se bude lišit volba parametrů pro spotřebitelské úvěry pro malé částky se splatností kolem jednoho roku a pro hypotéky, které jsou obvykle spojeny s velmi vysokou finanční částkou a se splatností až několik desítek let) a na další využití této definice (řízení rizik, marketing, ...).

Default – definice cílové prom.

- Další praktickým problémem definice dobrého klienta je souběh několika smluv jednoho klienta. Například je možné, že zákazník je po lhůtě splatnosti na více smlouvách, ale s rozdílnými dny po splatnosti a s různými částkami. V tomto případě jsou většinou částky klienta dlužné v jednom konkrétním časovém okamžiku sečteny, a ze dnů po splatnosti na jednotlivých smlouvách je brána maximální hodnota. Tento přístup lze uplatnit pouze v některých případech, a to zejména v situaci, kdy jsou k dispozici kompletní účetní data. Situace je podstatně složitější v případě agregovaných údajů, např. na měsíční bázi.

Default – definice cílové prom.

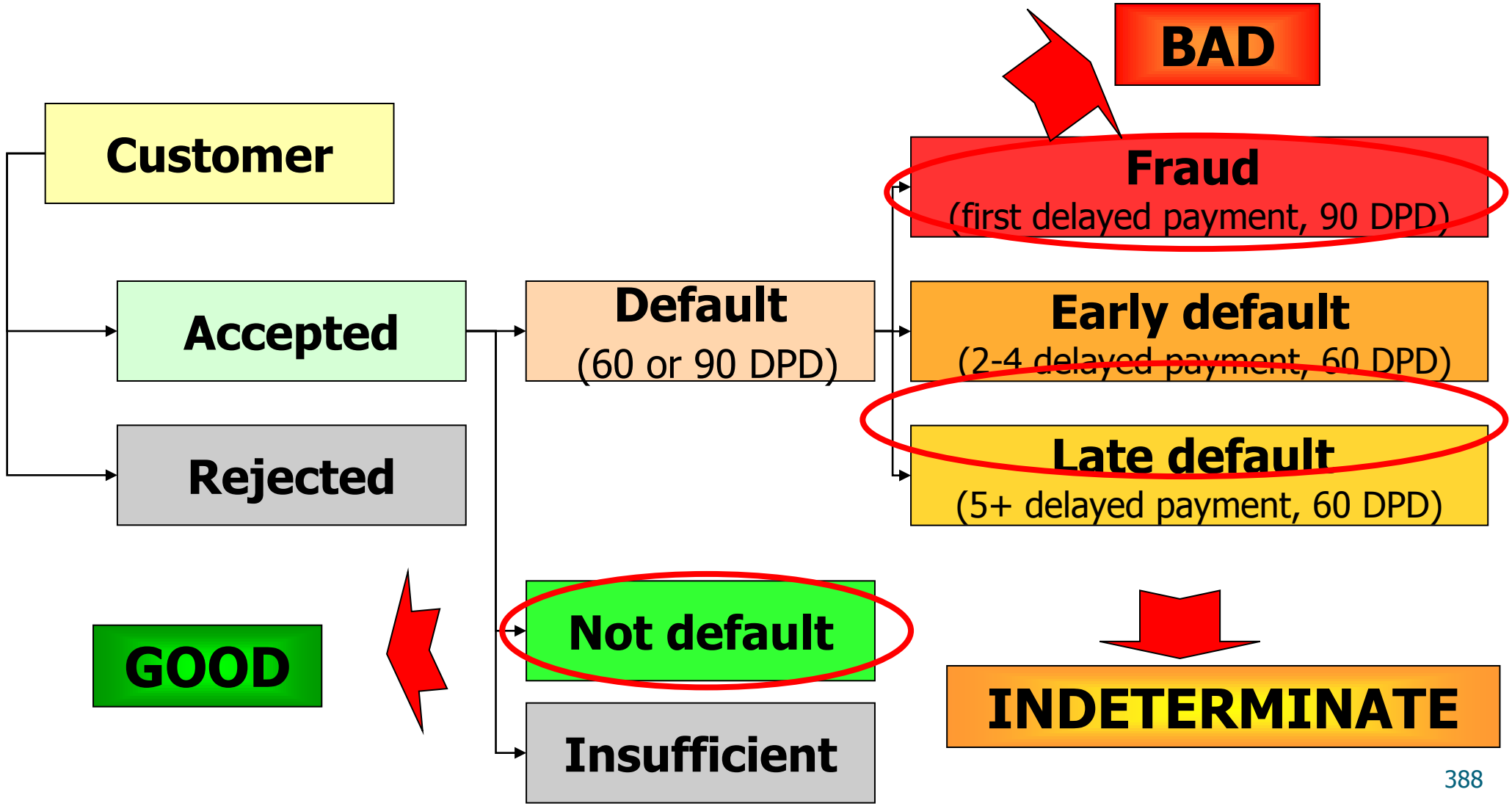
- Obecně uvažujeme následující typy klientů:

  - dobrý (good),
  - špatný (bad),
  - nedefinovaný (indeterminate),
  - s nedostatečnou úvěrovou historií (insufficient),
  - vyřazený (excluded),
  - zamítnutý (rejected).

Default – definice cílové prom.

- První dva typy byly diskutovány. Třetí typ, tj. indeterminate, je na hranici mezi dobrým a špatným klientem a při jeho použití přímo ovlivňuje definici dobrých/špatných klientů. Uvažujeme-li pouze DPD, klienti s vysokými DPD (např. 90 +) jsou typicky označeni za špatné, nedelikventní klienti (jejich DPD je rovno nule) jsou označeni za dobré. Za indeterminate jsou pak označeni delikventní klienti, kteří nepřekročí danou hranici DPD.
- Čtvrtý typ klientů jsou typicky klienti s velmi krátkou platební historií, u kterých je nemožná korektní definice cílové proměnné.
- Vyřazení klienti jsou klienti, jejichž data jsou natolik špatná, že by vedla ke zkreslení modelu(např. fraudy). Další skupinu tvoří klienti, kteří nejsou standardně hodnoceni daným modelem (VIP klienti)
- Poslední typ klientů jsou ti klienti, jejichž žádost o úvěr byla zamítnuta.

Definice dobrého/špatného klienta



Performance Definitions

- “Goods” and “bads” (and rejects) are used for model development.
- Indeterminates included for Gains chart and forecasting.

Segmentation

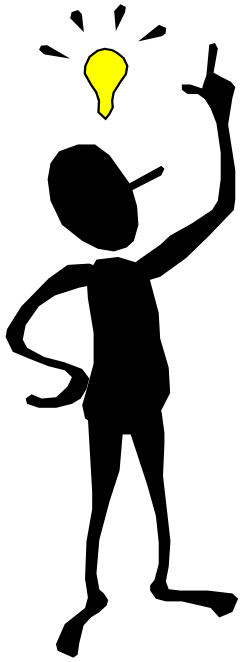
- Can one scorecard work efficiently for all the different populations within your portfolio?
- Or would more than one scorecard be better?

- Segmentation maximizes predictiveness for unique segments within your population.

Segmentation

- Experience (Heuristic)
  - Knowledge/experience, operational/industry based, common sense.
- Statistical
  - Let the data speak.
- "**Distinct** applicant/account sub-populations"
- "Better predictive power than single model".

Experience Based Segmentation



- Product
  - Card type, loan type (auto, home, unsecured), lease, used versus new, brand
- Demographics
  - Geographical (region, urban/rural, state/province, internal definition, neighborhood), age, time at bureau
- Source of business
  - Channel (net, branch, store-front, 'take one', brokers)
- Applicant type
  - new/existing, first time home buyer, groups (retired, students, engineers), thin/thick file, clean/dirty file
- Product Owned
  - Credit Card for existing mortgage/loan holders.

Experience Based Segmentation

- Consider future plans, not just historic operations

- How do we detect new segments?
  - Marketing/risk analysis:
    - Bad rates
    - Approval rate
    - Profit, and so on.
  - Look for significant performance difference.

Experience Based Segmentation

- Need to confirm experience using analytics.
- Definition of segments
  - What is a thin file?
  - What is ‘young’ versus ‘old’?
  - What is the best demographic split?
  - What break is best for ‘tenure at bank’?

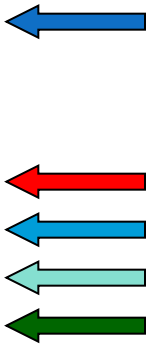
Confirming Experience

- Rule of thumb:
- "When the same information predicts differently across unique segments"

Bad Rate			
	Age > 30	Age < 30	Unseg
Res Status			
Rent	2.1%	4.8%	2.9%
Own	1.3%	1.8%	1.4%
Parents	3.8%	2.0%	3.2%
Trades			
0	5.0%	2.0%	4.0%
1-3	2.0%	3.4%	2.5%
4+	1.4%	5.8%	2.3%

Confirming Experience

Attributes	Bad Rates
Age	
Over 40 yrs	1.80%
30-40 yrs	2.50%
Under 30	6.90%
Source of business	
Internet	20%
Branch	3%
Broker	8%
Phone	14%
Applicant Type	
First Time buyer	5%
Renewal Mortgage	1%



That Is the Easy Way

- You can also build full segmented models, and compare "lift", sensitivity, and so on, with a base model.
- It is best to perform this analysis for both experience and statistically based segmentation.

Comparing Improvement

- Use different methods to measure improvement (lift, KS, c-stat, precision, and so on.)

Segment	Total c-stat	Seg c-stat	Improvement
Age < 30	0.65	0.69	6.15%
Age > 30	0.68	0.71	4.41%
Tenure < 2	0.67	0.72	7.46%
Tenure > 2	0.66	0.75	13.64%
Gold card	0.68	0.69	1.47%
Platinum card	0.67	0.68	1.49%

Comparing Improvement

- Portfolio stats will put improvements into measurable portfolio terms.

		After Segmentation		Before Segmentation	
Segment	**Size**	**Approve**	**Bad**	**Approve**	**Bad**
Total	100%	%	%	%	%
Age < 30	65%	%	%	%	%
Age > 30	35%	%	%	%	%
Tenure < 2	12%	%	%	%	%
Tenure > 2	88%	%	%	%	%
Gold card	23%	%	%	%	%
Platinum card	77%	%	%	%	%

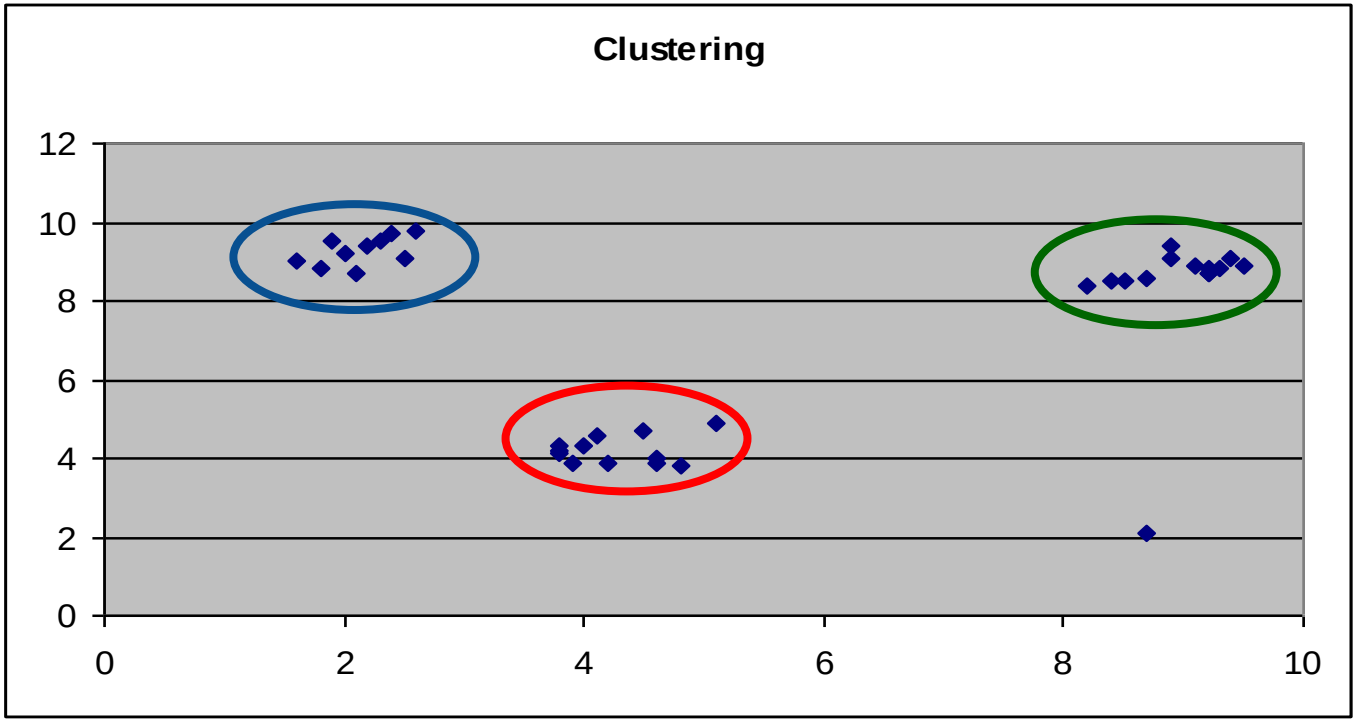
Choosing Segmentation

- Cost of scorecards (internal/external)
- Implementation
- Processing
- Data storage
- Monitoring/strategy development
- Segment size
- Do I have to?

Statistically Based Segmentation

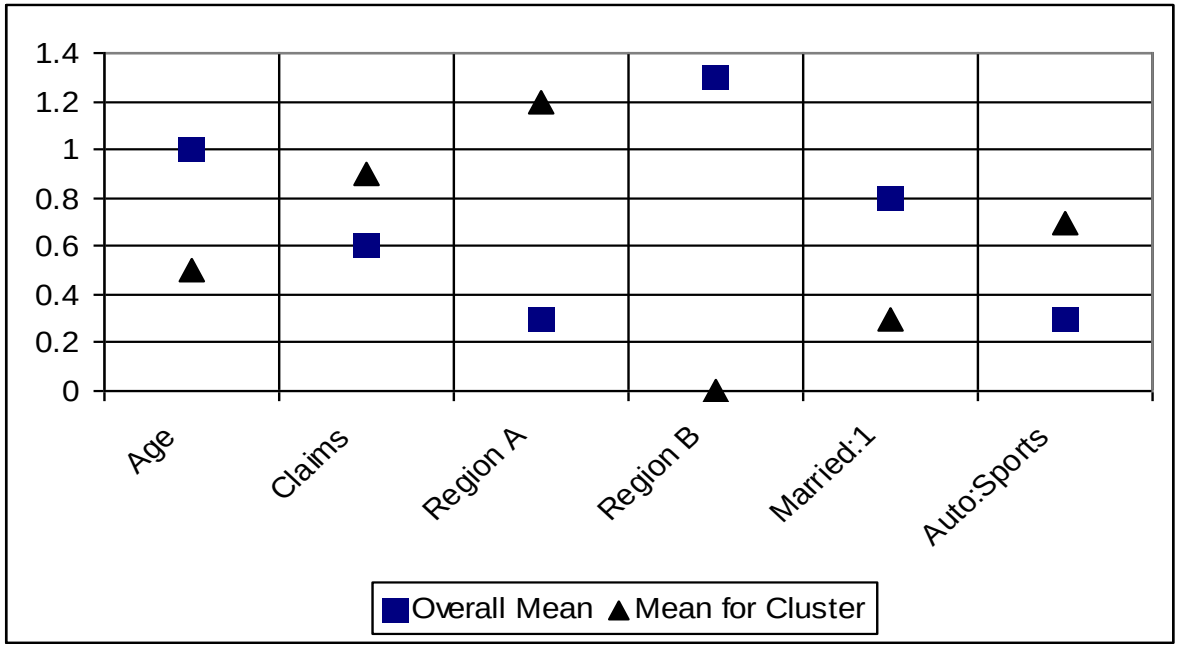
- Less preconceived notions
  - Clustering
  - Decision Trees.

Clustering



Showing 3 distinct groups and one outlier.

Clustering



Here is an insurance example of one cluster.

- What do we see here?

- lower than avg age
- more claims
- live in region A only
- likely to be single

and drive a sports car.

- this is obviously a high risk segment.
(confirm this group with claims analysis)

- Similar groups according to characteristics, not performance – so confirm performance for the clusters and combine those with similar risk behavior. We're not building a marketing profile, but a RISK PROFILE.

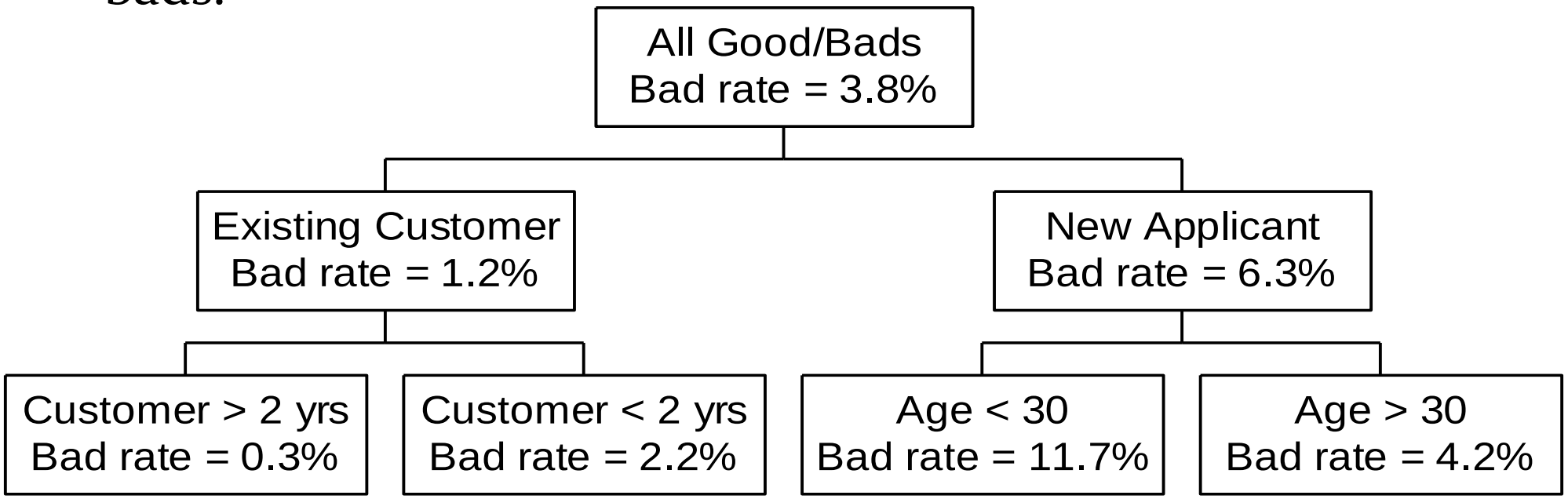
Clustering

- Defining characteristics for each group
- From previous example,
  - Young males region A
  - Young females region A, and so on.

- **Performance analysis to confirm segmentation**.

Decision Trees

- Isolates segments based on performance (target)
- Easily interpretable and differentiates between goods and bads.



So Now We Know …

- the business
- sample and performance windows
- “bad”, “good”, “indeterminate”
- exclusions
- bad rate, approval rate
- number of scorecards needed, and their segments.

Methodology/Format

- Implementation platform and format
- Interpretability, implementation
- Legal compliance
- Data quality, sample size, target type
- Tracking and diagnosis
- Specify parameters for scorecard (range of scores, “points to double the odds”).

Why ‘Scorecard’ Format?

- Easiest to interpret, justify, implement
- Reasons for decline/low scores can be explained to auditors, Mgmt, regulators, adjudicators
- No black box
- Diagnosis, tracking, monitoring
- Development process fairly simple to understand.

Review Implementation Plan

- Number of scorecards
- Data requirements
- Manage expectations
- Continuity.

Everyone is aware of what's going on.
This is a business process, not a mystery novel. You'd be surprised how many people in companies like to spring surprises on other departments.

Cvičení

Jsou k dispozici následující data:

Accepts.sas7bdat (64589 řádků)
Rejects.sas7bdat (35411 ř.)
Applicants.sas7bdat (100.000 ř.)

...24 sloupců
ID of applicant, Date of application/opening, Accept / Reject, 30-days deliquency, 30-days deliquency date, 60-days deliquency, 60-days deliquency date, 90-days deliquency, 90-days deliquency date, Worst previous deliquency, Current deliquency, Age, Age groups, Sex, Existing client?, Phone member?, Region, Income, Income groups, Debt, Income/Debt ratio, Income/Debt ratio groups, Probability of 60-days deliquency (old), Score (old).

Základní popis dat:

```
title 'Accepts';
proc means data=indata.accepts n nmiss min median mean
max;
  var age income debt idratio;
run;

title 'Accepts';
proc freq data=indata.accepts;
table sex client phone region;
table (sex client phone region)*bad60;
table bad30*(bad60 bad90) bad60*bad90;
run;

title 'All applicants';
goptions ftext='arial';
proc catalog c=gseg kill;
quit;
proc gchart data=indata.applicants;
vbar age / midpoints=18 to 75 name='_1data_a';
vbar income / name='_1data_b';
vbar debt / name='_1data_c';
vbar idratio / name='_1data_d';
vbar type / name='_1data_e';
vbar scoreold / levels=10 name='_1data_f';
vbar pbad60old / levels=30 name='_1data_f';
run;
quit;

proc univariate data=indata.applicants normal;
var age income debt idratio;
histogram age income debt idratio;
run;
```

Cvičení

Vybrané výstupy uvedeného kódu:

The FREQ Procedure

Sex	Frequency	Percent	Cumulative Frequency	Cumulative Percent
M	45138	69.88	45138	69.88
Z	19451	30.12	64589	100.00

Sex

Client	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	60188	93.19	60188	93.19
1	4401	6.81	64589	100.00

Existing client?

Phone	Frequency	Percent	Cumulative Frequency	Cumulative Percent
0	8081	12.51	8081	12.51
1	56508	87.49	64589	100.00

Phone member?

Region	Frequency	Percent	Cumulative Frequency	Cumulative Percent
1	12537	19.41	12537	19.41
2	16335	25.29	28872	44.70
3	10679	16.53	39551	61.23
4	10797	16.72	50348	77.95
5	7199	11.15	57547	89.10
6	7042	10.90	64589	100.00

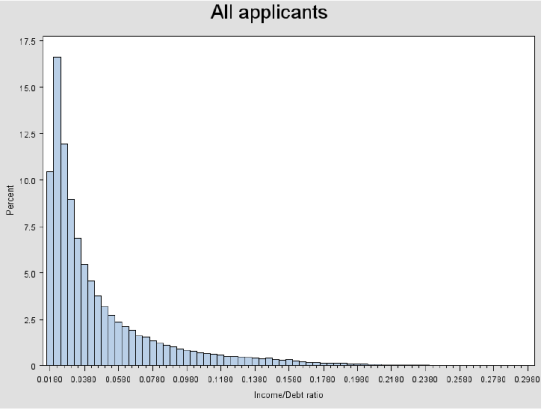
Region

Frequency
Percent
Row Pct
Col Pct

Table of Phone by Bad60

Phone(Phone member?)	Bad60(60-days deliquency) 0	1	Total
0	7805 12.08 96.58 12.47	276 0.43 3.42 13.97	8081 12.51
1	54809 84.86 96.99 87.53	1699 2.63 3.01 86.03	56508 87.49
Total	62614 96.94	1975 3.06	64589 100.00

The UNIVARIATE Procedure



Accepts

The MEANS Procedure

Variable	Label	N	N Miss	Minimum	Median	Mean	Maximum
Age	Age	64589	0	18.0000000	43.0000000	43.3129945	74.0000000
Income	Income	64589	0	15000.07	19631.47	19854.56	35790.94
Debt	Debt	64589	0	100444.85	560744.83	576945.05	1611457.12
IDRatio	Income/Debt ratio	64589	0	0.0175377	0.0345483	0.0500680	0.2994807

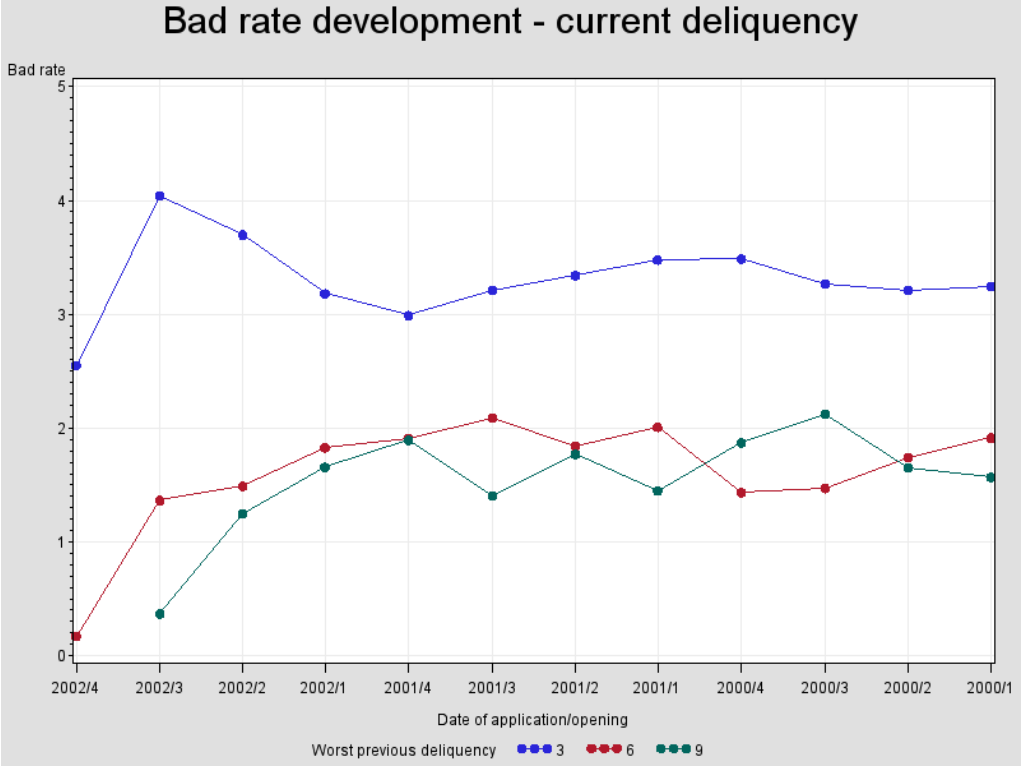
The UNIVARIATE Procedure
Variable: IDRatio (Income/Debt ratio)

Moments			
N	100000	Sum Weights	100000
Mean	0.04766914	Sum Observations	4766.91379
Std Deviation	0.03680037	Variance	0.00135427
Skewness	2.10159641	Kurtosis	4.8128053
Uncorrected SS	362.660032	Corrected SS	135.425362
Coeff Variation	77.1995691	Std Error Mean	0.00011637

Basic Statistical Measures			
Location		Variability	
Mean	0.047669	Std Deviation	0.03680
Median	0.033093	Variance	0.00135
Mode	.	Range	0.28194
		Interquartile Range	0.03334

Cvičení

```
/* 2a. Bad rate development, roll rate analysis */


%let performancewindow='31dec2002'd>=datappl;
%let deliq=worstdeliq;

proc freq data=indata.accepts /*noprint*/;
table datappl*&deliq / out=&deliq (keep=datappl &deliq pct_row
where=(&deliq ne '0')) outpct missing;
format datappl yyqs7.;
where &performancewindow;
run;

ods html path="&appl_root" file="2.&deliq..html";
goptions reset=all ftext='arial';
symbol1 i=j v=dot;
axis1 label=('Bad rate');
proc catalog c=gseg kill;
quit;
title 'Bad rate development - current deliquency';
proc gplot data=&deliq;
plot pct_row*datappl=&deliq / name='_2curdel' grid hreverse
vaxis=axis1 hminor=0;
run;
quit;
ods html close;
```



Cvičení

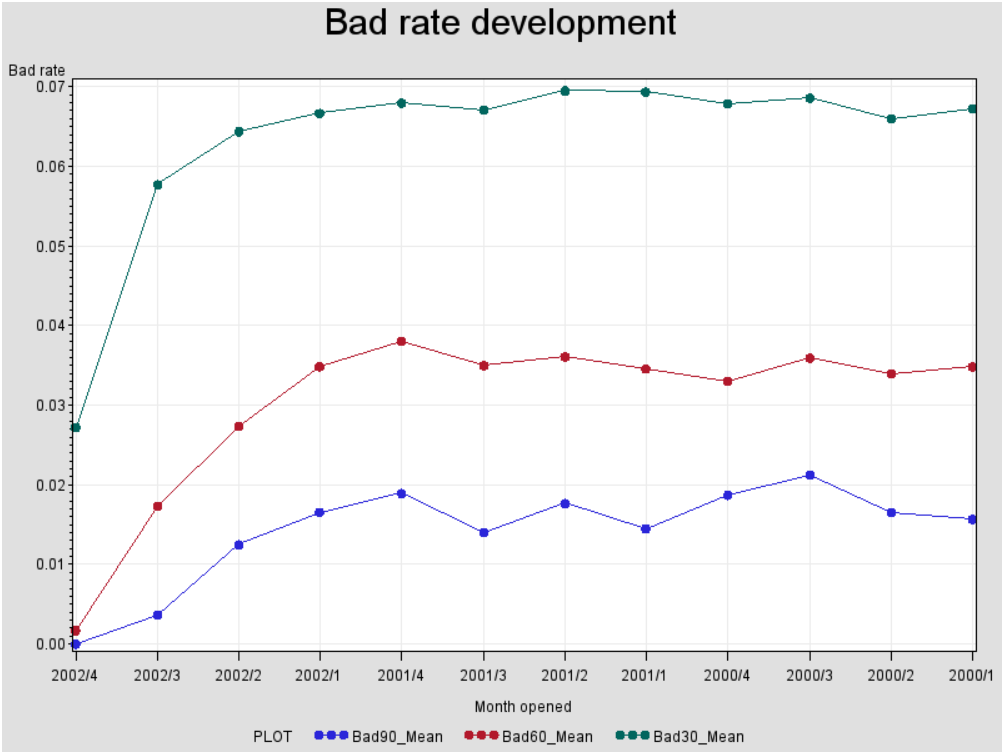
```
/* analyza kohort */

%let target=bad30;
%let date=dat30;

data cohorts;
set indata.accepts (keep=datappl bad: dat:);
if &target then qtr=int(yrdif(datappl,&date,'act/act')*4)+1;
datappl=intnx('month',datappl,0);
format datappl mmyys7.;
run;
proc freq data=cohorts noprint;
table datappl / out=cohorts1 (drop=percent
rename=(count=counttotal));
table datappl*qtr / out=cohorts (drop=percent);
run;
data cohorts;
merge cohorts cohorts1;
by datappl;
if first.datappl then cumpct=.;
if qtr ne . then do;
    cumpct+(count/counttotal);
    output;
end;
run;

ods html path="&appl_root" file='2.cohorts.html';
title "Cohort analysis for &target";
proc tabulate data=cohorts missing format=percent8.4;
class datappl qtr;
var cumpct;
table datappl,qtr*cumpct="*sum=";
run;
ods html close;
```

Cohort analysis for bad30

	qtr		
	1	2	3
Date of application/opening			
01/2000	5.987%	6.652%	.
02/2000	6.327%	6.887%	7.055%
03/2000	5.491%	6.441%	6.494%
04/2000	5.458%	6.254%	6.367%
05/2000	6.600%	7.648%	.
06/2000	5.219%	5.724%	.
07/2000	5.437%	6.130%	.
08/2000	6.321%	7.401%	7.455%
09/2000	6.345%	7.019%	.
10/2000	6.023%	6.782%	.
11/2000	5.613%	6.104%	6.213%
12/2000	6.400%	7.385%	.
01/2001	6.064%	7.109%	.
02/2001	5.836%	6.809%	.
03/2001	6.271%	6.766%	6.876%
04/2001	6.935%	7.870%	7.925%
05/2001	6.201%	6.884%	.
06/2001	5.391%	5.996%	6.051%
07/2001	4.859%	5.729%	.
08/2001	6.339%	7.377%	.
09/2001	6.378%	7.118%	.
10/2001	6.448%	7.200%	.
11/2001	5.956%	6.415%	.
12/2001	5.521%	6.704%	6.761%
01/2002	5.668%	6.812%	.
02/2002	5.913%	6.684%	6.812%
03/2002	5.924%	6.408%	.
04/2002	5.776%	6.436%	.
05/2002	5.304%	5.974%	.
06/2002	6.079%	6.900%	.
07/2002	5.374%	5.817%	.
08/2002	5.405%	5.622%	.
09/2002	5.493%	5.889%	.
10/2002	4.608%	.	.
11/2002	2.563%	.	.

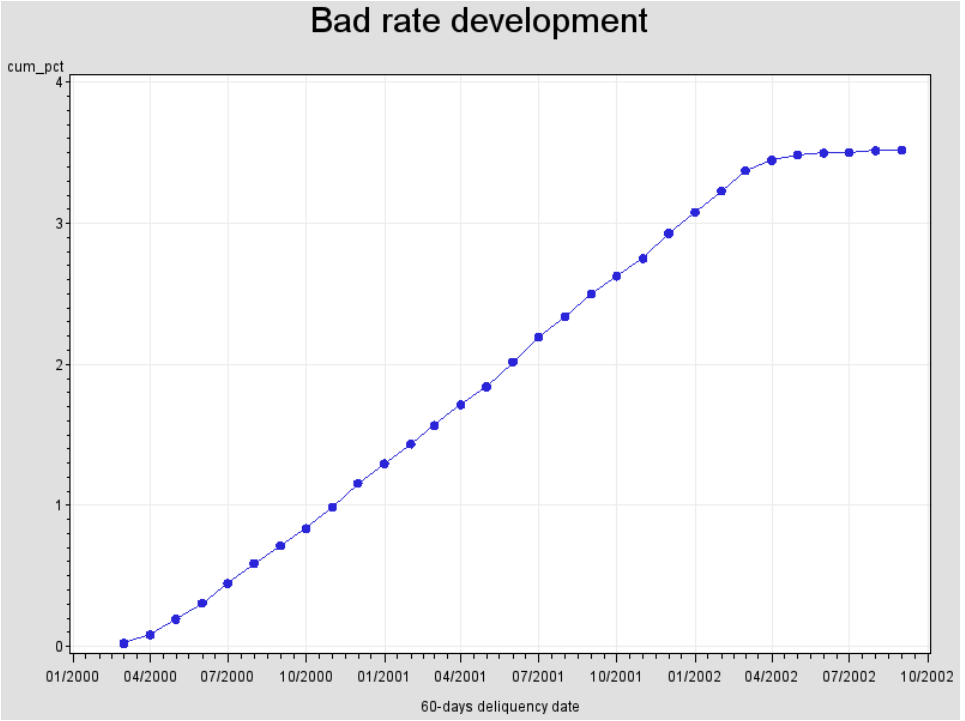
Cvičení

```
/* performance window */

%let performancewindow='31dec2002'd>=datappl;

proc tabulate data=indata.accepts out=brdev
(drop=_type_ _table_ _page_);
class datappl;
var bad90 bad60 bad30;
table datappl,(bad90 bad60 bad30)*mean*format=percent8.2;
format datappl yyqs7.;
where &performancewindow;
label datappl='Month opened';
run;

ods html path="&appl_root" file='2.perf.html';
goptions reset=all ftext='arial';
symbol1 i=j v=dot;
axis1 label=('Bad rate');
proc catalog c=gseg kill;
quit;
title 'Bad rate development';
proc gplot data=brdev;
plot (bad:)*datappl / name='_2perf' grid overlay legend hreverse
vaxis=axis1 hminor=0;
run;
quit;
ods html close;
```



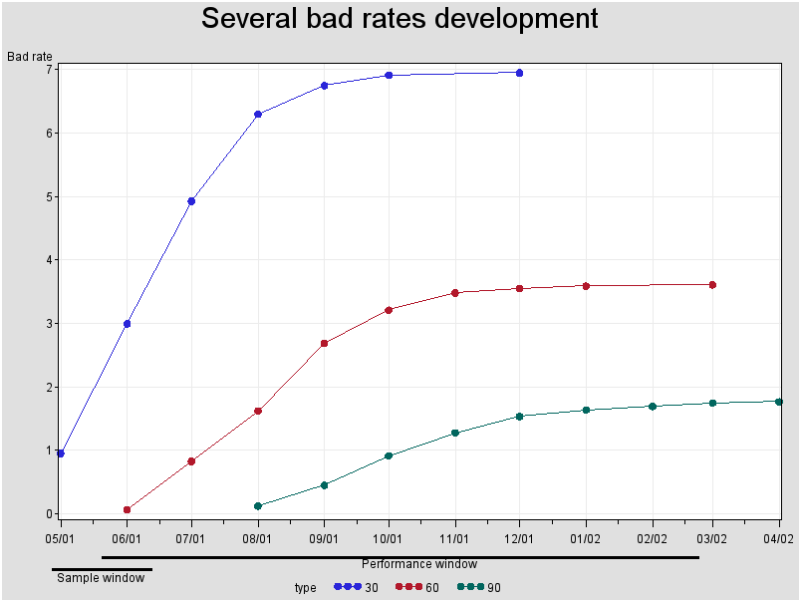
Cvičení

```
/* bad rate development */

%let samplewindow='30jun2001'd>=datappl>='01apr2001'd;
%let samplewindow='31dec2001'd>=datappl;

proc freq data=indata.accepts noprint;
table dat60 / out=development missing;
format dat60 mmyys7.;
where &samplewindow;
run;
data development;
set development;
if _n_>1 then do;
    dat60=intnx('month',dat60,0);
    cum_pct+percent;
    output;
end;
label datappl='Month of opening';
run;

ods html path="&appl_root" file='2.badratedev.html';
goptions reset=all ftext='arial';
symbol1 i=j v=dot;
axis1 label=('Bad rate');
proc catalog c=gseg kill;
quit;
title 'Bad rate development';
proc gplot data=development;
plot cum_pct*dat60 / name='_2brd' grid;
run;
quit;
ods html close;
```



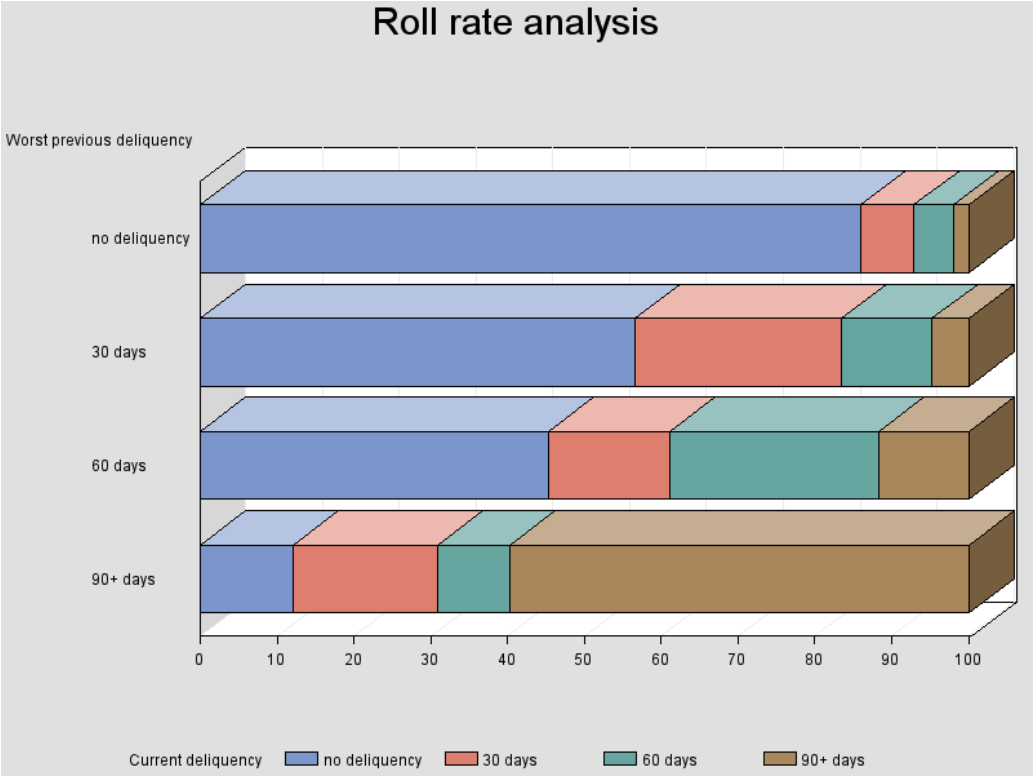
Cvičení

```
/* BRDEV macro */

%macro brdev(data,out,datevar,targetvar,samplewindow);
proc freq data=&data noprint;
table &datevar / out=&out missing;
format &datevar mmyys7.;
where &samplewindow;
run;
data &out (keep=date cum_pct);
set &out;
if _n_>1 then do;
    date=intnx('month',&datevar,0);
    cum_pct+percent;
    output;
end;
format date mmyys7.;
run;
%mend brdev;

%let samplewindow='30jun2001'd>=datappl>='01apr2001'd;
%brdev(indata.accepts,development,dat60,bad60,&samplewindow)

/* several bad rate development */

%let samplewindow='30jun2001'd>=datappl>='01apr2001'd;
%brdev(indata.accepts,development30,dat30,bad30,&samplewindow)
%brdev(indata.accepts,development60,dat60,bad60,&samplewindow)
%brdev(indata.accepts,development90,dat90,bad90,&samplewindow)

data developmentsev;
set development30 (in=__30) development60 (in=__60) development90;
if __30 then type='30';
else if __60 then type='60';
else type='90';
Run;

data anno;
function='label';x=20;y=2;text='Sample window';output;
size=2;function='move';x=10;y=2.5;output;
function='draw';x=30;y=2.5;output;
function='move';x=20;y=3.5;output;
function='draw';x=140;y=3.5;output;
run;

ods html path="&appl_root" file='2.badratedev_several.html';
goptions reset=all ftext='arial';
symbol1 i=j v=dot;
axis1 label=('Bad rate');
proc catalog c=gseg kill;
quit;
title 'Several bad rates development';
proc gplot data=developmentsev annotate=anno;
plot cum_pct*date=type / grid vminor=0 name='_2brds' vaxis=axis1;
format date mmyys5.;
label date='Performance window';
run;
quit;
ods html close;
```

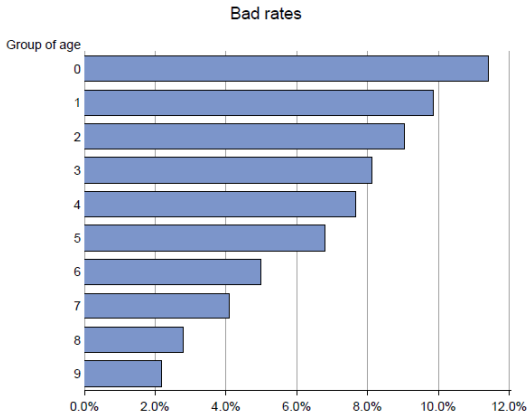


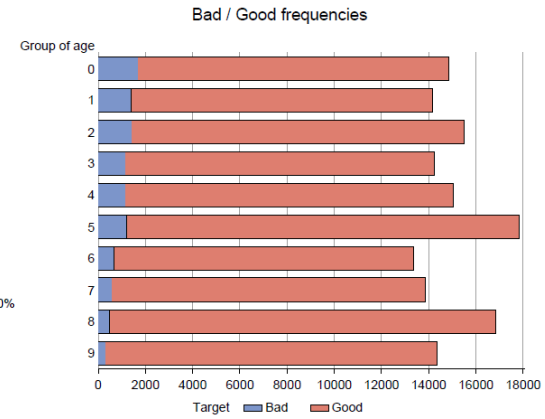
Cvičení

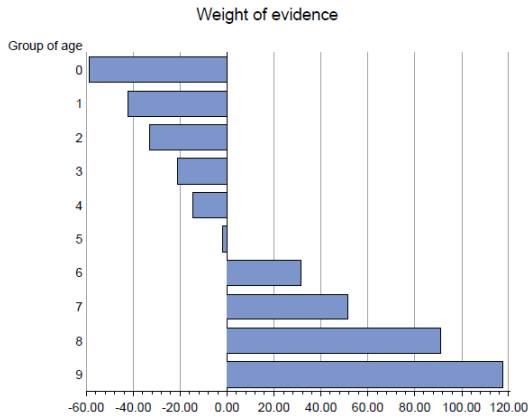
```
/* Roll rate analysis */

ods html path="&appl_root" file='2.roll_rate.html';

proc format;
value $deliq (notsorted)
    '0'=' no deliquency'
    '3'='30 days'
    '6'='60 days'
    '9'='90+ days';
run;

proc tabulate data=indata.accepts out=rollrate missing;
class curdeliq worstdeliq;
tables worstdeliq,curdeliq*rowpctn;
format curdeliq $deliq. worstdeliq $deliq.;
title 'Roll rate analysis';
run;
proc gchart data=rollrate;
hbar3d worstdeliq / sumvar=pctn_01 subgroup=curdeliq nostats
clipref autoref raxis=axis1;
axis1 label=none minor=none;
run;
quit;

ods html close;
```



6. Příprava dat II







Development

Stage 3: Development Database Creation

Development Sample Specification

- Development sample spec. means specifying what we need in the database we will use for development. We are not going to take a dump of everything from the CDW or datamart.
- Make the development process manageable and efficient:
  - list of characteristics (or "variables" to be considered for devp. You don't want to have the entire DW.)
  - sample sizes (for each segment and category. No point regressing on 100k when 3k will suffice.)
  - parameters from previous section.
- Do all this bearing in mind the number of scorecards you want developed and for which segments.

Characteristic Selection

How do you select characteristics? Reinforce: there is a need for some thought to be put into process in selecting characteristics ..
You get together with risk, mktg, product. And get operations areas such as collections aboard (WHO knows your bad guys better than anyone else?)

- Expected predictive power
- Reliability: (is this manipulated? or prone to be manipulated?, e.g. salary. Check historical data - cannot be confirmed or too expensive to confirm. Can it be interpreted e.g. occupation/industry type is the worst cases. Do poeple usually leave this blank.)
  - manipulation (non-confirmable)
  - interpretation (present and future)
  - missing
- Legal issues (Cant ask/get some info?.. Might get into trouble with some?)

Characteristic Selection

- Ease in collection
  - Do you want to spend time chasing missing info for a credit card?... may be OK for a mortgage. How easy it is to get this piece of info?
- Policy rules
  - Don't include anything that is unchangeable PR, e.g. bankruptcy. If you are going to decline all bankrupcy, no need to use it in scorecard.
- Derived variables – ratios
  - Can do a lot of ratios .. But put some business thought into it.
- Future direction.
  - Will this info be collected in the future (e.g. app form redesign)?
  - Industry direction - not relevant today but will change. can include in card or collect for future e.g. higher credit lines. Talk to credit bureaus industry trend and how they affect the scorecard.

What are you doing: you're looking at objectives, company operations, business knowledge, ground realities etc.

This is not just a stats exercise!!!

Sampling

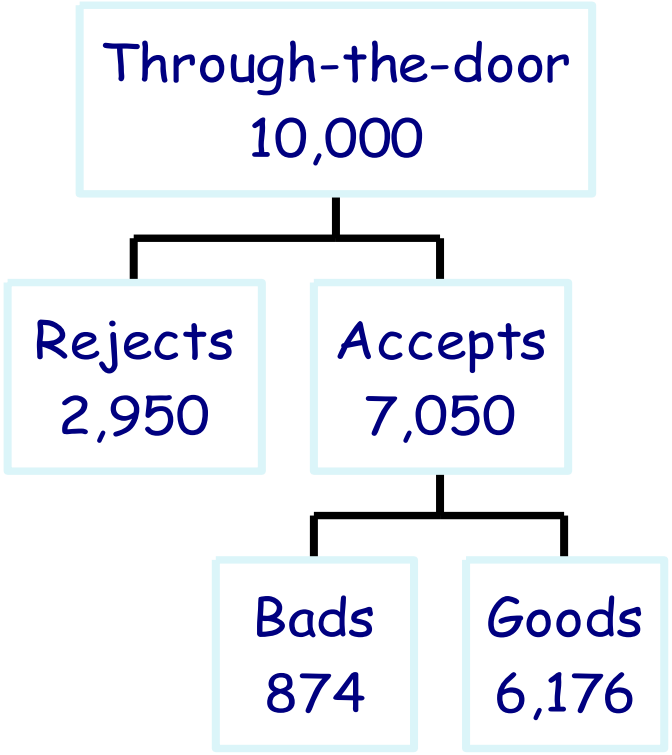
- Development, validation
  - 70:30, 80:20
  - If sample is small, do 100%, but validate with several 50–80%.
- Good, bad, reject
  - 2000 of each (or higher)
  - Oversampling (oversampling is common when modeling rare events … it leads to better predictions)
  - Proportional sample – not recommended for low bad rates.
  - Take what you got for bads and sample the goods.
- Ensure that each group has sufficient numbers for meaningful analysis.

Data Collection and Database Construction

- Random and representative
  - for each segment applicants (and accounts)
- One for unsegmented (to measure lift from segmentation)
- Data quirks, changes (preferably documented)
  - e.g. code for renters changed from R to E .. Stopped collecting some data item, new data fields, started collecting data recently etc. etc.
- Objective: Data collected, as specified.

Adjusting for Prior Probabilities

- When oversampling
- Adjust to actual:
  - Approval rate
  - Bad rate
- Analysis and reports reflect reality
- Do not need if you only want to know relationships or rank ordering.



Adjusting for Oversampling

- Separate sampling is standard practice (helps when you just did ‘bad’ definition)
- Prior probabilities must be known
- Can adjust before fitting the model or after.

- Two ways:
  - Offset
  - Sampling weights (frequency variable).

Offset Method

- Logit $(p_i)=\beta_o+ \beta_1 x_1+ ....+ \beta_k x_k$
- When oversampling, logits shifted by the *offset*:
- Logit $(p^*_i)= ln\ (\rho_1 \pi_o / \rho_o \pi_1)\ + \beta_o+ \beta_1 x_1+ ....+ \beta_k x_k$
- Where
  - ρ_1 and ρ_o= proportion of target classes in the sample
  - π_1 and π_o= proportion of target classes in the population.

Offset Method

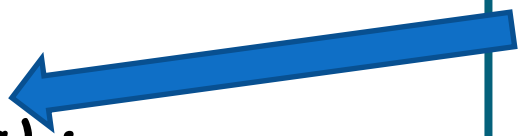
- Adjustment post-model (after model development):

- $p\hat{}_i = (p\hat{}^*_i \rho_0 \pi_1) \; / \; [(1 - p\hat{}^*_i)\, \rho_1 \pi_0 + p\hat{}^*_i \rho_0 \pi_1)]$

- Where $p\hat{}^*_i$ is the unadjusted estimate of posterior probability.

SAS Programs – Pre-model Adjustment

```
data develop;
set develop;
   off=(offset calc);
run;

proc logistic data=develop ...;
model ins=……./ offset=off;
run;

proc score ….;
   p=1 / (1+exp(-ins));
proc print;
var p ….;
run;
```



$ln\ (\rho_1\pi_o\ /\ \rho_o\pi_1)$

SAS Program – Post-model Adjustment

```
proc logistic data=develop...;
run;

proc score ... out=scored...;
run;

data scored;
set scored;
   off = (offset calc);
   p=1 / (1+exp(-(ins-off)));
run;

proc print data=scored ..;
var p ...;
run;
```

Sampling Weights

- Adjusts data to reflect true population
  - Weights: π_1/ρ_1 and π_0/ρ_0
  - Or set weight of bad=1 and weight of good = p(good)/p(bad) for population.
    - For example, p(bad)=4%, 2000 goods, 2000 bads. Sample will show 2000 bads and 48,000 goods.
  - Normalization causes less distortion in p values and standard errors.
  - Use FREQ variable in EM or calculate sample weight and use weight=sampwt in the LOGISTIC procedure.

SAS Program

- When using the WEIGHT statement, some output is not correct.

```
data develop;
set develop;
sampwt=( π0/ ρ0)* (ins=0) +
( π1/ ρ1)* (ins=1);
run;
proc logistic data=develop …;
weight=sampwt;
model ins=…….;
run;
```

What Is the Difference?

- The parameter estimates will be different.
- When linear-logistic model is correctly specified, offset is better.
- When logistic model is an approximation of some non-linear model, weights are better.
- For scorecards, weighting is better since it corrects the parameter estimates used to derive scores (prior probabilities only affect the predicted probabilities).

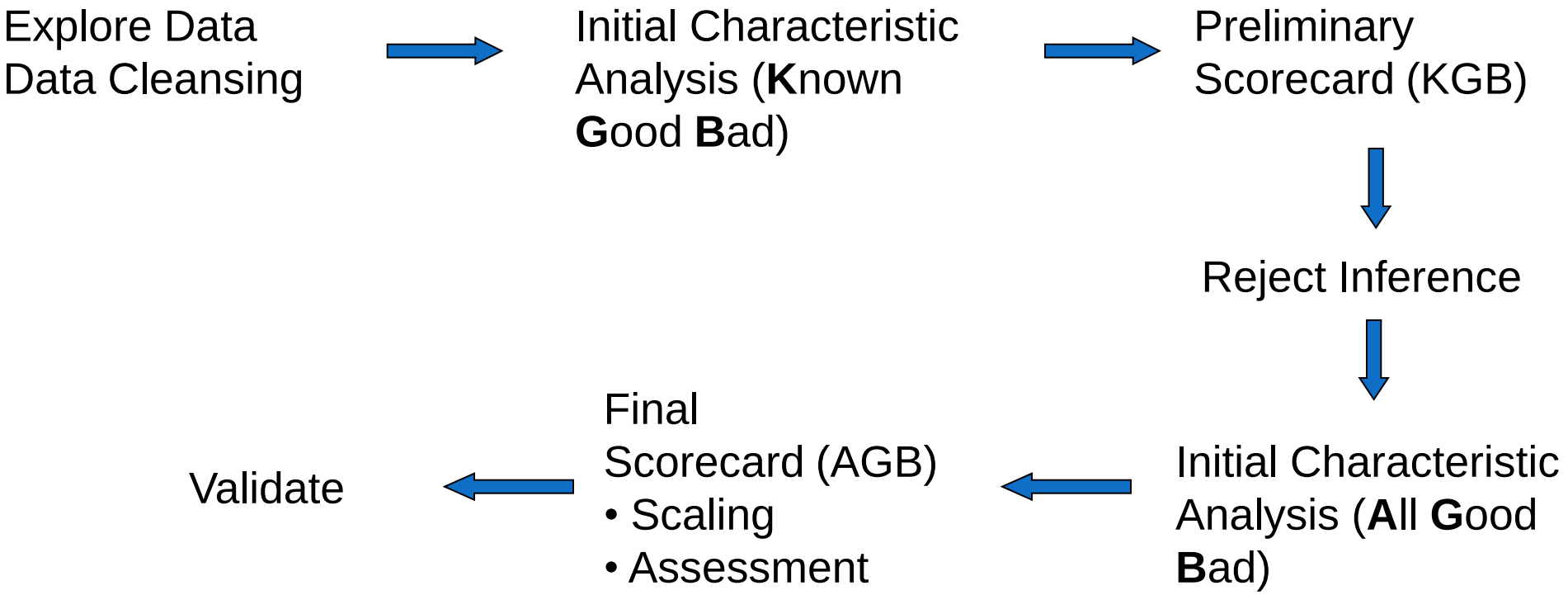
Development

Stage 4: Scorecard Development

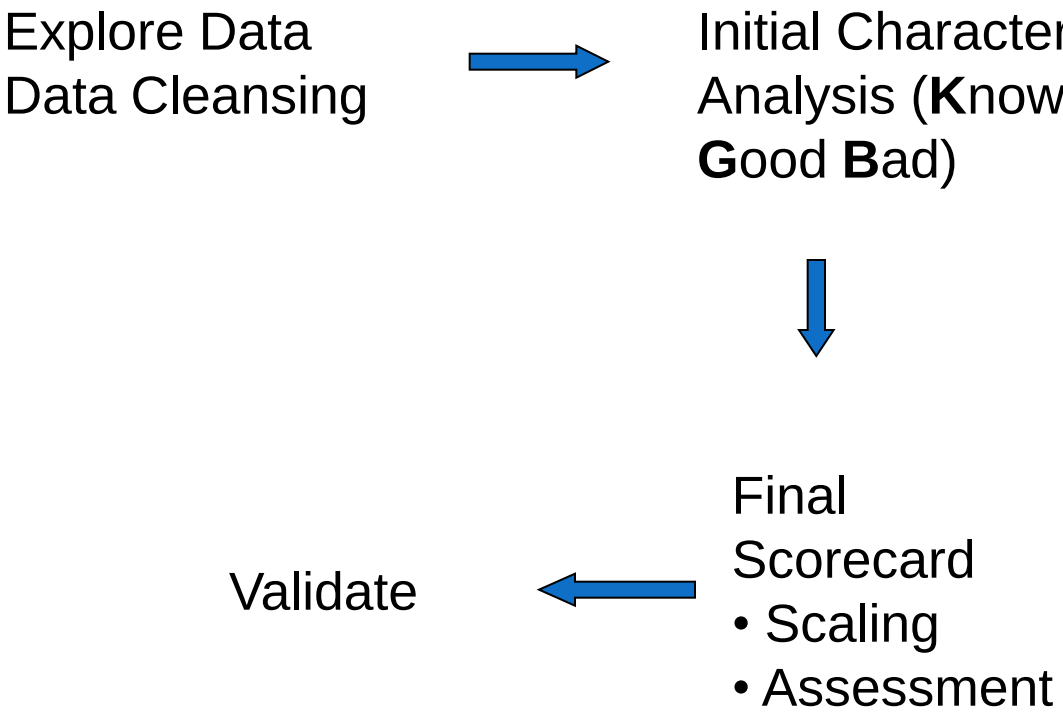
Objective

- Understand a methodology for developing and assessing risk scorecards.
  - Grouped attributes
  - Logistic regression
  - Reject inference
  - Scaled points.

Process Flow – Application Scorecard



Process Flow – Behavior Scorecard



Before you start …

- Explore the data, visualize (Insight in SAS EM)
- Distributions
  - mean, max/min, range, missing
- Compare with overall portfolio distributions
- Data integrity (any garbage, outliers)
- Ensure data meets the data specifications done earlier.
- Check that ‘0’s mean zero, not missing values.
- Population stability check:
  - Month by month table of distribution for each predictor (e.g. 200701 men 55%, women 45%, 200702 men 57%, women 43%)

Missing Values and Outliers

- Missing (ALL financial data has missing and garbage values)
  - Complete Case Analysis - Exclude everything with missing data .. In CS, you'll end up with nothing ☹.
  - Exclude characteristics or records with significant missing values
  - Group 'missing' as a distinct attribute -the weight of missing will tell you what missing contains. If it is close to neutral, good since it is random. Recommended – recognize that missing data has information value and may not be randomly missing. Find the value and use it. Plus, including missing 'points' in scorecard will take care of ppl who leave it blank.
  - Impute missing values – don't use mean/most likely, model based on decision tree may be better.
- Outliers (and mis-keys)
  - Exclude/replace records.

Missing Values

- Missing data is not usually random
- Missing data can be related to the target
  - New at job may leave yrs at empl blank
  - Low income or commercial customers leave income blank
- Do bad customers leave certain fields blank?
- Including and grouping missing data can answer this question.

Initial Characteristic Analysis

- Analyze individual characteristics
  - Identify strong characteristics
    - Best differentiators between ‘good’ and ‘bad’
    - Screening
- Select characteristics for regression (variable selection).

Initial Characteristic Analysis

- Start by performing initial grouping for each characteristic and rank order Information Value (PROC DMSPLIT or SPLIT, or EM node)
- Alternate: rank order characteristics by Chi Square or other method
- Fine tune grouping for stronger characteristics

- May want to perform other analysis prior to this (for example, use PC to identify collinear characteristics)
- Some people use principal components (PROC VARCLUS) to identify which characteristics they need from each cluster. And then concentrate on the best out of each.

Criteria for Variable Selection

- Predictive power of attribute:
  Weight of Evidence
- Range and trend of WOE across attributes
- Predictive power of characteristic:
  Information Value, Gini index(coefficient)
- Operational/business considerations.

Weight of Evidence

Age	Count	Distr Count	Goods	Distr Good	Bads	Distr Bad	Bad rate	Weight
Missing	50	3.00%	43	2.40%	8	4.10%	16%	-55.497
18-22	200	10.00%	152	8.40%	48	24.90%	24%	-108.405
23-26	300	15.00%	246	13.60%	54	28.00%	18%	-72.039
27-29	450	23.00%	405	22.40%	45	23.30%	10%	-3.951
30-35	500	25.00%	475	26.30%	25	13.00%	5%	70.771
35-44	350	18.00%	349	19.30%	11	5.70%	3%	122.044
44 +	150	8.00%	147	8.10%	3	1.60%	2%	165.509
Total	**2,000**		**1,807**		**193**		**9.65%**	
Information Value = 0.066								

$$\text{Ln}\left[\frac{\text{Distr Good}}{\text{Distr Bad}}\right] \times 100$$

Weight of Evidence

- Measures strength of each (grouped) attribute in separating goods and bads
- (Distr Good / Distr Bad) = odds of being good
- Negative weight: more bads than goods
- Logical trend
- For age 23-26:

WOE = ln (0.136 / 0.28) = -0.722 *(x 100 = -72.2)*

Information Value (Strength)

Age	Count	Distr Count	Goods	Distr Good	Bads	Distr Bad	Bad rate	Weight
Missing	50	3.00%	43	2.40%	8	4.10%	16%	-55.497
18-22	200	10.00%	152	8.40%	48	24.90%	24%	-108.405
23-26	300	15.00%	246	13.60%	54	28.00%	18%	-72.039
27-29	450	23.00%	405	22.40%	45	23.30%	10%	-3.951
30-35	500	25.00%	475	26.30%	25	13.00%	5%	70.771
35-44	350	18.00%	349	19.30%	11	5.70%	3%	122.044
44 +	150	8.00%	147	8.10%	3	1.60%	2%	165.509
Total	**2,000**		**1,807**		**193**		**9.65%**	
Information Value = 0.066								

$$\sum \left\{ \left[\text{Distr Good} - \text{Distr Bad} \right] \times \text{Weight} \right\}$$

Kullback, S., Information Theory and Statistics (1959)

Information Value

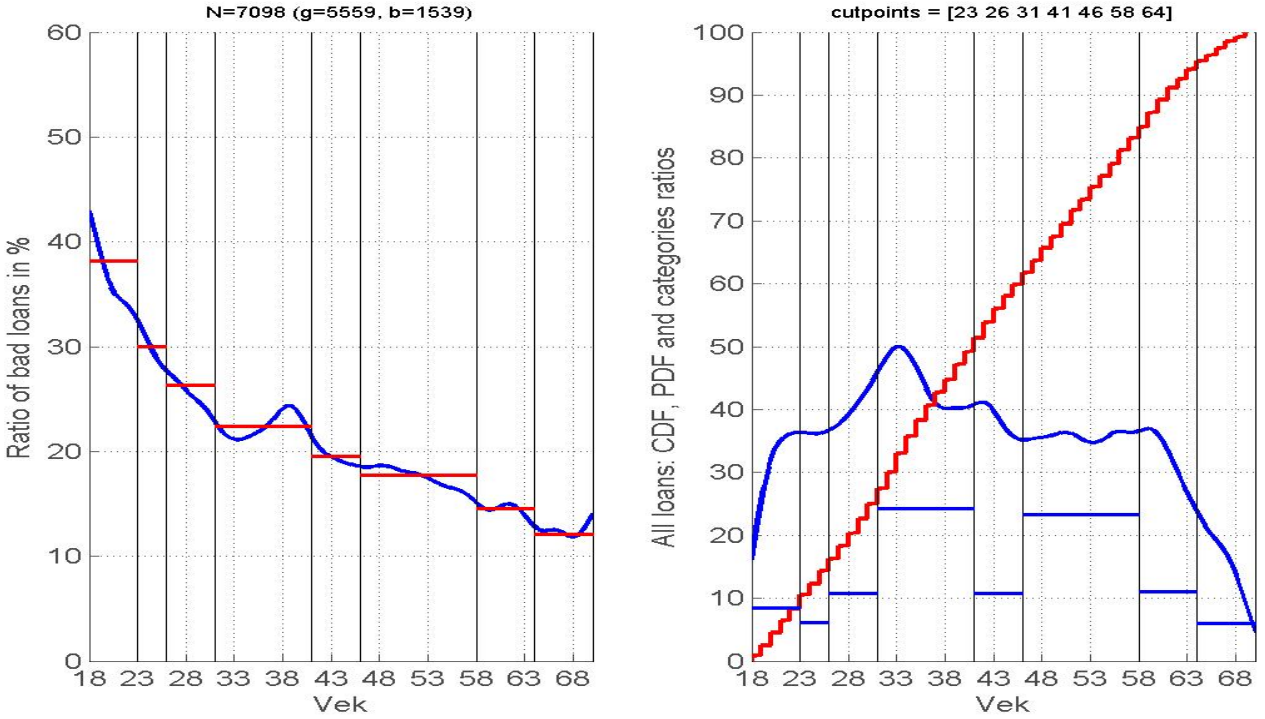
- Σ [(Distr Good - Distr Bad) x {ln (Distr Good / Distr Bad)}]
- When figures used in decimals format (for example, 0.136).
- Rule of thumb:
  - < 0.02: unpredictive
  - 0.02 – 0.1: weak
  - 0.1 – 0.3: medium
  - 0.3 +: strong
- Too strong? (IV>0.5) – use it in a controlled way (add them in the end of regression to see if they add any incremental value)

Grouping

- Groups with similar WOE are put together
- For continuous variables, groups are created so as to maximize difference from one group to next – and maintain logical trend for WOE

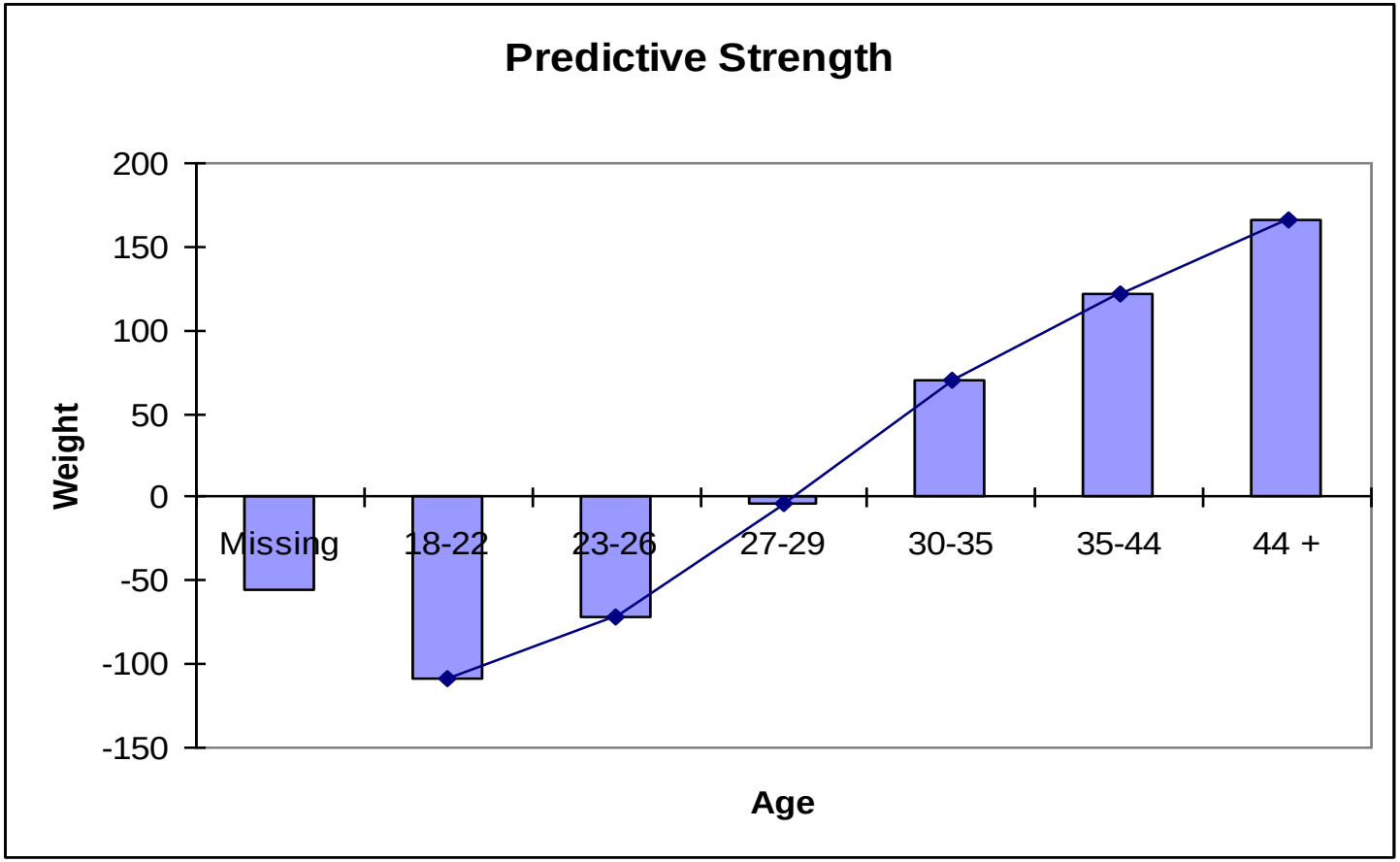
- Why Group?
  - Easier way to deal with outliers with interval variables, and for rare classes
  - Format of the scorecard
  - Easy to understand relationships
  - Model non-linear dependencies with linear models
  - Control the process

Grouping



Grouping of the demographic scorecard variable "age". On the left pictures, the dependence of bad rate (smoothed using normal probability density function) on the variables is presented. On the right, the cumulative distribution function is presented. Vertical lines represent the borders between categories, horizontal red lines in the left picture represent the mean bad rate in categories, horizontal blue lines in the right picture represent the relative distribution of observations in the categories.

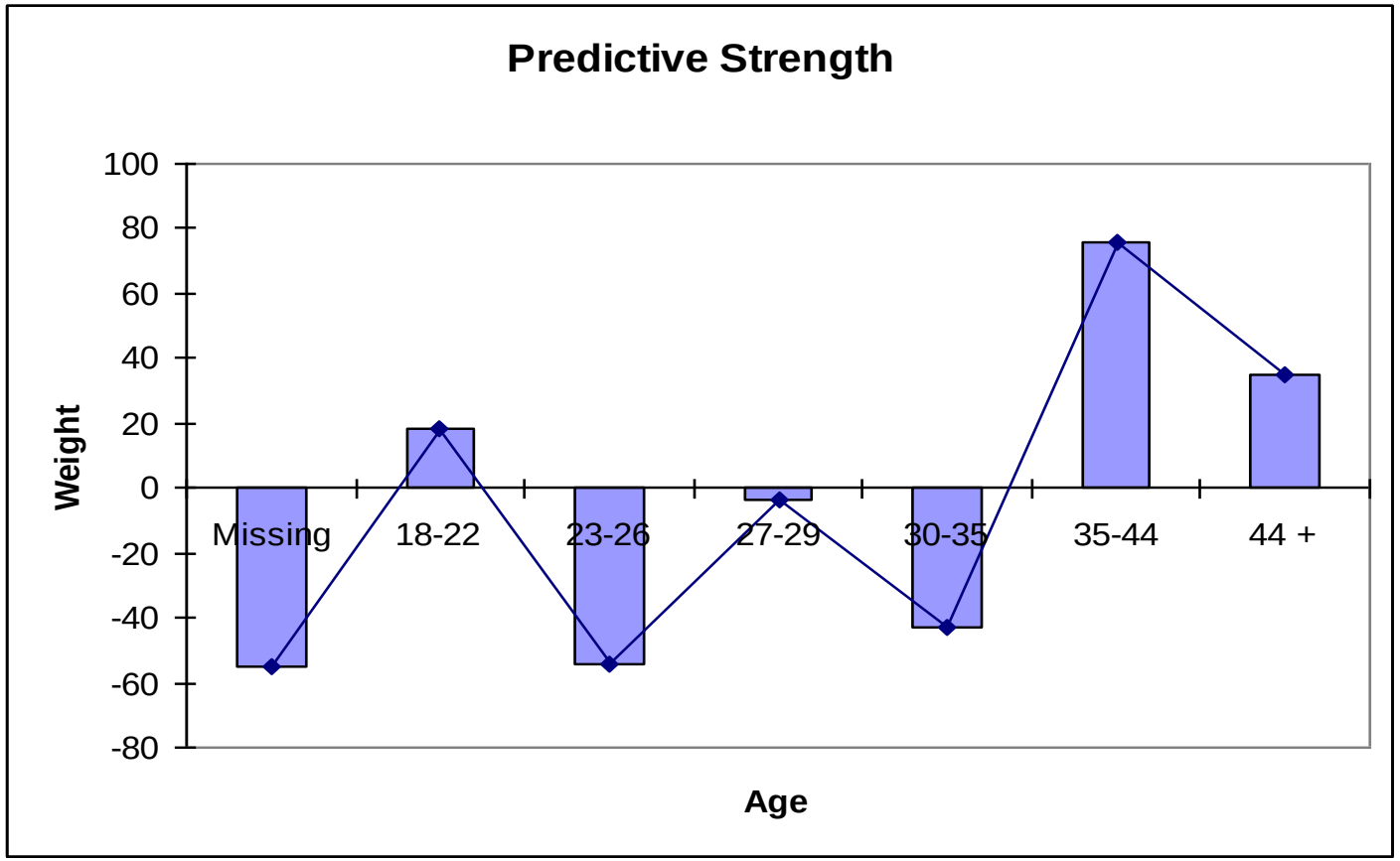
Logical Trend



Logical Trend

- Final weightings make sense.
- Enables buy-in from risk managers.
- Confirms business experience
  - young people are higher risk
  - higher debt service means higher risk
- Reduces overfitting if done right – model overall trend, not quirks. Remember how long the scorecard has to last. This is not going to be used for the next campaign and then discarded.
- Linear relationship not always true, but need trend to confirm, and back up with business experience. E.g. revolving open burden shows a 'banana curve' everywhere and is now accepted as that. People don't try to make it straight.

Logical Trend

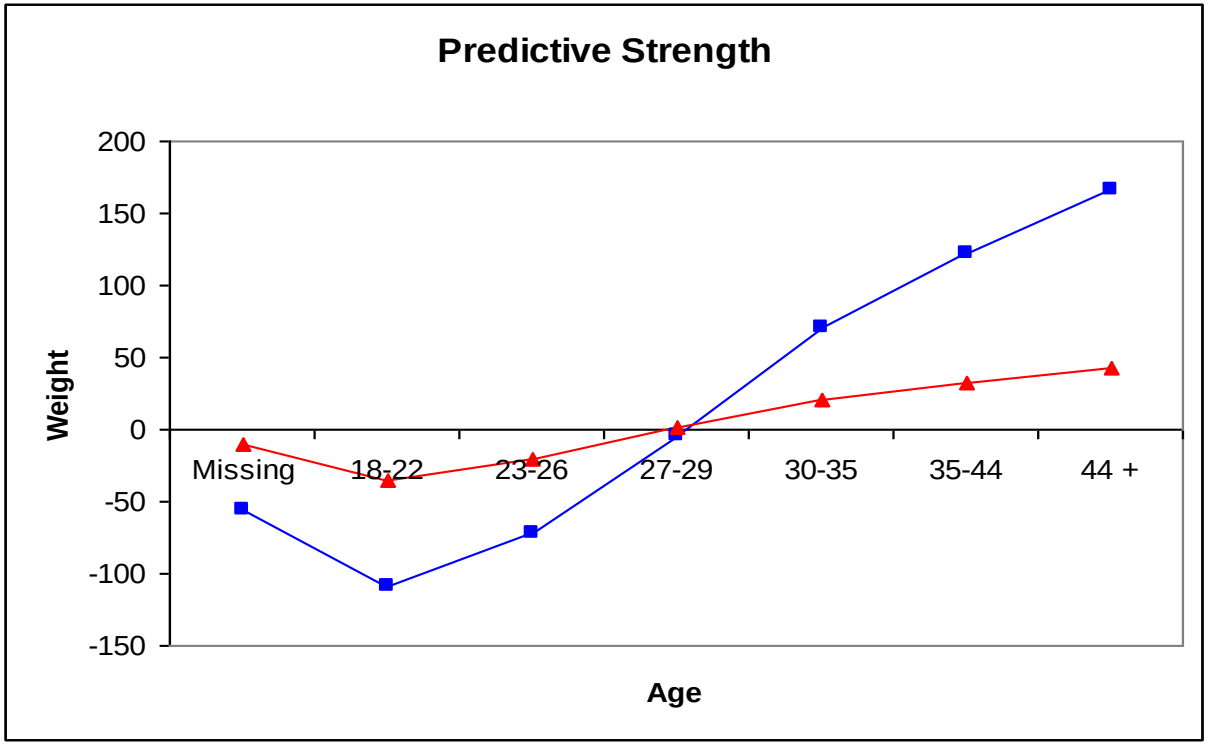


Obviously not a logical trend!!!

Logical Trend

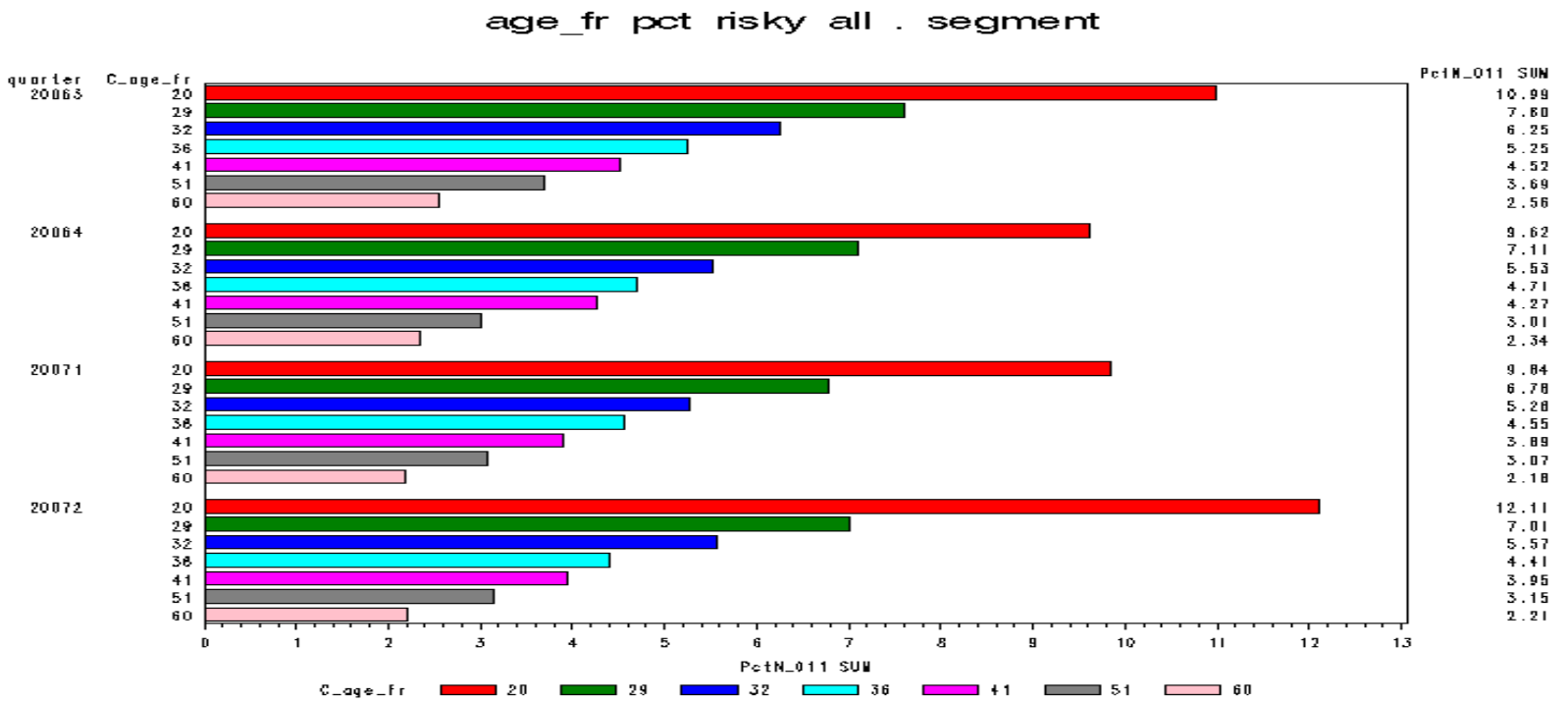
Which line shows logical trend?
Both are logical. What's the difference?

Blue line shows good differentiation.
Red line is flat, and this characteristic is likely very week and will be reflected in the *IV*.



Stability check

Check the stability of grouping throughout the whole developmnet time window:



Business Factors

- Nominal values
  - group based on similar weight (for example, postal code, occupation)
  - investigate splits on urban/rural, regional
- Breaks concurrent with policy rules
- Sanity check.

Variable Selection

List of information values of variables (predictors)

No	Character	IV Rank	Information Value
1	Max delinq L9M	1	0.176
2	Months since delinquent	2	0.176
3	Active contract (Y/N)	3	0.045
4	Average Delinquency L9M	4	0.087
5	**Months since >10 dpd**	5	0.144
6	Max delinq L3M	6	0.117
7	Average Delinquency L3M	7	0.108
8	Age of oldest contract	8	0.013
9	Number of months on collections as % total time on book	9	0.132
10	Months since >20 dpd	10	0.091
11	**Months since >30 dpd**	11	0.054
12	Num rejected applications L9M	12	0.033
13	Times 30+ dpd L9M	13	0.042
14	Total Payment L3M	14	0.018
15	Months since >40 dpd	15	0.030
16	Current balance as % of highest ever balance	16	0.048
17	Times 30+ dpd L3M	17	0.024
18	Payment Method	18	0.001

Cvičení –profile

```
/* 2b. Profiles */

%let input=income;
%let groups=yes;
%let n_groups=4;

/* grouping 1 - kvantily */

proc rank data=indata.accepts (keep=&input) groups=&n_groups
out=bins;
var &input;
ranks bin;
run;
proc summary data=bins nway missing;
class bin;
output out=bins (drop=_type_) min(&input)=start max(&input)=end;
run;
data bins;
set bins;
label=compress(put(start,best.))||' - '||compress(put(end,best.));
fmtname='__bin';
type='N';
run;
proc format cntlin=bins;
run;
```

```
%macro profile(input,groups);

/* Profile of &input according to BAD60 */

proc summary data=indata.accepts;
class &input;
output out=__bins (drop=_type_ rename=(_freq_=__n))
       sum(bad60)=__n1;
%if %upcase(&groups)=YES %then %do;
   format &input __bin.;
%end;
run;

data __bins;
set __bins end=__finish;
if _n_=1 then do;
   __all_n=__n;
   __all_n1=__n1;
   __all_n0=__n-__n1;
   retain __all_n:;
end;
else do;
   __p=__n/__all_n;
   __n0=__n-__n1;
   __p1=__n1/__all_n1;
   __p0=__n0/__all_n0;
   __r1=__n1/__n;
   __r0=__n0/__n;
   __woe=log((__p0)/(__p1))*100;
   __all_iv+(__p0-__p1)*__woe/100;
   output;
end;
if __finish then do;
   call symput('groups',compress(put(_n_-1,best.)));
   call symput('iv',compress(put(__all_iv,8.4)));
   call symput('br',compress(put(__all_n1/__all_n,best.)));
end;
attrib
   __n   label='N'
   __p   label='%'                    format=percent8.1
   __n1  label="N of Bad"
   __n0  label="N of Good"
   __p1  label="% of Bad"  format=percent8.1
   __p0  label="% of Good"  format=percent8.1
   __r1  label="Bad rate"  format=percent8.1
   __r0  label="Good rate"  format=percent8.1
   __woe label='WOE'             format=8.2
   &input   label="Group of &input"
;
drop __all:;
Run;

.
.
.
```

```
data __chart (keep=&input __sub __n __p __r);
set __bins (keep=&input __n0 __p0 __r0 __n1 __p1 __r1);
length __sub $4;
__sub="Good";
__n=__n0;
__p=__p0;
__r=__r0;
output;
__sub="Bad";
__n=__n1;
__p=__p1;
__r=__r1;
output;
attrib
    __n   label='N'      format=8.0
    __p   label='%'      format=percent8.1
    __r   label='Rate'   format=percent8.1
    __sub label='Target'
;
run;

proc datasets nolist;
delete gseg / memtype=catalog;
quit;

ods listing close;

goptions reset=all ftext='arial' htext=1.5 ftitle='arial' htitle=2;

proc gchart data=__chart;
axis1 style=0;
axis2 minor=none order=(0 to 1 by .25) label=none;
axis3 minor=none label=none;
axis4 minor=(n=4) label=none;
where __sub="Bad";
hbar &input / discrete sumvar=__r noframe nostats
            maxis=axis1 raxis=axis3 autoref cref=graya0 clipref
            name="__1";
title "Bad rates";
run;
where;
hbar &input / discrete subgroup=__sub sumvar=__n noframe nostats
            maxis=axis1 raxis=axis3 autoref cref=graya0 clipref
            name="__2";
title "Bad / Good frequencies";
run;
Quit;
```

```
proc gchart data=__bins;
hbar &input / discrete sumvar=__woe noframe nostats
            maxis=axis1 raxis=axis4 autoref cref=graya0 clipref
            name="__3";
title "Weight of evidence";
run;
hbar &input / discrete sumvar=__p1 noframe nostats
            maxis=axis1 raxis=axis4 autoref cref=graya0 clipref
            name="__4";
title "Bad distribution";
run;
quit;


ods html path="&appl_root" file="5.profile.html" style=statdoc;

proc report data=__bins nofs style(summary)=[htmlclass="Header"];
columns ("Attributes of &input" &input) ('Total' __n __p)
        ("Good" __n0 __p0) ("Bad" __n1 __p1) ('Measures' __r1 __woe);
define &input / group;
compute after;
__r1.sum=&br;
__woe.sum=.;
endcomp;
rbreak after / summarize;
title "Bad / Good by &input";
footnote "IV=&iv (<0.02 unpredictive, <0.1 week, <0.3 medium, <0.5 strong, >0.5 over)";
run;

goptions device=gif;
proc greplay nofs;
footnote;
igout gseg;
tc sashelp.templt;
template l2r2;
treplay 1:__1 2:__2 3:__3 4:__4 name="5_profil";
run;
quit;
title;
footnote;

ods html close;

ods listing;

%mend profile;

%profile(&input,&groups)
```

Bad / Good by income

Attributes of income	Total		Good		Bad		Measures	
Group of income	N	%	N of Good	% of Good	N of Bad	% of Bad	Bad rate	WOE
15000.067206 - 17541.45177	16147	25.0%	15631	25.0%	516	26.1%	3.2%	-4.55
17541.61083 - 19631.429437	16147	25.0%	15688	25.1%	459	23.2%	2.8%	7.52
19631.471069 - 21723.106242	16148	25.0%	15683	25.0%	465	23.5%	2.9%	6.19
21723.273059 - 35790.940583	16147	25.0%	15612	24.9%	535	27.1%	3.3%	-8.29
	64589	100.0%	62614	100.0%	1975	100.0%	3.1%	.

IV=0.0046 (<0.02 unpredictive, <0.1 week, <0.3 medium, <0.5 strong, >0.5 over)

Cvičení

```
/*profile multiple characteristics at once*/
%model_profilevar
(
    data=data.accepts,
    interval=age income idratio ,
    binary=sex phone client,
    ordinal=age_grp income_grp region,
    groups=5,
    target=bad30,
    rep_out=&appl_root
)
```

Bad / Good by Sex

Attributes of sex		Total		Good		Bad		Measures	
Group of sex	Sex	N	%	N of Good	% of Good	N of Bad	% of Bad	Bad rate	WOE
M	M	45138	69.9%	42061	69.6%	3077	74.7%	6.8%	-7.16
Z	Z	19451	30.1%	18410	30.4%	1041	25.3%	5.4%	18.59
		64589	**100.0%**	**60471**	**100.0%**	**4118**	**100.0%**	**6.4%**	**.**

Unpredictive (IV = 0.0133, 2 groups)

Bad / Good by Phone member?

Attributes of phone		Total		Good		Bad		Measures	
Group of phone	Phone member?	N	%	N of Good	% of Good	N of Bad	% of Bad	Bad rate	WOE
0	0	8081	12.5%	7431	12.3%	650	15.8%	8.0%	-25.04
1	1	56508	87.5%	53040	87.7%	3468	84.2%	6.1%	4.07
		64589	**100.0%**	**60471**	**100.0%**	**4118**	**100.0%**	**6.4%**	**.**

Unpredictive (IV = 0.0102, 2 groups)

Bad / Good by Existing client?

Attributes of client		Total		Good		Bad		Measures	
Group of client	Existing client?	N	%	N of Good	% of Good	N of Bad	% of Bad	Bad rate	WOE
0	0	60188	93.2%	56251	93.0%	3937	95.6%	6.5%	-2.74
1	1	4401	6.8%	4220	7.0%	181	4.4%	4.1%	46.23
		64589	**100.0%**	**60471**	**100.0%**	**4118**	**100.0%**	**6.4%**	**.**

Unpredictive (IV = 0.0126, 2 groups)

Cvičení

Bad / Good by Age groups

Attributes of age_grp		Total		Good		Bad		Measures	
Group of age_grp	Age groups	N	%	N of Good	% of Good	N of Bad	% of Bad	Bad rate	WOE
do 30	do 30	2957	4.6%	2662	4.4%	295	7.2%	10.0%	-48.69
30 - 60	30 - 60	58713	90.9%	55057	91.0%	3656	88.8%	6.2%	2.52
nad 60	nad 60	2919	4.5%	2752	4.6%	167	4.1%	5.7%	11.53
		64589	**100.0%**	**60471**	**100.0%**	**4118**	**100.0%**	**6.4%**	**.**

Unpredictive (IV = 0.0146, 3 groups)

Bad / Good by Income groups

Attributes of income_grp		Total		Good		Bad		Measures	
Group of income_grp	Income groups	N	%	N of Good	% of Good	N of Bad	% of Bad	Bad rate	WOE
do 17	do 17	12070	18.7%	11213	18.5%	857	20.8%	7.1%	-11.54
17 - 22	17 - 22	37859	58.6%	35567	58.8%	2292	55.7%	6.1%	5.52
22 - 27	22 - 27	13680	21.2%	12820	21.2%	860	20.9%	6.3%	1.50
nad 27	nad 27	980	1.5%	871	1.4%	109	2.6%	11.1%	-60.85
		64589	**100.0%**	**60471**	**100.0%**	**4118**	**100.0%**	**6.4%**	**.**

Unpredictive (IV = 0.0118, 4 groups)

Cvičení

Bad / Good by Region

Attributes of region		Total		Good		Bad		Measures	
Group of region	Region	N	%	N of Good	% of Good	N of Bad	% of Bad	Bad rate	WOE
1	1	12537	19.4%	11404	18.9%	1133	27.5%	9.0%	-37.77
2	2	16335	25.3%	15498	25.6%	837	20.3%	5.1%	23.18
3	3	10679	16.5%	10034	16.6%	645	15.7%	6.0%	5.77
4	4	10797	16.7%	10170	16.8%	627	15.2%	5.8%	9.95
5	5	7199	11.1%	6783	11.2%	416	10.1%	5.8%	10.47
6	6	7042	10.9%	6582	10.9%	460	11.2%	6.5%	-2.59
		64589	**100.0%**	**60471**	**100.0%**	**4118**	**100.0%**	**6.4%**	**.**

Weak predictivity (IV = 0.0483, 6 groups)

Cvičení

Bad / Good by Age

Attributes of age		Total		Good		Bad		Measures	
Group of age	Age	N	%	N of Good	% of Good	N of Bad	% of Bad	Bad rate	WOE
0	18 - 35	13688	21.2%	12420	20.5%	1268	30.8%	9.3%	-40.49
1	36 - 40	11385	17.6%	10485	17.3%	900	21.9%	7.9%	-23.15
2	41 - 45	14645	22.7%	13918	23.0%	727	17.7%	5.0%	26.52
3	46 - 51	12383	19.2%	11806	19.5%	577	14.0%	4.7%	33.17
4	52 - 74	12488	19.3%	11842	19.6%	646	15.7%	5.2%	22.18
		64589	**100.0%**	**60471**	**100.0%**	**4118**	**100.0%**	**6.4%**	.

Weak predictivity (IV = 0.0931, 5 groups)

Cvičení

Bad / Good by Income

Attributes of income		Total		Good		Bad		Measures	
Group of income	Income	N	%	N of Good	% of Good	N of Bad	% of Bad	Bad rate	WOE
0	15.000 - 17.105	12917	20.0%	12011	19.9%	906	22.0%	7.0%	-10.23
1	17.105 - 18.822	12918	20.0%	12204	20.2%	714	17.3%	5.5%	15.18
2	18.822 - 20.398	12918	20.0%	12122	20.0%	796	19.3%	6.2%	3.64
3	20.398 - 22.339	12918	20.0%	12102	20.0%	816	19.8%	6.3%	0.99
4	22.340 - 35.791	12918	20.0%	12032	19.9%	886	21.5%	6.9%	-7.82
		64589	**100.0%**	**60471**	**100.0%**	**4118**	**100.0%**	**6.4%**	**.**

Unpredictive (IV = 0.0080, 5 groups)

Cvičení

Bad / Good by Income/Debt ratio

Attributes of idratio		Total		Good		Bad		Measures	
Group of idratio	Income Debt ratio	N	%	N of Good	% of Good	N of Bad	% of Bad	Bad rate	WOE
0	0.0175 - 0.0225	12917	20.0%	11994	19.8%	923	22.4%	7.1%	-12.23
1	0.0225 - 0.0293	12918	20.0%	11998	19.8%	920	22.3%	7.1%	-11.87
2	0.0293 - 0.0421	12918	20.0%	12061	19.9%	857	20.8%	6.6%	-4.25
3	0.0421 - 0.0714	12918	20.0%	12216	20.2%	702	17.0%	5.4%	16.98
4	0.0714 - 0.2995	12918	20.0%	12202	20.2%	716	17.4%	5.5%	14.89
		64589	**100.0%**	**60471**	**100.0%**	**4118**	**100.0%**	**6.4%**	**.**

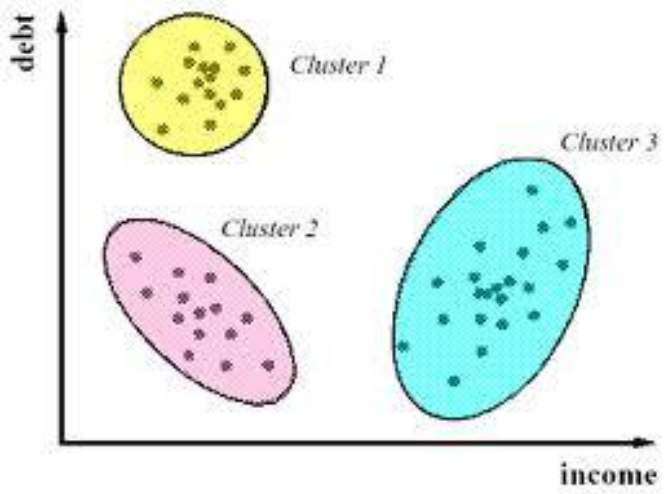
Unpredictive (IV = 0.0160, 5 groups)

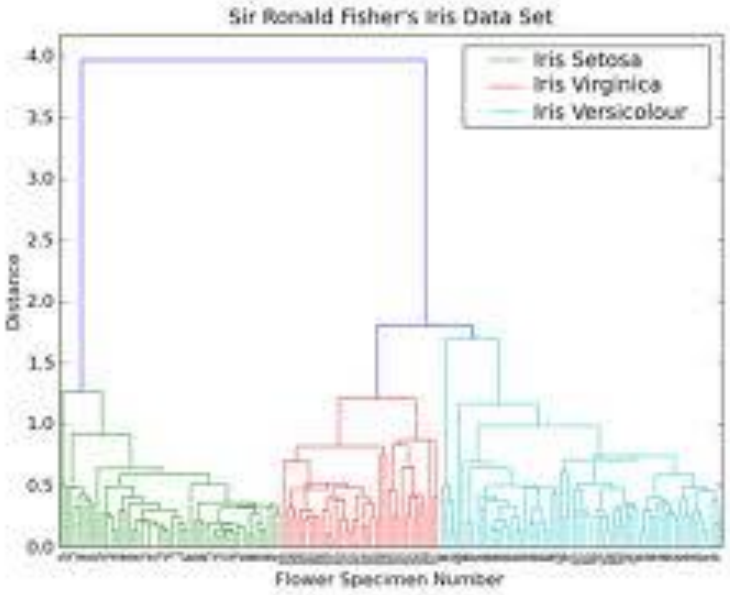
Cvičení

Profiles for target DATA.ACCEPTS.BAD30

Type	Type	Label	Information value	Note	Format
BINARY	BINARY	SEX - Sex	0.0133	Unpredictive (IV = 0.0133, 2 groups)	
BINARY	BINARY	CLIENT - Existing client?	0.0126	Unpredictive (IV = 0.0126, 2 groups)	
BINARY	BINARY	PHONE - Phone member?	0.0102	Unpredictive (IV = 0.0102, 2 groups)	
INTERVAL	INTERVAL	AGE - Age	0.0931	Weak predictivity (IV = 0.0931, 5 groups)	
INTERVAL	INTERVAL	IDRATIO - Income/Debt ratio	0.0160	Unpredictive (IV = 0.0160, 5 groups)	8.4
INTERVAL	INTERVAL	INCOME - Income	0.0080	Unpredictive (IV = 0.0080, 5 groups)	COMMAX20.
ORDINAL	ORDINAL	REGION - Region	0.0483	Weak predictivity (IV = 0.0483, 6 groups)	
ORDINAL	ORDINAL	AGE_GRP - Age groups	0.0146	Unpredictive (IV = 0.0146, 3 groups)	
ORDINAL	ORDINAL	INCOME_GRP - Income groups	0.0118	Unpredictive (IV = 0.0118, 4 groups)	

7. Úvod do shlukové analýzy (SA). Hierarchická SA





Úvod

Shluková (klastrová, z angl. Cluster) analýza je metoda, která na základě informací obsažených ve vícerozměrných pozorováních roztřídí základní množinu objektů do několika relativně stejnorodých shluků. Uvažujeme datovou matici typu *n x p,* kde *n* je počet objektů a *p* je počet proměnných. Uvažujeme různé rozklady množiny *n* objektů do *g* shluků a hledáme takový rozklad, který je z určitého hlediska nejvýhodnější. Cílem je dosáhnout stavu, kdy objekty uvnitř shluku jsou si podobné co nejvíce a objekty z různých shluků si jsou podobné co nejméně.

Unsupervised Learning

- Metody shlukové analýzy patří mezi tzv. „unsupervised learning“ metody.
- “Learning without *a priori* knowledge about the classification of samples; learning without a teacher.”

Kohonen (1995), *“Self-Organizing Maps”*

Cluster Profiling

- Cluster profiling can be defined as the derivation of a class label from a proposed cluster solution.
- The objective is to identify the features, or combination of features, that uniquely describe each cluster.

Types of clustering

- Rozlišujeme tři základní shlukovací metody:
  - Hiearchické shlukování (hierarchical clustering),
  - Shlukování s předem neznámým počtem shluků - s možným překryvem shluků (overlapping clusters),
  - Shlukování do předem daného počtu shluků (partitive/partitioning methods).
  - Fuzzy shlukování – fuzzy shluky jsou definovány stupněm příslušnosti objektů do daných shluků.

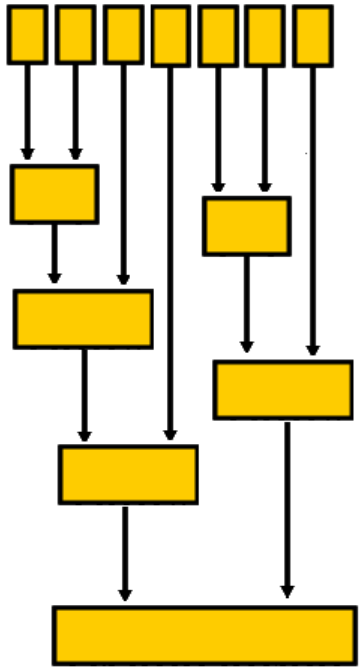
Klasifikace shlukovacích algoritmů

- Hiearchické shlukovací algoritmy:
  - Aglomerativní
  - Divisive
- Partitioning algoritms:
  - K-means
  - K-medoids
  - Probabilistic
  - Density based
- Grid-based algoritms
- Constraint-Based Clustering
- Evolutionary Algoritms
- Scalable Clustering Algoritms

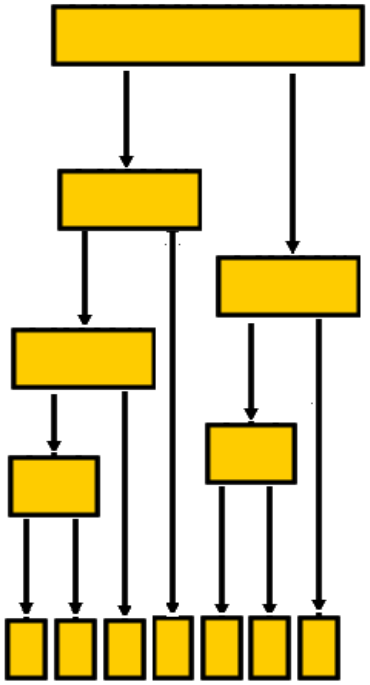
…

Hierarchical Clustering

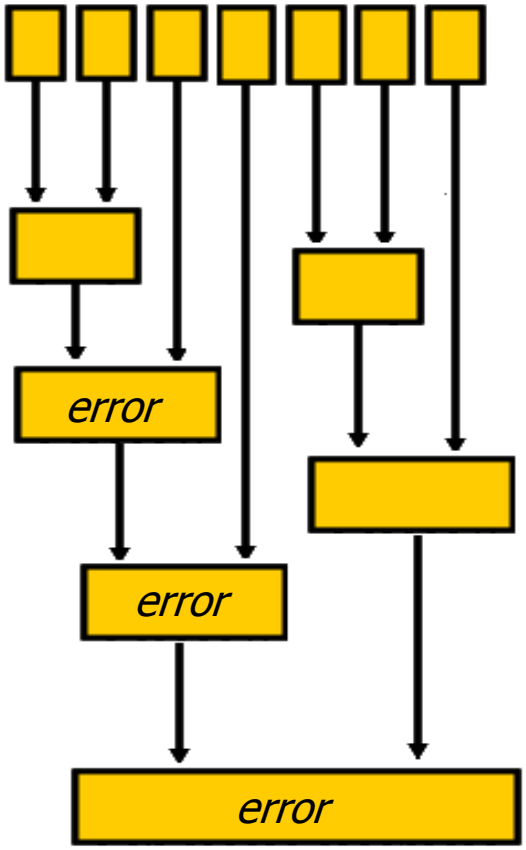
Agglomerative



Divisive

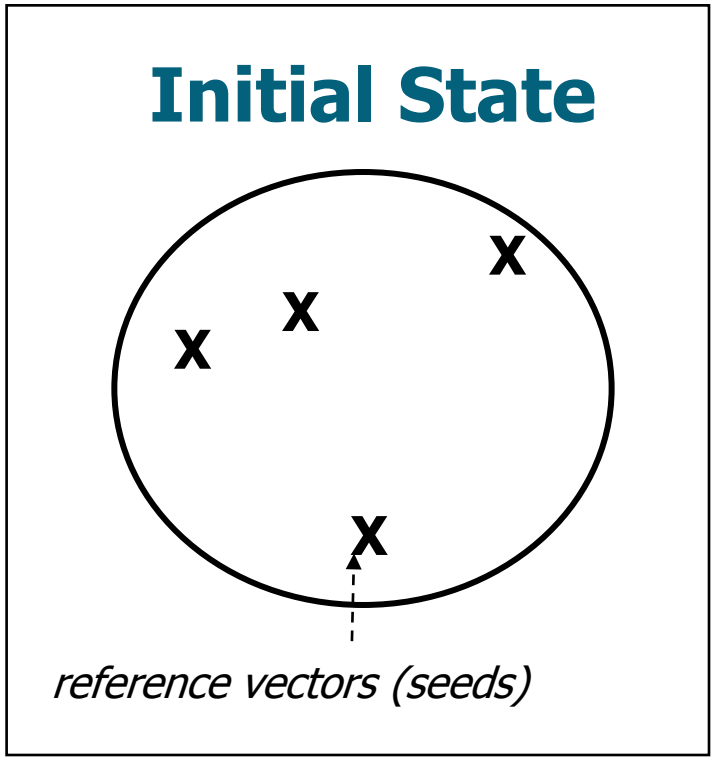


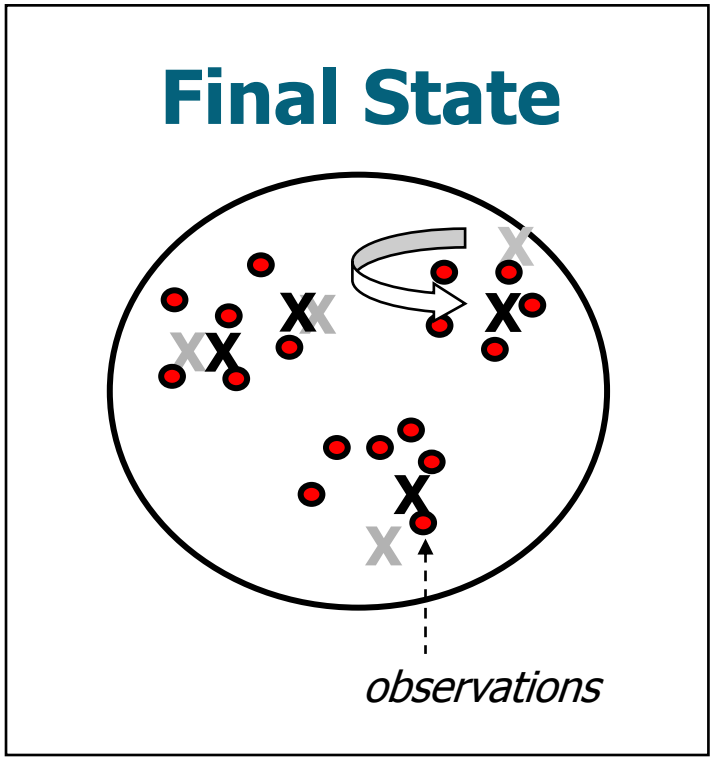
Problems with Hierarchical Clustering



Partitive Clustering

- The goal of partitive clustering is to minimize or maximize some specified criterion.





Problems with Partitive Clustering

- Many partitive clustering methods
  a. make you guess the number of clusters present,
  b. make assumptions about the shape of the clusters, usually that they are (hyper)spherical, and
  c. are influenced by seed location, by outliers, even by the order the observations are read in.
- It is impossible to determine the optimal grouping, due to the combinatorial explosion of potential solutions.

Problems with Partitive Clustering

- The number of possible partitions of n objects into g groups is given by:

$$\mathrm{N}(n,g) = \frac{1}{g!}\sum_{m=1}^{g}(-1)^{g-m}\binom{g}{m}m^{n}$$

- For example, the number of partitions of 50 observations into 4 clusters, $N(50,4)$, is equal to 5.3×10^{28}. $N(100, 4)$ generates 6.7×10^{58} partitions. Complete enumeration of every possible partition, therefore, is generally impossible.

Heuristic Search

1. Generate an initial partitioning (based on the seeds) of the observations into clusters.
2. Calculate the change in error produced by moving each observation from its own cluster to another.
3. Make the move that produces the greatest reduction.
4. Repeat steps 2 and 3 until no move reduces error.

Hierarchická shluková analýza

- Je třeba zvolit:
  - jak měřit vzdálenost/(ne)podobnost mezi objekty (euklidovská,...)
    - do úvahy je třeba vzít typ dat (intervalová, nominální,...)
    - značnou roli také hraje souměřitelnost dat -> standardizace (z-skóre, ...)
  - jak měřit vzdálenost/(ne)podobnost mezi shluky (wardova,...)
  - jak určit finální rozklad objektů do shluků

Příklad aglomerativního hiearchického shlukování

- Uvažujeme 5 objektů A,B,C,D a E popsaných čtyřmi proměnnými X1-X4.
- Neprovádíme žádnou standardizaci.
- Vzdálenost mezi objekty měříme pomocí euklidovské vzdálenosti.
- Vzdálenosti mezi shluky měříme pomocí metody průměrné vzdálenosti (average linkage).

Data:

obj	X1	X2	X3	X4
A	100	80	70	60
B	80	60	50	40
C	80	70	40	50
D	40	20	20	10
E	50	10	20	10

Matice vzdáleností:

	A	B	C	D	E
A	0	0	0	0	0
B	40,00	0	0	0	0
C	38,73	17,32	0	0	0
D	110,45	70,71	78,10	0	0
E	111,36	72,11	80,62	14,14	0

Příklad aglomerativního hiearchického shlukování

1. krok:

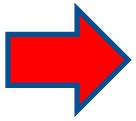
- V matici vzdáleností hledáme nejmenší hodnotu. V našem případě je to 14,1 (vzd. mezi D a E).
- Sloučíme objekty D a E do shluku D' , zmenšíme a přepočteme matici vzdáleností. Používáme metodu průměrné vzdálenosti, takže:

$$D_{AD'} = \frac{1}{n_A n_{D'}} \sum_{i \in A} \sum_{j \in D'} d\left(x_i, x_j\right) = \frac{1}{1 \cdot 2}\left(110{,}4 + 111{,}4\right) = 110{,}9$$

$$D_{BD'} = \frac{1}{1 \cdot 2}\left(70{,}7 + 72{,}1\right) = 71{,}4$$

$$D_{CD'} = \frac{1}{1 \cdot 2}\left(78{,}1 + 80{,}6\right) = 79{,}35$$

	A	B	C	D	E
A	0	0	0	0	0
B	40,00	0	0	0	0
C	38,73	17,32	0	0	0
D	110,45	70,71	78,10	0	0
E	111,36	72,11	80,62	14,14	0



	A	B	C	D'
A	0			
B	40,00	0		
C	38,73	17,3	0	
D'	110,90	71,41	79,36	0

Příklad aglomerativního hiearchického shlukování

2. krok:

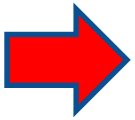
- V redukované matici vzdáleností hledáme nejmenší hodnotu. V našem případě je to 17,3 (vzd. mezi B a C).
- Sloučíme objekty B a C do shluku B' , zmenšíme a přepočteme matici vzdáleností.

$$D_{AB'} = \frac{1}{n_A n_{B'}} \sum_{i \in A} \sum_{j \in B'} d(x_i, x_j) = \frac{1}{1 \cdot 2}(40 + 38{,}7) = 39{,}35$$

	A	B	C	D	E
A	0	0	0	0	0
B	40,00	0	0	0	0
C	38,73	17,32	0	0	0
D	110,45	70,71	78,10	0	0
E	111,36	72.11	80.62	14,14	0

$$D_{D'B'} = \frac{1}{n_{D'} n_{B'}} \sum_{i \in D'} \sum_{j \in B'} d(x_i, x_j) = \frac{1}{2 \cdot 2}(70{,}7 + 78{,}1 + 72{,}1 + 80{,}6) = \frac{1}{2}(71{,}4 + 79{,}3) = 75{,}375$$

	A	B	C	D'
A	0			
B	40,00	0		
C	38,73	17,3	0	
D'	110,90	71,41	79,36	0



	A	B'	D'
A	0		
B'	39,36	0	
D'	110,90	75,39	0

Příklad aglomerativního hiearchického shlukování

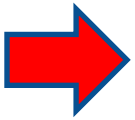
3. krok:

- V redukované matici vzdáleností hledáme opět nejmenší hodnotu. V našem případě je to 39,3 (vzd. mezi A a B').
- Sloučíme objekty A a B' do shluku A' , zmenšíme a přepočteme matici vzdáleností.

$$D_{A'D'} = \frac{1}{n_{A'} n_{D'}} \sum_{i \in A'} \sum_{j \in D'} d(x_i, x_j) = \frac{1}{3 \cdot 2}(110{,}45 + 111{,}36 + 70{,}71 + 72{,}11 + 78{,}10 + 80{,}62) = \frac{1}{3}(110{,}90 + 2 \cdot 75{,}39) = 87{,}23$$

	A	B	C	D	E
A	0	0	0	0	0
B	40,00	0	0	0	0
C	38,73	17,32	0	0	0
D	110,45	70,71	78,10	0	0
E	111,36	72,11	80,62	14,14	0

	A	B'	D'
A	0		
B'	39,36	0	
D'	110,90	75,39	0



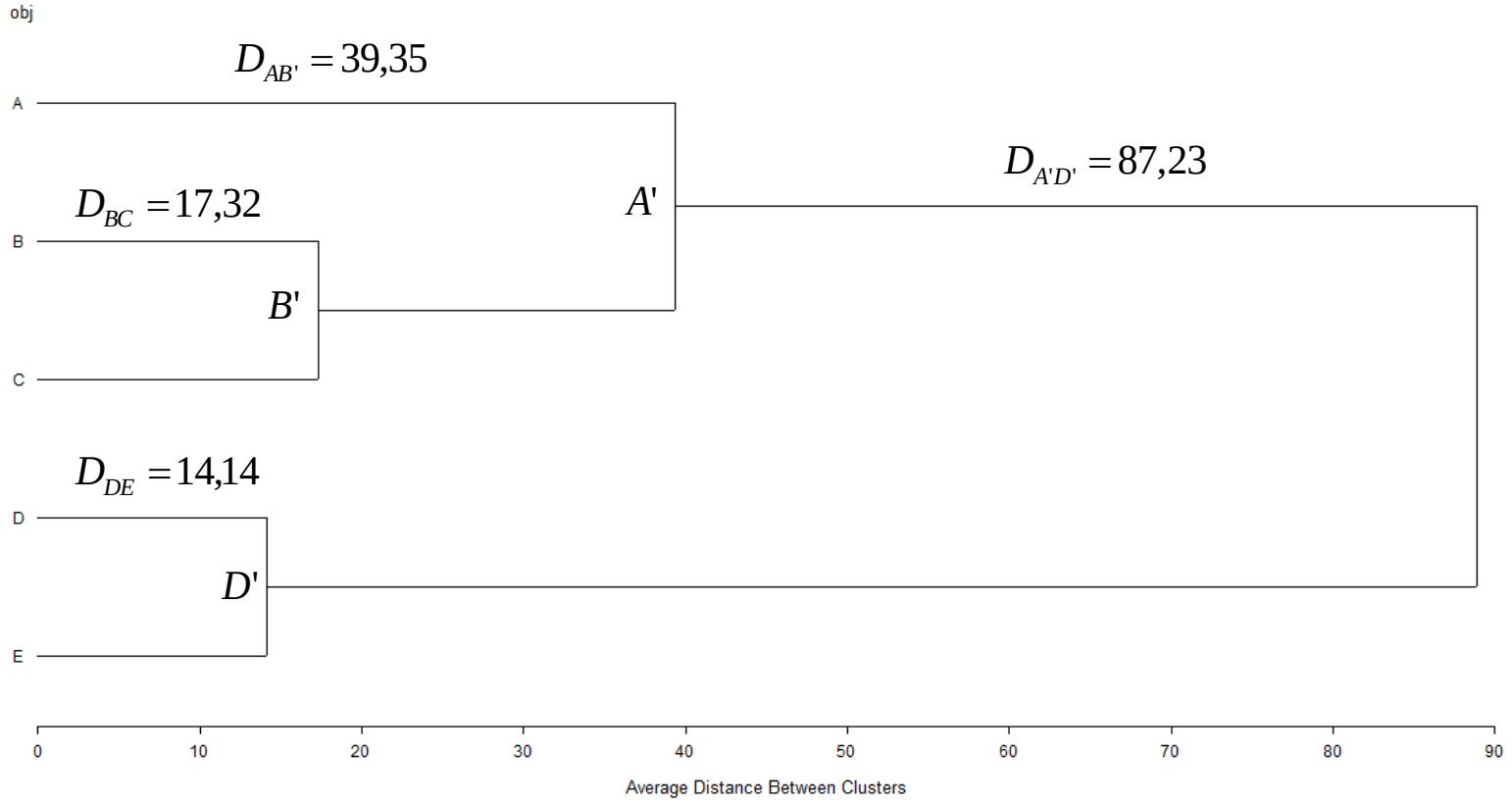
	A'	D'
A'	0	
D'	87,23	0

Pozor!!! Slučují se dva nestejně velké objekty a nelze tedy počítat obyčejný průměr průměrů!

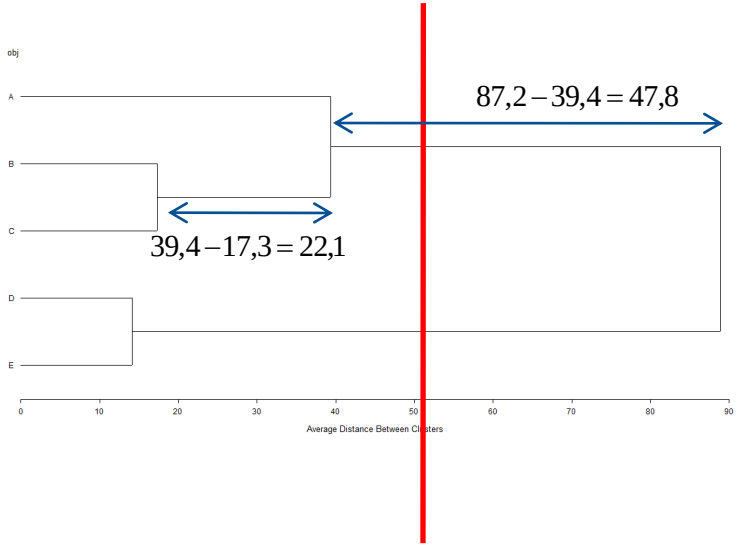
Příklad aglomerativního hiearchického shlukování

```
proc distance data=aaa method=euclid out=dist;
var interval(X1 X2 X3 X4);
id obj;
run;

proc cluster data=dist method=ave outtree=tree nonorm;
id obj;
run;

proc tree data=tree horizontal;
id obj;
run;
```

Příklad aglomerativního hiearchického shlukování



Příklad aglomerativního hiearchického shlukování



obj	X1	X2	X3	X4
A	100	80	70	60
B	80	60	50	40
C	80	70	40	50
D	40	20	20	10
E	50	10	20	10

	A	B	C	D	E
A	0				
B	40	0			
C	38,73	17,321	0		
D	110,45	70,71	78,10	0	
E	111,36	72,11	80,62	14,14	0

- Určili jsme tedy dva shluky A’ ={A, B, C} a D’ = {D, E}.

What Is Similarity?

- To illustrate the difficulties involved in judging similarity, consider your answer to the following question:

  Which is more similar to a duck,
  a crow or a penguin?

- The answer to this question largely depends on how you choose to define similarity.

- Volba míry (ne)podobnosti závisí na typu proměnných (nominální, ordinální, poměrové, intervalové, binární).

Principles of a Good Similarity Metric

- The following principles have been suggested as the foundation of a good similarity metric:

1. symmetry: $d(x,y)=d(y,x)$.
2. non-identical distinguishability: if $d(x,y)\neq 0$ then $x\neq y$.
3. identical non-distinguishability: if $d(x,y)=0$ then $x=y$.

- Most good metrics are also consistent with the triangle inequality: $d(x,y) \leq d(x,z) + d(y,z)$.

The DISTANCE Procedure

General form of the DISTANCE procedure:

```
PROC DISTANCE DATA=SAS-data-set
        METHOD=similarity-metric
<options>;
    VAR level (variables < / option-list >);
RUN;
```

- A distance method must be specified (no default), and all input variables are identified by level.

Více na:
http://support.sas.com/documentation/cdl/en/statugdistance/61780/PDF/default/statugdistance.pdf

The DISTANCE Procedure

- Metody měření vzdálenosti v SASu:

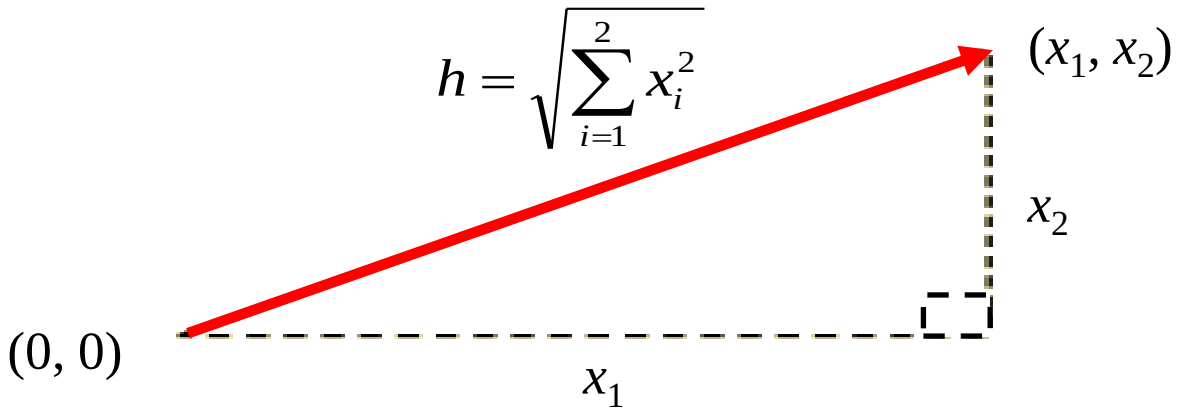
Method	Range	Type	Accepting variables
GOWER	0 to 1	sim	all
DGOWER	0 to 1	dis	all
EUCLID	≥0	dis	Ratio, interval, ordinal
SQEUCLID	≥0	dis	Ratio, interval, ordinal
SIZE	≥0	dis	Ratio, interval, ordinal
SHAPE	≥0	dis	Ratio, interval, ordinal
COV	≥0	sim	Ratio, interval, ordinal
CORR	–1 to 1	sim	Ratio, interval, ordinal
DCORR	0 to 2	dis	Ratio, interval, ordinal
SQCORR	0 to 1	sim	Ratio, interval, ordinal
DSQCORR	0 to 1	dis	Ratio, interval, ordinal
L(p)	≥0	dis	Ratio, interval, ordinal
CITYBLOCK	≥0	dis	Ratio, interval, ordinal
CHEBYCHEV	≥0	dis	Ratio, interval, ordinal
POWER(p,r)	≥0	dis	Ratio, interval, ordinal
SIMRATIO	0 to 1	sim	Ratio
DISRATIO	0 to 1	dis	Ratio
NONMETRIC	0 to 1	dis	Ratio
CANBERRA	0 to 1	dis	Ratio
COSINE	0 to 1	sim	Ratio
DOT	≥0	sim	Ratio
OVERLAP	≥0	sim	Ratio
DOVERLAP	≥0	dis	Ratio
CHISQ	≥0	dis	Ratio
CHI	≥0	dis	Ratio
PHISQ	≥0	dis	Ratio
PHI	≥0	dis	Ratio

Method	Range	Type	Accepting variables
HAMMING	0 to n	dis	Nominal
MATCH	0 to 1	sim	Nominal
DMATCH	0 to 1	dis	Nominal
DSQMATCH	0 to 1	dis	Nominal
HAMANN	–1 to 1	sim	Nominal
RT	0 to 1	sim	Nominal
SS1	0 to 1	sim	Nominal
SS3	0 to 1	sim	Nominal
DICE	0 to 1	sim	Asymmetric nominal
RR	0 to 1	sim	Asymmetric nominal
BLWNM	0 to 1	dis	Asymmetric nominal
K1	≥0	sim	Asymmetric nominal
JACCARD	0 to 1	sim	Asymmetric nominal, ratio
DJACCARD	0 to 1	dis	Asymmetric nominal, ratio

Euclidean Distance

$$D_E = \|\mathbf{x} - \mathbf{w}\| = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}$$

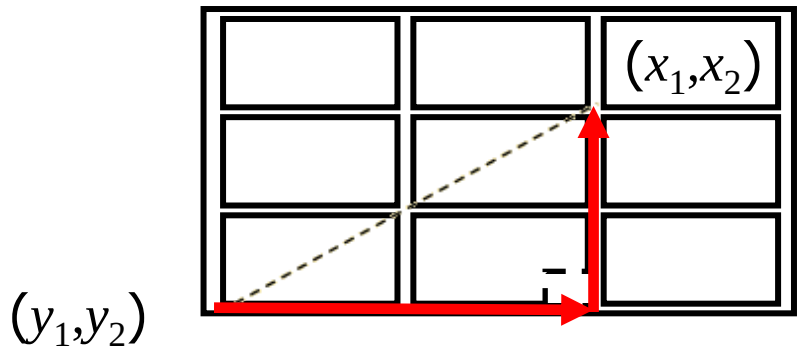
- Euclidean distance gives the linear distance between any two points in *n*-dimensional space.
- It is a generalization of the Pythagorean theorem.



City Block (Manhattan) Distance

$$D_{M_1} = \sum_{i=1}^{d} |x_i - y_i|$$

- The distance between two points is measured along the sides of a right-angled the triangle.
- It is the distance that you would travel if you had to walk along the streets of a right-angled city.



Hamming Distance

$$D_H = \sum_{i=1}^{d} |x_i - y_i|$$

	1 2 3 4 5 … 17
Gene A	0 1 1 **0** 0 **1** 0 0 1 **0 0** 1 1 1 0 **0** 1
Gene B	0 1 1 **1** 0 **0** 0 0 1 **1 1** 1 1 1 0 **1** 1
D_H =	0 0 0 **1** 0 **1** 0 0 0 **1 1** 0 0 0 0 **1** 0 = **5**

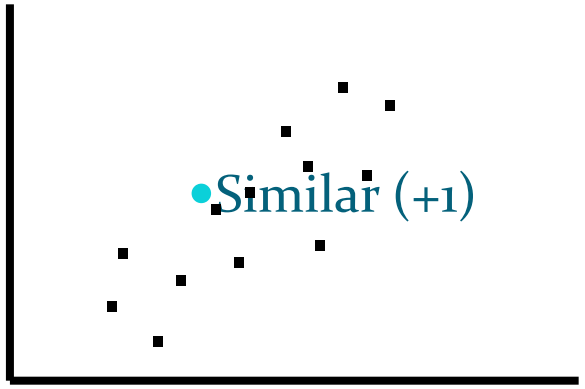
Gene expression levels under 17 conditions
(low=0, high=1)

Power Distance

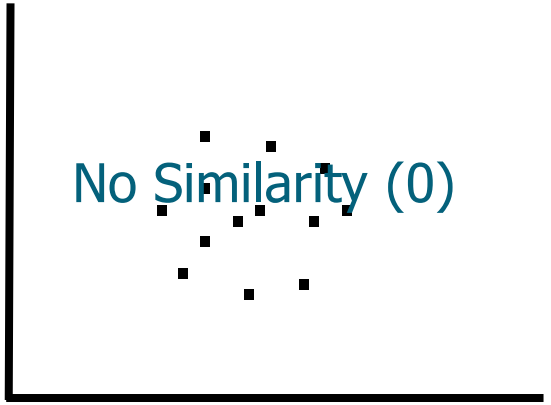
$$D_P = \sqrt[r]{\sum_{i=1}^{d} |x_i - y_i|^q}$$

- Minkowského metrika (r=q)
- Hemmingova vzdálenost (r=q=1)
- Euklidovská vzdálenost (r=q=2)
- Čebyšovova vzdálenost (r=q->∞)

Correlation



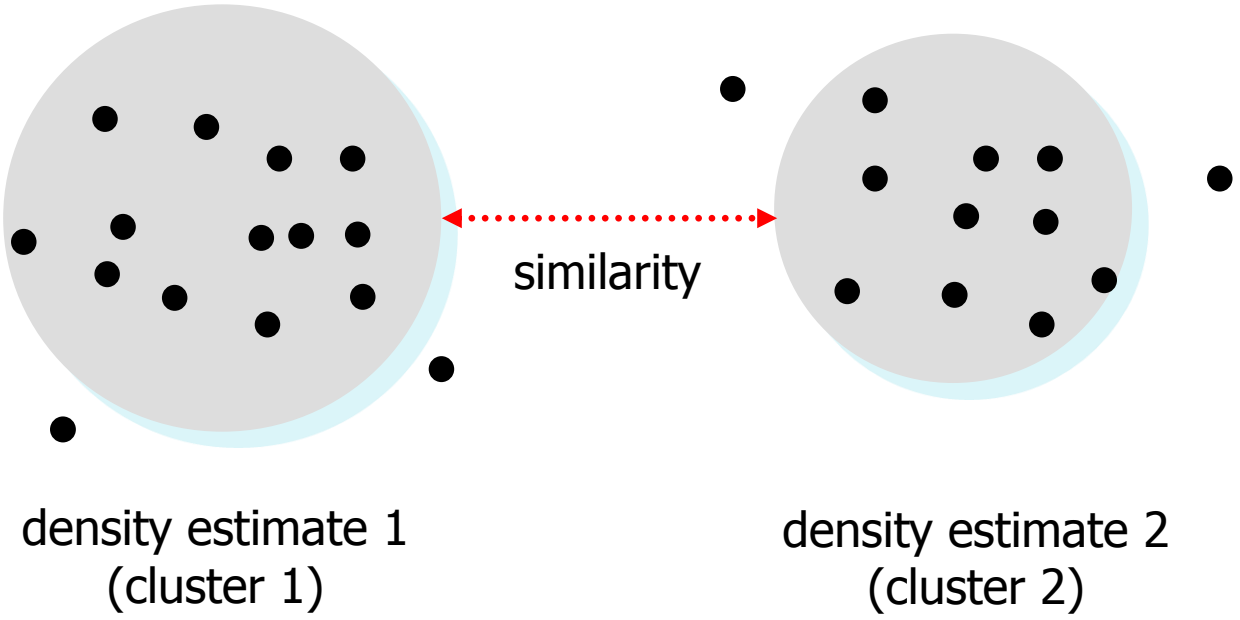
Similar (+1)



No Similarity (0)

Dissimilar (-1)

Density-Based Similarity



- Density-based methods define similarity as the distance between derived density "bubbles" (hyper-spheres).

Gower's Metric

$$D_{Gower} = \frac{\sum_{j=1}^{v} w_j \delta_j(\mathbf{x},\mathbf{y}) d_j(\mathbf{x},\mathbf{y})}{\sum_{j=1}^{v} w_j \delta_j(\mathbf{x},\mathbf{y})}$$

w_j ...váha pro j-tou proměnnou

- For nominal, ordinal, interval, or ratio variable: $\delta_j(\mathbf{x},\mathbf{y}) = 1$
- For asymmetric nominal variable:

$$\delta_j(\mathbf{x},\mathbf{y}) = 1, \quad \text{if either } x_j \text{ or } y_j \text{ is present}$$

$$\delta_j(\mathbf{x},\mathbf{y}) = 0, \quad \text{if both } x_j \text{ and } y_j \text{ are absent}$$

- For nominal or asymmetric nominal variable:

$$d_j(\mathbf{x},\mathbf{y}) = 1, \quad \text{if } x_j = y_j$$

$$d_j(\mathbf{x},\mathbf{y}) = 0, \quad \text{if } x_j \neq y_j$$

- For ordinal, interval, or ratio variable: $d_j(\mathbf{x},\mathbf{y}) = 1 - |x_j - y_j|$

- Gower's is the only similarity metric that **accepts any measurement level**.

Gower, J. C. (1971) A general coefficient of similarity and some of its properties, *Biometrics* **27**, 623–637.

Další míry podobnosti

- Jaccardův koeficient

$$D_J = \frac{\sum_{j=1}^{v} x_j y_j}{\sum_{j=1}^{v} x_j^2 + \sum_{j=1}^{v} y_j^2 - \sum_{j=1}^{v} x_j y_j}$$

- Diceův koeficient

$$D_D = \frac{2\sum_{j=1}^{v} x_j y_j}{\sum_{j=1}^{v} x_j^2 + \sum_{j=1}^{v} y_j^2}$$

- Czekanowského koeficient

$$D_C = \frac{2\sum_{j=1}^{v} \min(x_j, y_j)}{\sum_{j=1}^{v} (x_j + y_j)}$$

Míry podobnosti pro binární data

	Kategorie objektu w	
Kat. objektu x	**1**	**0**
1	**a**	**b**
0	**c**	**d**

- Koeficient souhlasu $\frac{a+d}{a+b+c+d}$
- Jaccardův koeficient $\frac{a}{a+b+c}$
- Diceův (Czekanowského) koef. $\frac{2a}{2a+b+c}$
- Yuleův koeficient $\frac{\sqrt{ad}-\sqrt{bc}}{\sqrt{ad}+\sqrt{bc}}$

Míry (ne)podobnosti pro binární data

- Goodman-Kruskalovo lambda

$$\frac{\max(a,b)+\max(c,d)+\max(a,c)+\max(b,d)-\max(a+c,b+d)-\max(a+b,c+d)}{2(a+b+c+d)-\max(a+c,b+d)-\max(a+b,c+d)}$$

- Binární Lanceova-Williamsova míra nepodobnosti $\frac{b+c}{2a+b+c}$

- Euklidovská vzdálenost $\sqrt{b+c}$

- Bin. čtvercová euklid. vzdálenost (=Hammingova vzd.) $b+c$

Standardizace/normalizace

- Před vlastním výpočtem vzdáleností je nanejvýš vhodné standardizovat (normalizovat) proměnné.
- Důvodem je snaha o unifikaci měřítka a tím vyvážení vlivu jednotlivých proměnných.
- Typicky: $$x_{\text{standard.}} = \frac{x - \text{location}}{\text{scale}}$$
- Obecně: $$\text{result} = \text{add} + \text{multiply} \cdot \frac{\text{original} - \text{location}}{\text{scale}}$$

result	= final output value
add	= constant to add (ADD= option)
multiply	= constant to multiply by (MULT= option)
original	= original input value
location	= location measure
scale	= scale measure

The STDIZE Procedure

General form of the STDIZE procedure:

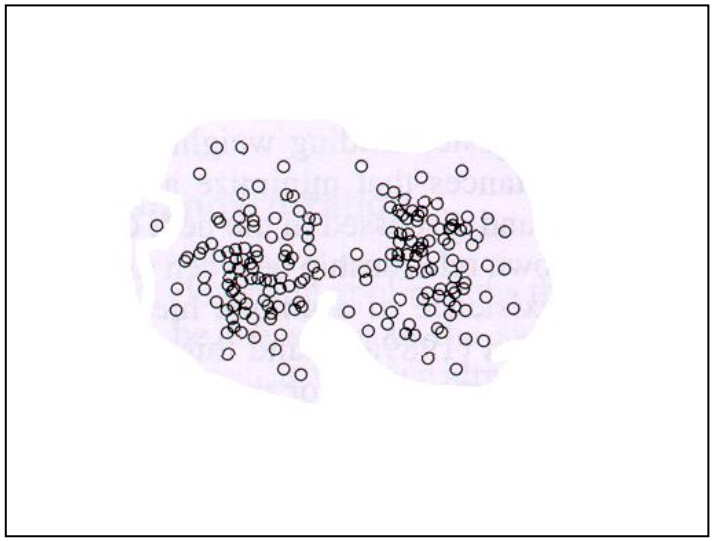
```
PROC STDIZE DATA=SAS-data-set
        METHOD=method <options>;
    VAR variables;
RUN;
```

Standardization Methods

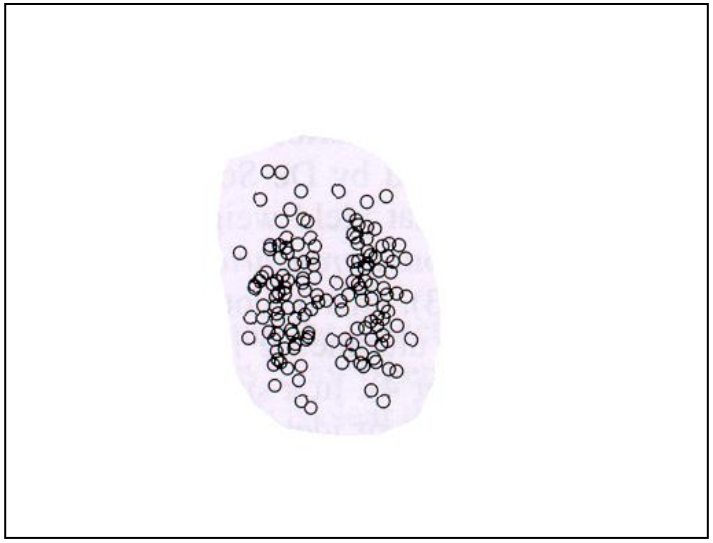
METHOD	LOCATION	SCALE
MEAN	mean	1
MEDIAN	median	1
SUM	0	sum
EUCLEN	0	Euclidean Length
USTD	0	standard deviation about origin
STD	mean	standard deviation
RANGE	minimum	range
MIDRANGE	midrange	range/2
MAXABS	0	maximum absolute value
IQR	median	interquartile range
MAD	median	median absolute deviation from median
ABW(c)	biweight 1-step M-estimate	biweight A-estimate
AHUBER(c)	Huber 1-step M-estimate	Huber A-estimate
AWAVE(c)	Wave 1-step M-estimate	Wave A-estimate
AGK(p)	mean	AGK estimate (ACECLUS)
SPACING(p)	mid minimum-spacing	minimum spacing
L(p)	L(p)	L(p) (Minkowski distances)
IN(ds)	read from data set	read settings from data set "ds"

- „Z-skóre“

The Problem with Z-Score Standardization



Before



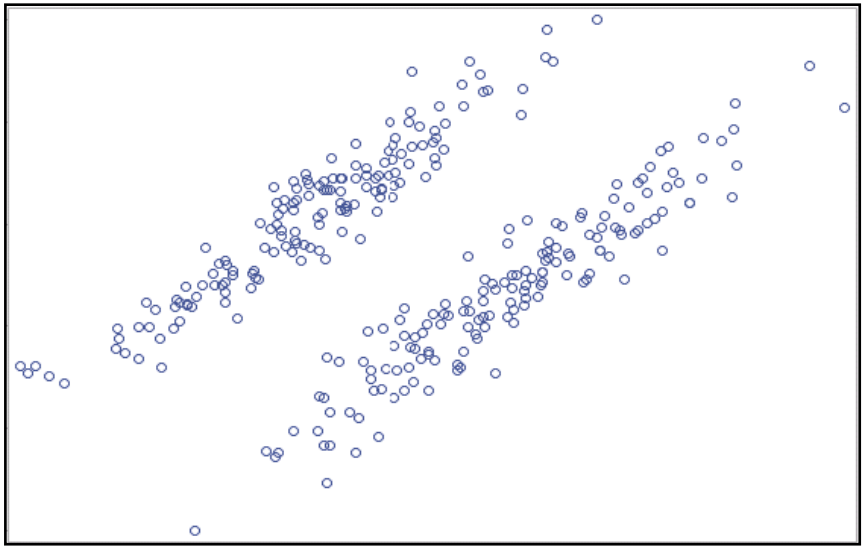
After

Standardization using the reciprocal of the variance can actually **dilute** the differences between groups!

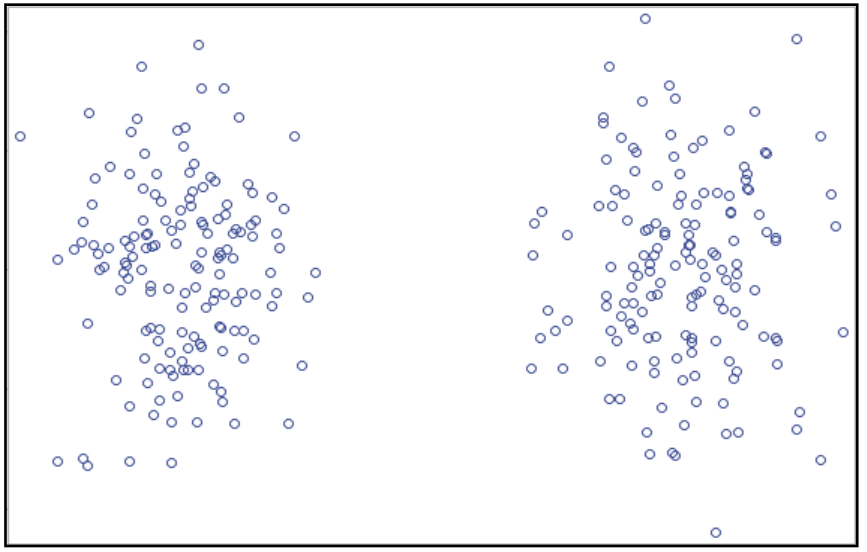
Source: Everitt et al. (2001)

Cluster Preprocessing

- Řešením tohoto problému může být procedura ACECLUS (approximate covariance estimation for clustering)



Before ACECLUS



After ACECLUS

Více na: http://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_aceclus_sect002.htm

The ACECLUS Procedure

General form of the ACECLUS procedure:

```
PROC ACECLUS DATA=SAS-data-set
<options>;
    VAR variables;
RUN;
```

Vzdálenost mezi shluky

- Mimo určení jak měřit vzdálenosti mezi objekty uvnitř shluků je třeba definovat jak měřit vzdálenosti shluků mezi sebou.

- Mezi základní metody patří metoda:
  - metoda nejbližšího souseda (single linkage),
  - metoda nejvzdálenějšího souseda (complete linkage),
  - metoda průměrné vazby (average linkage),
  - centroidní metoda,
  - Wardova metoda.

The CLUSTER Procedure

The general form of the CLUSTER procedure:

```
PROC CLUSTER DATA=SAS-data-set
        METHOD=method <options>;
    VAR variables;
    FREQ variable;
    RMSSTD variable;
RUN;
```

- The required **METHOD=** option specifies the hierarchical technique to be used to cluster the observations.

The CLUSTER Procedure, method=…

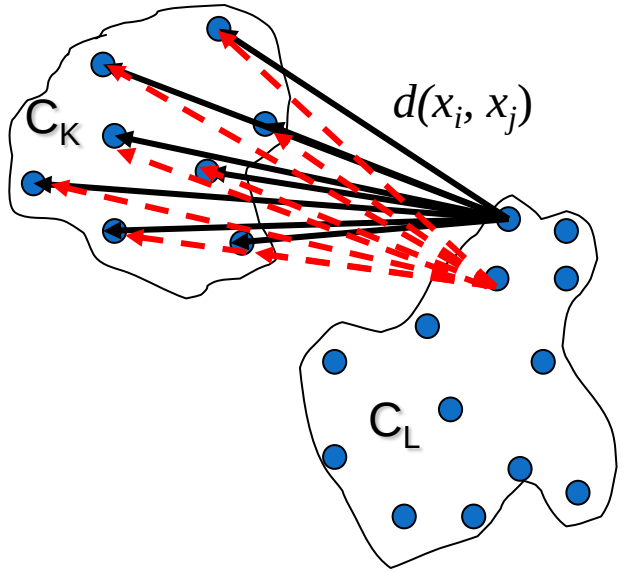
- The METHOD= specification determines the clustering method used by the procedure. Any one of the following 11 methods can be specified for *name*:
  - AVERAGE | AVE requests average linkage (group average, unweighted pair-group method using arithmetic averages, UPGMA). Distance data are squared unless you specify the NOSQUARE option.
  - CENTROID | CEN requests the centroid method (unweighted pair-group method using centroids, UPGMC, centroid sorting, weighted-group method). Distance data are squared unless you specify the NOSQUARE option.
  - COMPLETE | COM requests complete linkage (furthest neighbor, maximum method, diameter method, rank order typal analysis). To reduce distortion of clusters by outliers, the TRIM= option is recommended.
  - DENSITY | DEN requests density linkage, which is a class of clustering methods using nonparametric probability density estimation. You must also specify either the K=, R=, or HYBRID option to indicate the type of density estimation to be used. See also the MODE= and DIM= options in this section.
  - EML requests maximum-likelihood hierarchical clustering for mixtures of spherical multivariate normal distributions with equal variances but possibly unequal mixing proportions. Use METHOD=EML only with coordinate data. See the PENALTY= option for details. The NONORM option does not affect the reported likelihood values but does affect other unrelated criteria. The EML method is much slower than the other methods in the CLUSTER procedure.
  - FLEXIBLE | FLE requests the Lance-Williams flexible-beta method. See the BETA= option in this section.
  - MCQUITTY | MCQ requests McQuitty's similarity analysis (weighted average linkage, weighted pair-group method using arithmetic averages, WPGMA).
  - MEDIAN | MED requests Gower's median method (weighted pair-group method using centroids, WPGMC). Distance data are squared unless you specify the NOSQUARE option.
  - SINGLE | SIN requests single linkage (nearest neighbor, minimum method, connectedness method, elementary linkage analysis, or dendritic method). To reduce chaining, you can use the TRIM= option with METHOD=SINGLE.
  - TWOSTAGE | TWO requests two-stage density linkage. You must also specify the K=, R=, or HYBRID option to indicate the type of density estimation to be used. See also the MODE= and DIM= options in this section.
  - WARD | WAR requests Ward's minimum-variance method (error sum of squares, trace W). Distance data are squared unless you specify the NOSQUARE option. To reduce distortion by outliers, the TRIM= option is recommended. See the NONORM option.

Supported Data Types

Hierarchical Method	Coordinate Data	Distance Data
Average Linkage	Yes	Yes
Centroid Linkage	Yes	Yes
Complete Linkage	No	Yes
Density Linkage	No	Some Options
EML	Yes	No
Flexible-Beta Method	No	Yes
McQuitty's Similarity	No	Yes
Median Linkage	No	Yes
Single Linkage	No	Yes
Two-Stage Linkage	No	Some Options
Ward's Method	Yes	Yes

Average Linkage

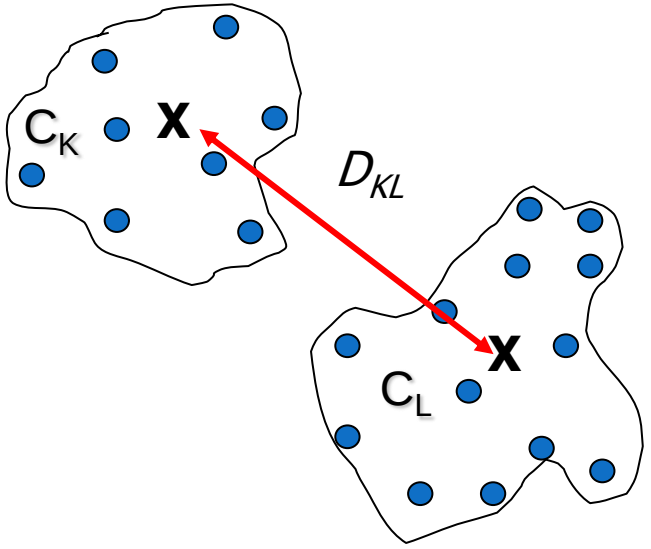
- The distance between clusters is the average distance between pairs of observations.



$$D_{KL} = \frac{1}{n_K n_L} \sum_{i \in C_K} \sum_{j \in C_L} d\left(x_i, x_j\right)$$

Centroid Linkage

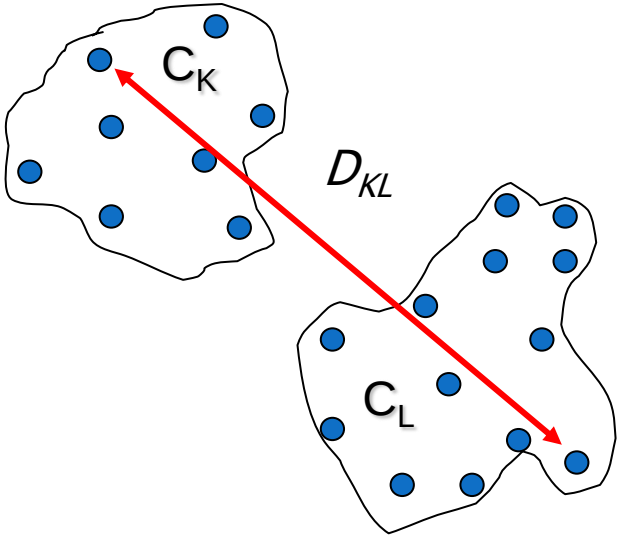
- The distance between clusters is the squared Euclidean distance between cluster centroids $\bar{\mathbf{x}}_K$ and $\bar{\mathbf{x}}_L$.



$$D_{KL} = \left\| \bar{\mathbf{x}}_K - \bar{\mathbf{x}}_L \right\|^2$$

Complete Linkage

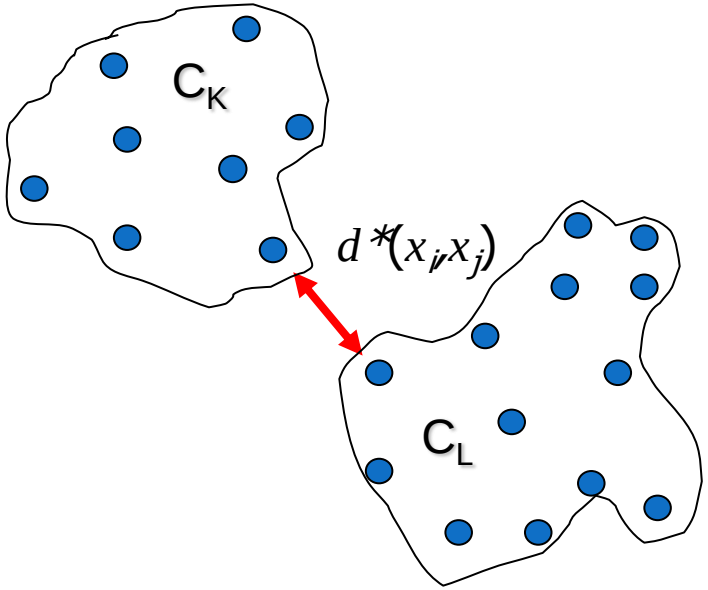
- The distance between clusters is the maximum distance between two observations, one in each cluster.



$$D_{KL} = i \in C_K \, j \in C_L \, \max \, d(x_i, x_j)$$

Density Linkage

1. Calculate a new distance metric, d^*, using k-nearest neighbor, uniform kernel, or Wong's hybrid method.
2. Perform single linkage clustering with d^*.

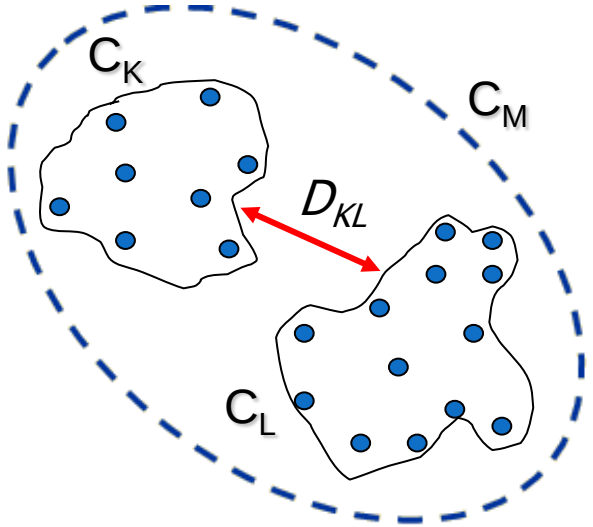


$$d^*(x_i, x_j) = \frac{1}{2}\left(\frac{1}{f(x_i)} + \frac{1}{f(x_j)}\right)$$

Equal Variance Maximum Likelihood

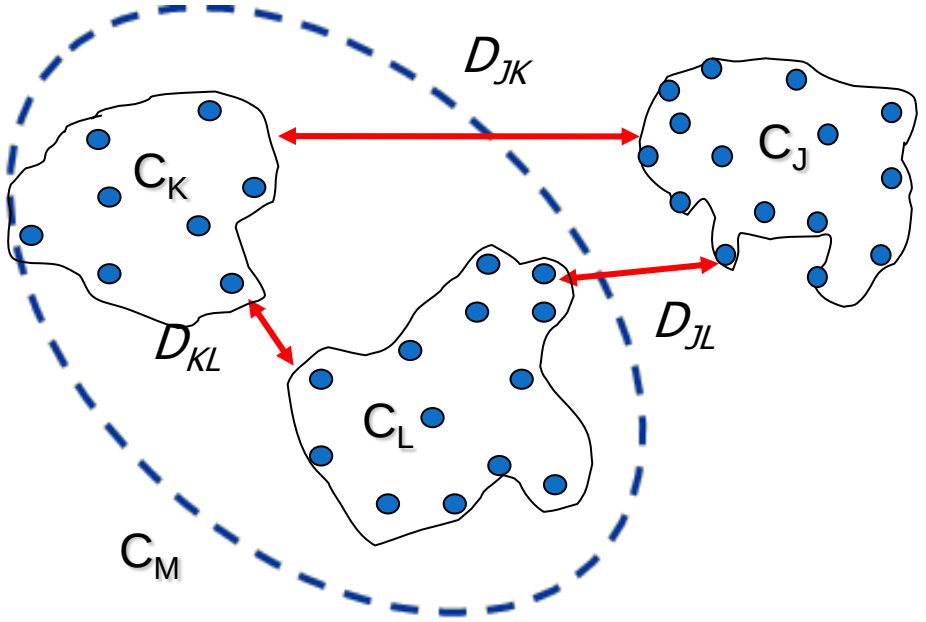
- The distance between clusters C_K and C_L is given by a penalized maximum-likelihood variant.

$$D_{KL} = n\gamma \ln\left(1 + \left[\frac{w_M - w_K - w_L}{P_G}\right]\right) - p\left(n_M \ln\left(n_M\right) - n_K \ln\left(n_K\right) - n_L \ln\left(n_L\right)\right)$$



Flexible-Beta

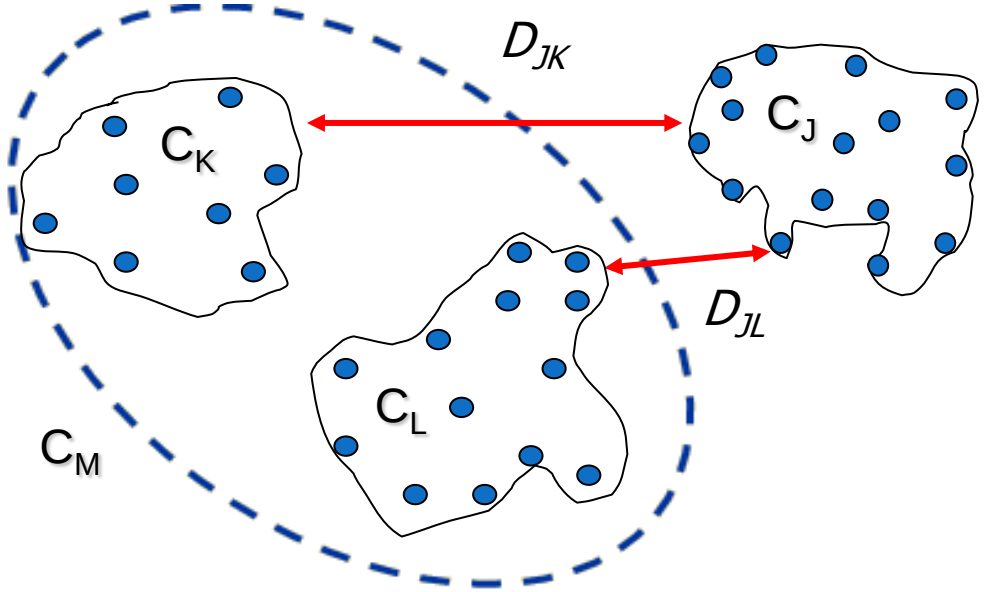
- The distance between clusters C_K and C_L is a BETA scaled measure of the component distances.



$$D_{JM} = \left(D_{JK} + D_{JL}\right)\frac{\left(1-b\right)}{2} + D_{KL}b$$

McQuitty's

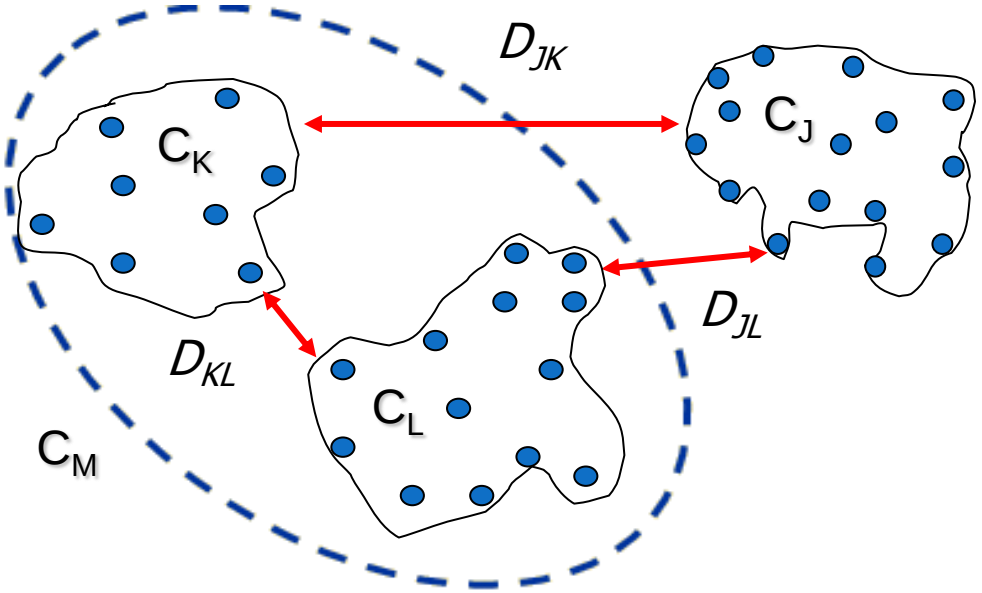
- The average distance between an external cluster, J, and each of the component clusters (C_K and C_L)



$$D_{JM} = \frac{D_{JK} + D_{JL}}{2}$$

Median Linkage

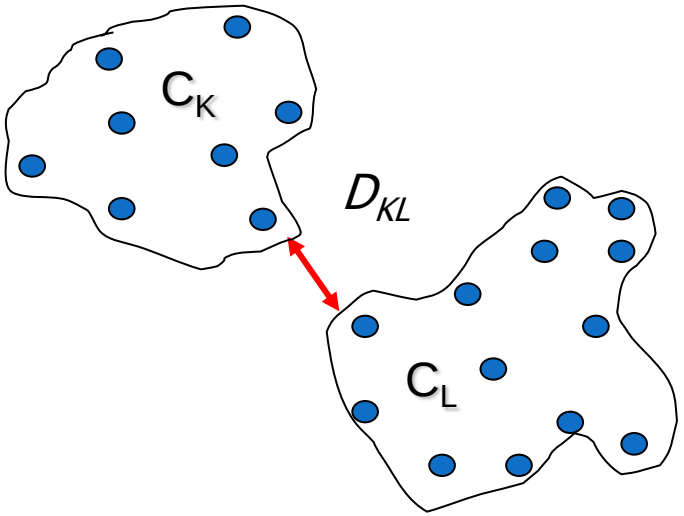
- The average distance between an external cluster and each of the component clusters, minus the distance between the component clusters.



$$D_{JM} = \frac{D_{JK} + D_{JL}}{2} - \frac{D_{KL}}{4}$$

Single Linkage

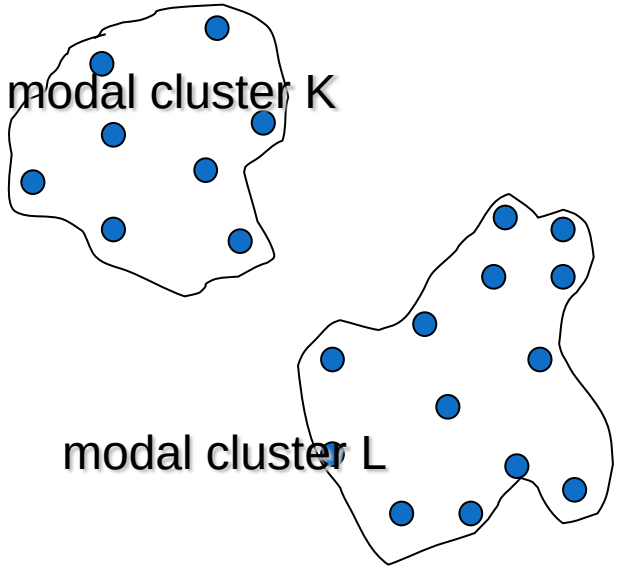
- The distance between clusters is the distance between the two nearest observations, one in each cluster.



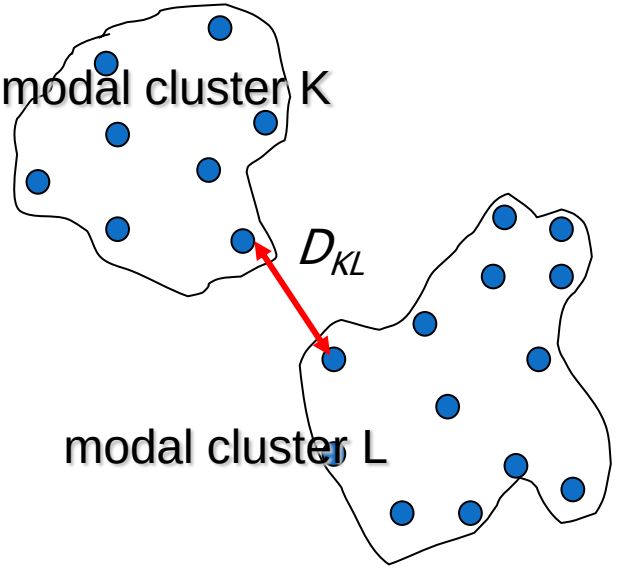
$$D_{KL} = i \in C_K \;\; j \in C_L \min d(x_i, x_j)$$

Two-Stage Density Linkage

- The same as density linkage except that a cluster must have at least "*n*" members before it can be fused.



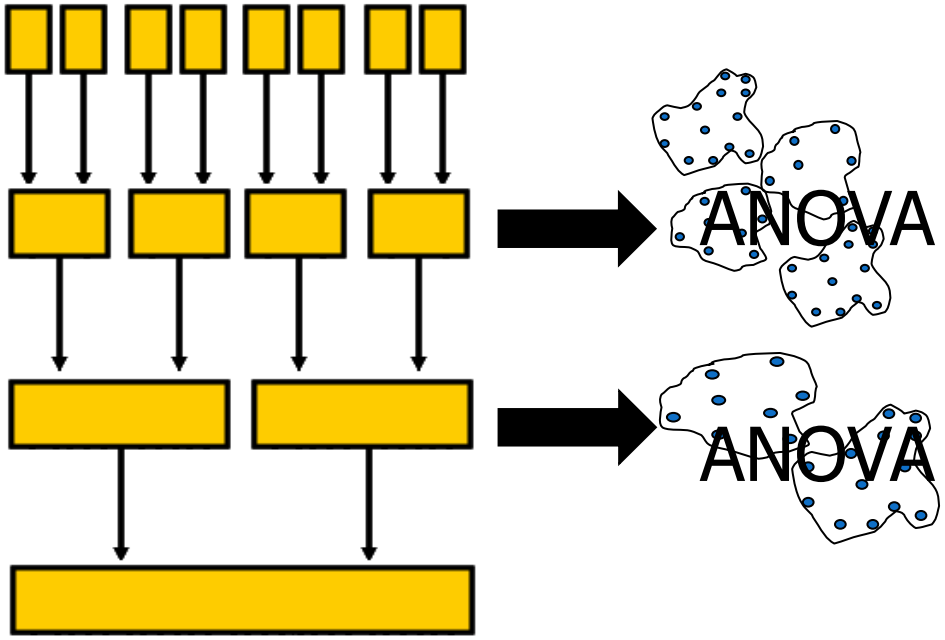
1. Form 'modal' clusters



2. Apply single linkage

Ward's

- Ward's method uses ANOVA at each fusion point to determine if the proposed fusion is warranted.



$$D_{KL} = \frac{\left\| \bar{x}_K - \bar{x}_L \right\|^2}{\left(\frac{1}{n_K} + \frac{1}{n_L} \right)}$$

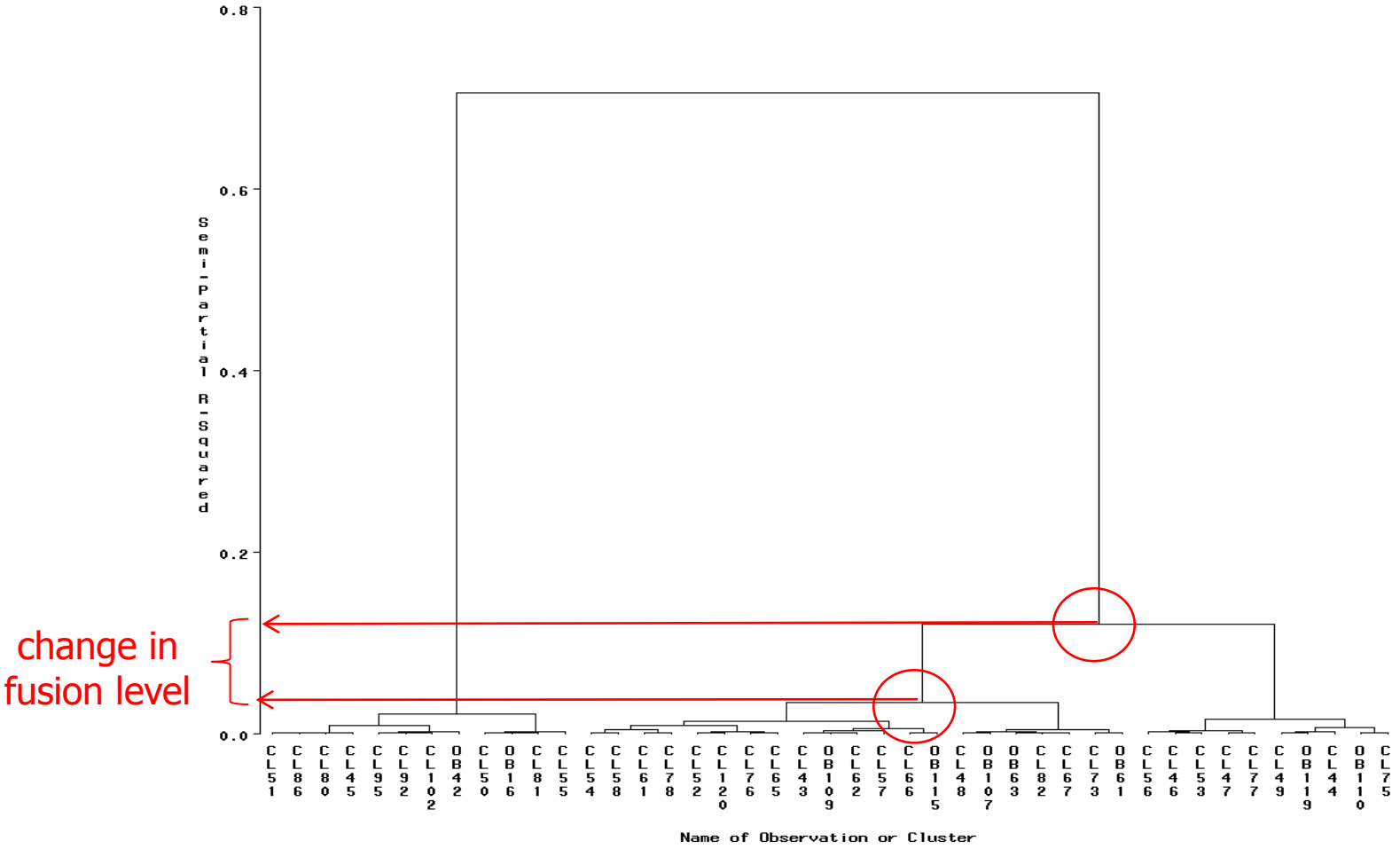
The TREE Procedure

General form of the TREE procedure:

```
PROC TREE DATA=<dendrogram>
<options>;
RUN;
```

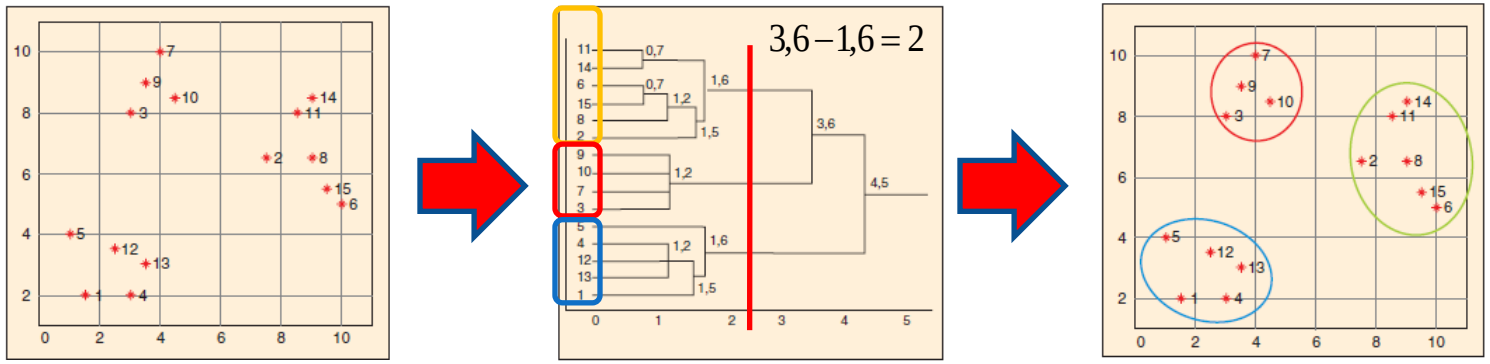
- The TREE procedure either
  - displays the dendrogram (**LEVEL=** option) or
  - assigns the observations to a specified number of clusters (**NCLUSTERS=** option).

Interpreting Dendrograms



Určení finálních shluků

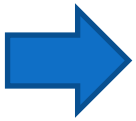
- Finální shluky získáme vhodným „řezem“ dendrogramu.
- Neexistuje univerzální postup, vždy záleží na konkrétních datech a interpretovatelnosti výsledku.
- Lze ale použít např. tento postup:
  - Označíme μ_i vzdálenosti shluků, které vznikli v průběhu shlukovacího algoritmu v okamžicích spojování objektů/shluků.
  - Spočteme $r_i = \mu_{i+1} - \mu_i$.
  - Spočteme $\max(r_i)$ a určíme tím místo, kde „říznout“.



Zdroj obrázků : L.Žák, Shluková analýza (II), http://www.volny.cz/elzet/Libor/Aut_cl_2.pdf

Cvičení

Generování dat: c_data.sas



- COMPACT : Three well-separated, compact clusters.
  Source : SAS/STAT User's Guide (Introduction to Clustering Procedures).
- DERMATOLOGY : Differential diagnosis of erythemato-squamous disease.
  Source : Nilselliter, N. and Altay Guvenir, H. (1998)
- ELONGATED : Two parallel elongated clusters in which the variation in one dimension is 6 times the variation of the other dimension. There are 150 members in each of the clusters, for a total of 300 observations.
  Source : SAS/STAT User's Guide (Introduction to Clustering Procedures).
- FISH : Seven species of fish caught off the coast of Finland.
  Source : Data Archive of the Journal of Statistics Education
- INVESTORS : ...(training data)
- OUTLIERS: Create two clusters with severe outliers.
- PIZZA : Nutrient levels of various brands of frozen pizza.
  Source: D.E. Johnson (1998), Applied Multivariate Methods for Data Analysis, Duxbury Press, Cole Publishing Company, Pacific Grove, CA. (Example 9.2)
- RING : A normal cluster surrounded by a ring cluster.
  Source : SAS/STAT User's Guide (The MODECLUS Procedure - Examples).
- STOCK : Dividend yields for 15 utility stocks in the U.S. for 1986-1990.
  Source : SAS/STAT User's Guide (The DISTANCE Procedure - Examples).
- TINVESTORS : Investors data set (test data)
- UNEQUAL : Generate three unequal variance and unqual size clusters.
  Source : SAS/STAT User's Guide.

Cvičení

```
/*
   clus01d01: Generating distances.

   The sasuser.stock data set contains the dividend yields for 15 utility stocks in the
U.S.
   The observations are names of the companies, and the variables correspond to
the annual
   dividend yields over the period 1986-1990.
*/

options nodate nonumber;
goptions reset=all;

%let inputs = div_1986 div_1987 div_1988 div_1989 div_1990;

/* display the input data set */
title 'Stock Dividends';
title2 'The STOCK Data Set';
proc print data=sasuser.stock;
                var company &inputs;
run;

/* calculate the range standardized Euclidean distance */
proc distance data=sasuser.stock method=euclid out=dist;
                var interval(&inputs/std=range);
   id company;
run;

/* display the distance matrix generated */
title2 'Euclidean Distance Matrix';
proc print data=dist;
                id company;
run;
```

Stock Dividends
The STOCK Data Set

Obs	company	div_1986	div_1987	div_1988	div_1989	div_1990
1	Cincinnati G&E	8.4	8.2	8.4	8.1	8.0
2	Texas Utilities	7.9	8.9	10.4	8.9	8.3
3	Detroit Edison	9.7	10.7	11.4	7.8	6.5
4	Orange & Rockland Utilitie	6.5	7.2	7.3	7.7	7.9
5	Kentucky Utilities	6.5	6.9	7.0	7.2	7.5
6	Kansas Power & Light	5.9	6.4	6.9	7.4	8.0
7	Union Electric	7.1	7.5	8.4	7.8	7.7
8	Dominion Resources	6.7	6.9	7.0	7.0	7.4
9	Allegheny Power	6.7	7.3	7.8	7.9	8.3
10	Minnesota Power & Light	5.6	6.1	7.2	7.0	7.5
11	Iowa-Ill Gas & Electric	7.1	7.5	8.5	7.8	8.0
12	Pennsylvania Power & Light	7.2	7.6	7.7	7.4	7.1
13	Oklahoma Gas & Electric	6.1	6.7	7.4	6.7	6.8
14	Wisconsin Energy	5.1	5.7	6.0	5.7	5.9
15	Green Mountain Power	7.1	7.4	7.8	7.8	8.3

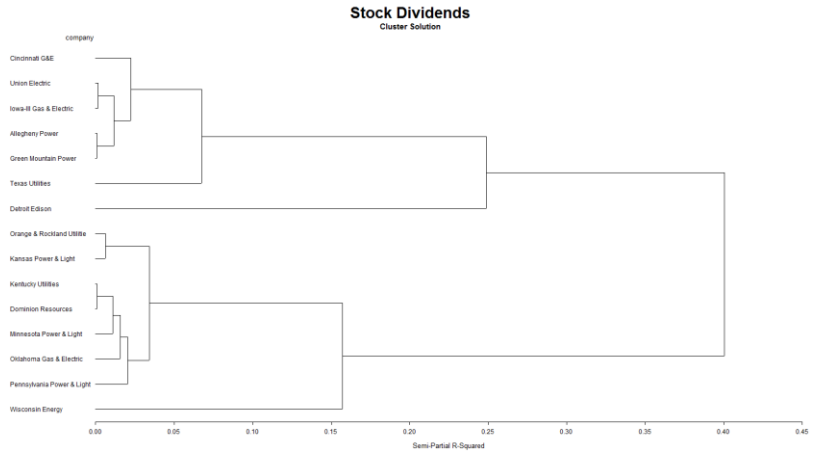
Stock Dividends
Euclidean Distance Matrix

company	Cincinnati_ G_E	Texas_ Utilities	Detroit_ Edison	Orange___ Rockland_ Utilitie	Kentucky_ Utilities	Kansas_ Power___ Light	Union_ Electric	Dominion_ Resources
Cincinnati G&E	0.00000	.	.	.	.	.	.	.
Texas Utilities	0.49670	0.00000	.	.	.	.	.	.
Detroit Edison	1.01879	0.99884	0.00000	.	.	.	.	.
Orange & Rockland Utilitie	0.51910	0.84035	1.37538	0.00000	.	.	.	.
Kentucky Utilities	0.65416	1.02098	1.39076	0.24265	0.00000	.	.	.
Kansas Power & Light	0.74161	1.04618	1.58831	0.24213	0.27325	0.00000	.	.
Union Electric	0.35197	0.65306	1.13477	0.26463	0.37514	0.47420	0.00000	.
Dominion Resources	0.67235	1.05719	1.36747	0.31596	0.08679	0.34455	0.40903	0.00000
Allegheny Power	0.44817	0.70706	1.37683	0.20627	0.43497	0.36120	0.29152	0.49807
Minnesota Power & Light	0.87055	1.17406	1.57540	0.40330	0.26298	0.26451	0.55096	0.29307
Iowa-Ill Gas & Electric	0.32954	0.60517	1.18664	0.26965	0.43259	0.46890	0.12636	0.47342
Pennsylvania Power & Light	0.53641	0.90027	1.10787	0.39363	0.30207	0.54776	0.30952	0.28190
Oklahoma Gas & Electric	0.90316	1.23233	1.39103	0.57064	0.35232	0.55849	0.60493	0.30885
Wisconsin Energy	1.51338	1.85581	1.86900	1.15137	0.92124	1.06088	1.22855	0.87666
Green Mountain Power	0.37713	0.68574	1.32741	0.23652	0.44185	0.40876	0.27431	0.49258

company	Allegheny_ Power	Minnesota_ Power___ Light	Iowa_Ill_ Gas___ Electric	Pennsylvania_ Power___Light	Oklahoma_ Gas___ Electric	Wisconsin_ Energy	Green_ Mountain_ Power
Cincinnati G&E	.	.	.	.	.	.	.
Texas Utilities	.	.	.	.	.	.	.
Detroit Edison	.	.	.	.	.	.	.
Orange & Rockland Utilitie	.	.	.	.	.	.	.
Kentucky Utilities	.	.	.	.	.	.	.
Kansas Power & Light	.	.	.	.	.	.	.
Union Electric	.	.	.	.	.	.	.
Dominion Resources	.	.	.	.	.	.	.
Allegheny Power	0.00000	.	.	.	.	.	.
Minnesota Power & Light	0.56333	0.00000	.	.	.	.	.
Iowa-Ill Gas & Electric	0.20632	0.59042	0.00000	.	.	.	.
Pennsylvania Power & Light	0.53868	0.51280	0.42317	0.00000	.	.	.
Oklahoma Gas & Electric	0.75376	0.34849	0.69464	0.39515	0.00000	.	.
Wisconsin Energy	1.34430	0.82285	1.31501	0.99204	0.62670	0.00000	.
Green Mountain Power	0.09454	0.59991	0.18119	0.51773	0.76232	1.35883	0

Cvičení

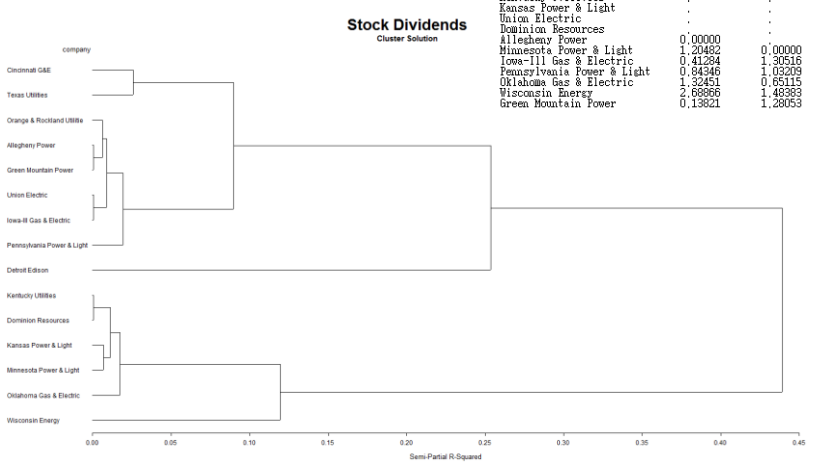
```
/* generate hierarchical clustering solution (Ward's method)*/
proc cluster data=dist method=ward outtree=tree noprint;
                id company;
run;

/* display the EUCLID dendrogram  horizontally */
title2 "Cluster Solution";
proc tree data=tree horizontal;
                id company;
run;

/* calculate the range standardized city block distance */
proc distance data=sasuser.stock method=cityblock out=dist;
                var interval(&inputs/std=range);
   id company;
run;

/* display the distance matrix generated */
title2 'City Block Distance Matrix';
proc print data=dist;
                id company;
run;

/* generate hierarchical clustering solution (Ward's method)*/
proc cluster data=dist method=ward outtree=tree noprint;
                id company;
run;

/* display the CITYBLOCK dendrogram horizontally */
title2 "Cluster Solution";
proc tree data=tree horizontal;
                id company;
run;
```



Stock Dividends
City Block Distance Matrix

company	Cincinnati_G_E	Texas_Utilities	Detroit_Edison	Orange_Rockland_Utilitie	Kentucky_Utilities	Kansas_Power_Light	Union_Electric	Dominion_Resources
Cincinnati G&E	0.00000	.	.	.	.	.	.	.
Texas Utilities	0.99407	0.00000	.	.	.	.	.	.
Detroit Edison	2.05691	2.03024	0.00000	.	.	.	.	.
Orange & Rockland Utilitie	0.98341	1.76009	2.76949	0.00000	.	.	.	.
Kentucky Utilities	1.42189	2.19856	2.87463	0.43847	0.00000	.	.	.
Kansas Power & Light	1.40001	2.17668	3.26942	0.49993	0.51979	0.00000	.	.
Union Electric	0.64136	1.41803	2.26077	0.50872	0.78053	1.00865	0.00000	.
Dominion Resources	1.46257	2.25525	2.85199	0.58612	0.14764	0.66743	0.84122	0.00000
Allegheny Power	0.84818	1.37485	2.78009	0.38524	0.82371	0.80183	0.51932	0.88440
Minnesota Power & Light	1.80300	2.57968	3.25575	0.81959	0.45519	0.51411	1.16164	0.47783
Iowa-Ill Gas & Electric	0.53488	1.27451	2.36725	0.48557	0.92405	0.90217	0.14352	0.96473
Pennsylvania Power & Light	1.10425	1.88032	2.22366	0.73233	0.65097	1.04576	0.54637	0.62833
Oklahoma Gas & Electric	1.92269	2.69096	2.73210	0.97631	0.64895	0.91482	1.28133	0.58626
Wisconsin Energy	3.28684	4.06351	3.90625	2.30342	1.86495	1.88683	2.64548	1.80426
Green Mountain Power	0.77247	1.29914	2.64188	0.46094	0.89942	0.87754	0.38111	0.96010

company	Allegheny_Power	Minnesota_Power_Light	Iowa_Ill_Gas_Electric	Pennsylvania_Power_Light	Oklahoma_Gas_Electric	Wisconsin_Energy	Green_Mountain_Power
Cincinnati G&E	.	.	.	.	.	.	.
Texas Utilities	.	.	.	.	.	.	.
Detroit Edison	.	.	.	.	.	.	.
Orange & Rockland Utilitie	.	.	.	.	.	.	.
Kentucky Utilities	.	.	.	.	.	.	.
Kansas Power & Light	.	.	.	.	.	.	.
Union Electric	.	.	.	.	.	.	.
Dominion Resources	.	.	.	.	.	.	.
Allegheny Power	0.00000	.	.	.	.	.	.
Minnesota Power & Light	1.20482	0.00000	.	.	.	.	.
Iowa-Ill Gas & Electric	0.41284	1.30516	0.00000	.	.	.	.
Pennsylvania Power & Light	0.84346	1.03209	0.68989	0.00000	.	.	.
Oklahoma Gas & Electric	1.32451	0.65115	1.42485	0.81844	0.00000	.	.
Wisconsin Energy	2.68866	1.48383	2.78900	2.18259	1.36415	0.00000	.
Green Mountain Power	0.13821	1.28053	0.27463	0.70526	1.40022	2.76437	0



Cvičení

/* clus02d4: Impact of input standardization on clustering.

This demonstration evaluates the impact on cluster performance of changing the method of input standardization. Several methods are ranked according to their Cramer's V value and their misclassification rate. PROC FASTCLUS is used to cluster the observations. The input data set is the pizza data set. The input variables are the three inputs recommended using by the PROC VARCLUS 1-R**2 criterion.
*/

options nodate nonumber;

%let group = brand;
%let inputs = carb mois sodium;

data results;
length method$ **12;**
length misclassified **8;**
length chisq **8;**
length pchisq **8;**
length cramersv **8;**
stop;
run;

%macro standardize(dsn=, nc=, method=);
...
%mend standardize;

%***standardize(dsn=sasuser.pizza,nc=10,method=ABW(11));***
%***standardize(dsn=sasuser.pizza,nc=10,method=AGK(1));***

Method: ABW(11)

The FREQ Procedure

Table of brand by CLUSTER

brand CLUSTER(Cluster)

Frequency	1	2	3	4	5	6	7	8	9	10	Total
A	0	14	0	0	15	0	0	0	0	0	29
B	15	0	0	0	0	0	16	0	0	0	31
C	0	0	0	0	0	0	0	16	0	11	27
D	5	0	0	0	0	0	0	1	0	26	32
E	0	0	0	0	0	28	0	0	0	0	28
F	0	0	0	30	0	0	0	0	0	0	30
G	0	0	0	29	0	0	0	0	0	0	29
H	0	0	0	0	0	33	0	0	0	0	33
I	0	0	0	0	0	0	0	0	29	0	29
J	0	0	32	0	0	0	0	0	0	0	32
Total	20	14	32	59	15	61	16	17	29	37	300

Statistics for Table of brand by CLUSTER

Statistic	DF	Value	Prob
Chi-Square	81	1850.9203	<.0001
Likelihood Ratio Chi-Square	81	1139.3575	<.0001
Mantel-Haenszel Chi-Square	1	0.0318	0.8584
Phi Coefficient		2.4939	
Contingency Coefficient		0.9276	
Cramer's V		0.8280	

WARNING: 80% of the cells have expected counts less than 5. Chi-Square may not be a valid test.

Sample Size = 300

Method: AGK(1)

The FREQ Procedure

Table of brand by CLUSTER

brand CLUSTER(Cluster)

Frequency	1	2	3	4	5	6	7	8	9	10	Total
A	0	29	0	0	0	0	0	0	0	0	29
B	8	0	0	0	0	23	0	0	0	0	31
C	1	0	0	0	0	0	0	5	0	21	27
D	2	0	0	0	0	3	0	26	0	1	32
E	0	0	9	0	19	0	0	0	0	0	28
F	0	0	0	0	0	0	30	0	0	0	30
G	0	0	0	0	1	0	28	0	0	0	29
H	0	0	4	0	29	0	0	0	0	0	33
I	0	0	0	29	0	0	0	0	0	0	29
J	0	0	0	0	0	0	0	0	32	0	32
Total	11	29	13	29	49	26	58	31	32	22	300

Statistics for Table of brand by CLUSTER

Statistic	DF	Value	Prob
Chi-Square	81	1905.5468	<.0001
Likelihood Ratio Chi-Square	81	1138.9933	<.0001
Mantel-Haenszel Chi-Square	1	21.5931	<.0001
Phi Coefficient		2.5203	
Contingency Coefficient		0.9295	
Cramer's V		0.8401	

WARNING: 86% of the cells have expected counts less than 5. Chi-Square may not be a valid test.

Sample Size = 300

Cvičení

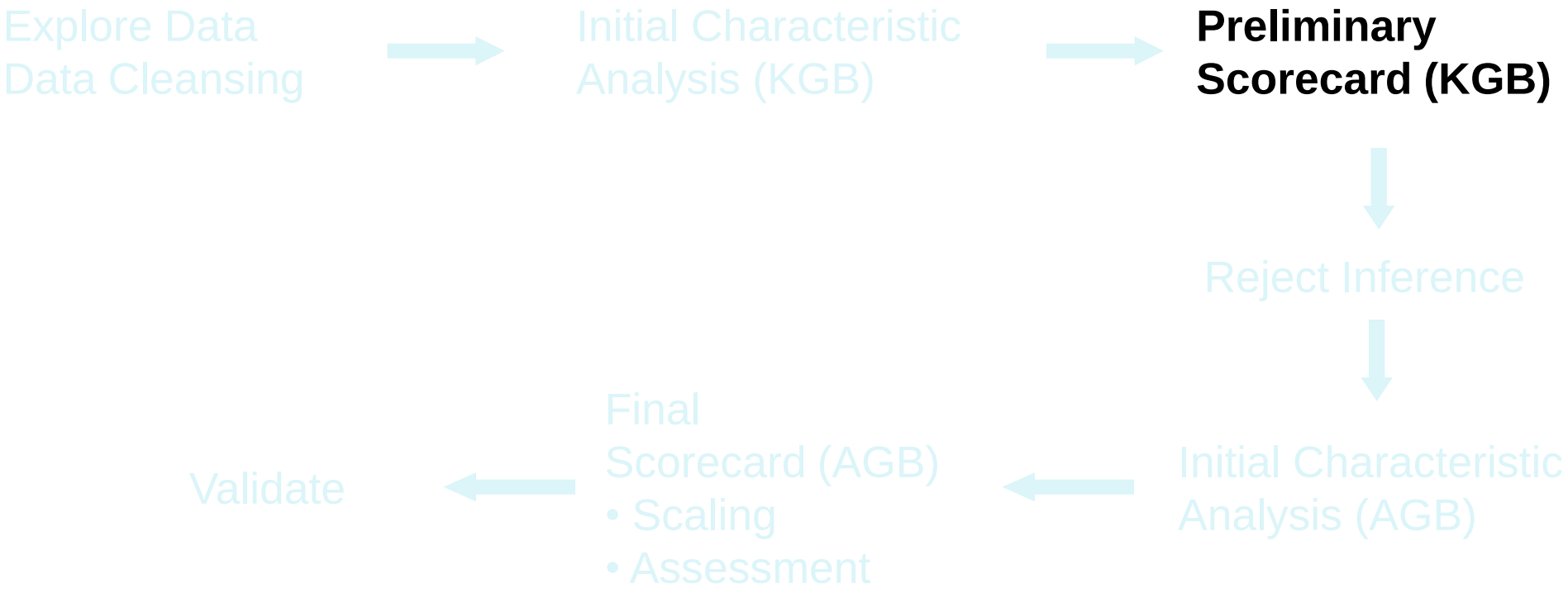
```
%standardize(dsn=sasuser.pizza,nc=10,method=AHUBER(.1));
%standardize(dsn=sasuser.pizza,nc=10,method=AWAVE(.2));
%standardize(dsn=sasuser.pizza,nc=10,method=EUCLEN);
%standardize(dsn=sasuser.pizza,nc=10,method=IQR);
%standardize(dsn=sasuser.pizza,nc=10,method=L(1));
%standardize(dsn=sasuser.pizza,nc=10,method=L(1.5));
%standardize(dsn=sasuser.pizza,nc=10,method=L(2));
%standardize(dsn=sasuser.pizza,nc=10,method=MAD);
%standardize(dsn=sasuser.pizza,nc=10,method=MAXABS);
%standardize(dsn=sasuser.pizza,nc=10,method=MEAN);
%standardize(dsn=sasuser.pizza,nc=10,method=MEDIAN);
%standardize(dsn=sasuser.pizza,nc=10,method=MIDRANGE);
%standardize(dsn=sasuser.pizza,nc=10,method=NONE);
%standardize(dsn=sasuser.pizza,nc=10,method=RANGE);
%standardize(dsn=sasuser.pizza,nc=10,method=SPACING(.9));
%standardize(dsn=sasuser.pizza,nc=10,method=STD);
%standardize(dsn=sasuser.pizza,nc=10,method=SUM);
%standardize(dsn=sasuser.pizza,nc=10,method=USTD);

/* sort by number of misclassifications within Cramer's V */
proc sort data=results;
        by descending cramersv misclassified;
run;

/* display Cramer's V and misclassifications for each method */
title1 'Results';
proc print data=results;
        var method cramersv misclassified ;
run;

quit;
```

Results

Obs	method	cramersv	misclassified
1	SPACING(.9)	0.85070	30
2	AWAVE(.2)	0.84716	45
3	L(1)	0.84484	32
4	AGK(1)	0.84009	34
5	L(2)	0.84009	34
6	STD	0.84009	34
7	MIDRANGE	0.84003	48
8	RANGE	0.84003	48
9	MAXABS	0.83122	50
10	L(1.5)	0.82816	43
11	ABW(11)	0.82796	46
12	EUCLEN	0.81871	60
13	USTD	0.81871	60
14	SUM	0.81076	63
15	MEAN	0.78492	72
16	MEDIAN	0.78492	72
17	NONE	0.78492	72
18	IQR	0.77553	50
19	AHUBER(.1)	0.76885	58
20	MAD	0.70940	79

8. Vývoj CS modelu

Process Flow



Preliminary Scorecard (Known Good/Bad)

- Group of characteristics, that together, offer the most predictive power
- Logistic Regression (forward, backward, stepwise)
- 8–20 characteristics
  - stability.

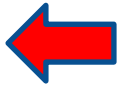
Logistic Regression

$$\text{Logit}\,(p_i)=\beta_o+\beta_1 x_1+\ldots.+\beta_k x_k$$

p – posterior probability of 'event' given inputs
x – input variables
β – parameters

- Logit transformation is log of the odds, and is used to linearize posterior probability and limit outcome to between 0 and 1.
- Maximum Likelihood used to estimate parameters.
- Parameters estimates measure rate of change of logit for one unit change in input variable (adjusted for other inputs)
  - Depends on the unit of the input, therefore need to standardise (e.g. WOE)

Logistic Regression

- Binary target (good/bad)
- Variables
  - Raw data
  - Grouped data (for example, mid value of each group)
  - Weight of evidence 

Logistic Regression

- Forward Stepwise
  - Select best variable, add it to the model, and then add/subtract variables until no improvement in indicator.
  - Efficient, but weak when too many variables or high correlation
- Backward Elimination
  - Start with all variables in the model, then eliminate least important variables.
  - Correlation is better taken care of
  - Better than stepwise, but can be computationally intensive.

Preliminary Scorecard

- Choose the best – and build the most comprehensive **risk profile**
- With as many independent data items as possible
  - independent data items representing different data types e.g. demog, financials, inquiries, trades info
- 10 characteristics with '100' each preferred to 4 with '250' each.
- Correlation, co linearity etc. considered
- Scorecard coherent with decision support structure
  - **Sole arbiter or decision support tool: model needs to be coherent with overall decision support structure**
- Interpretability, implementability, and other business considerations.

Example of a Good Scorecard

- Age
- Residential status
- Time at address
- Inquiries 12 months [1)]
- Inquiries 3 months
- Trades 90 days+ as % of total
- Revolving balance/Total
- Utilization
- Number of products at bank
- Delinquency at bank
- Total Debt Service Ratio

➢ contains some demographics, some inquiries, some trade, some utilization, internal bank perforamnce and capacity to pay.

[1)] Počet žádostí o úvěr za posledních 12 měsíců

How Do We Get There?

- Try statistically optimal approach (let the data speak)
- "Design" a scorecard using stepwise/backward
  - Force characteristics in, or fix at each loop and adjust the hurdle rate
  - Consider:
    - "must have"
    - Weaker/stronger
    - Similar

Weaker, Similar

- Weaker – consider first
  - Can 2 characteristics worth 40 points each model behavior better than one worth 70?
  - Same strength, broader base
- Similar – put together
  - Time related, inquiries, trades, debt capacity, demographic
  - Takes care of correlation

Putting It Together

- Try different combinations of characteristics in regression
  - Instead of putting all characts in, separate into categories, and try combinations.
- Leave very strong characteristics out, or use at the end (for example, bureau scores)
- Example "levels"
  - Weaker application info
  - Stronger application info
  - Weaker bureau
  - Stronger bureau
- Mix and adjust with experience.

Putting It Together

- Age, time at address, time at employment, time at bank
- Region, postal code, province
- TDSR, GDSR, capacity, Loan To Value
- Time at bureau, current customer (Y/N)
- Inq 3 months, inq 6 months, inq 12 months, inq 3/12 months
- Trades delq, trades 3 mth/total, current trades
- Utilization, public records
- Bureau score, bankruptcy.

GDSR(Gross Debt Service Ratio) = (Annual Mortgage Payments + Property Taxes + Other Shelter Costs)/(Gross Family Income)
TDSR (Total Debt Service Ratio) = (Annual Mortgage Payments + Property Taxes + Other Shelter Costs + Other Debt Payments)/(Gross Family Income)

Logistic Regression

- Use stepwise or backward
  - stepwise means dominating variable will stay in.
  - **Backward:** set of weak variables may end up staying.. That together add value (sometimes better than stepwise).. Also backward takes care of correlation better than others.

- Modify to consider only selected characteristics at each “level”
  - series of regression runs, each as one “level”, force selected characteristics from previous “levels” in.
  - EM nodes in series.

Logistic Regression

- It is strongly recommended that all coefficients are logical. If some of them are not, include comments with explanation why it is good to keep them in scorecard. Include column, where it is easy to see the contribution of each category (either scaled scorepoints for linearization, either simply bi*xi*1000). Order categories in each predictor according to badrate (WOE) so that the worst are the first.

LR – scorecard example

Own / Rent		Rent	Other	NI				
		15	10	17				
Years at address	<5	.5-2.49	2.5-6.49	6.5-10.49	>10.49	NI		
	12	10	15	19	23	14		
Occupation	Prof	SemiPrf	Mgr	Offc.	Bl.Col	Retired	Other	NI
	50	44	31	28	25	31	22	27
Years on job	<.5	.5-1.49	1.5-2.49	2.5-5.49	5.5-12.49	12.5	Retired	NI
	2	8	19	25	30	39	43	20
Dept St / Major CC	None	Dept-St	Maj-CC	Both	No answr	NI		
	0	11	16	27	10	12		
Bank reference	Check	Sav	Ck&Sav	Other	NI			
	5	10	20	11	9			
Debt ratio	<15	15-25	26-35	36-49	50+	NI		
	22	15	12	5	0	13		
No. of recent inquiries	0	1	2	3	4	5-9	No Rcrd	
	3	11	3	-7	-7	-20	0	
Years in file	<.5	1-2	3-4	5-7	8+			
	0	5	15	30	40			
# Rev trades outstanding	0	1-2	3-5	6+				
	5	12	8	-4				
% Credit line utilization	0-15%	16-30%	31-40%	41-50%	>50%			
	15	5	-3	-10	-18			
Worst reference								

LR – scorecard example

Obs	colnamew	colvalue	Bi_x_Xi_x_1000	bad_rate	freq_rate	Bi	Xi
1	Intercept		3129	.	.	3.129451	.
3	age_fr_w	20	-359	10.3	3.5	0.377612	-0.950967
4	age_fr_w	29	-154	6.3	28	0.377612	-0.408556
5	age_fr_w	32	-47	4.8	8.4	0.377612	-0.123244
6	age_fr_w	36	20	4.1	10	0.377612	0.05253
7	age_fr_w	41	48	3.8	11	0.377612	0.12723
8	age_fr_w	51	173	2.7	23	0.377612	0.458154
9	age_fr_w	60	327	1.8	16	0.377612	0.865979
10	car_owner_fr_w	0	-60	4.5	76	1.179055	-0.051044
11	car_owner_fr_w	1	211	3.6	24	1.179055	0.179078
12	child_num_fr_w	99	-59	4.9	1.9	0.424104	-0.138759
13	child_num_fr_w	0	-33	4.6	60	0.424104	-0.078474
14	child_num_fr_w	1	34	4	27	0.424104	0.080184
15	child_num_fr_w	2	134	3.2	11	0.424104	0.315117
16	education_fr_w	5	-174	6.1	4.3	0.453663	-0.384073
17	education_fr_w	4	-79	5	3.3	0.453663	-0.174838
18	education_fr_w	2	-10	4.4	73	0.453663	-0.021929
19	education_fr_w	36	112	3.4	19	0.453663	0.247396

Logistic Regression

- For all predictors in scorecard estimated coefficient, wald chi-square, p-value (e.g. in SAS output Analysis of maximum likelihood estimate), summary of predictor selections (order of predictors entering the model, e.g. in SAS output summary of stepwise selection)

Analysis of Maximum Likelihood Estimates

Parameter	DF	Estimate	Standard Error	Wald Chi-Square	Pr > ChiSq
Intercept	1	3.1295	0.0106	86436.032	<.0001
age_fr_w	1	0.3776	0.0254	221.2018	<.0001
car_owner_fr_w	1	1.1791	0.1093	116.2664	<.0001
education_fr_w	1	0.4537	0.066	47.2419	<.0001
fam_state_fr_w	1	0.4167	0.0297	196.864	<.0001
goods_group_fr_w	1	0.2716	0.0311	76.3139	<.0001
child_num_fr_w	1	0.4241	0.0867	23.9081	<.0001
ident_card_age_fr_w	1	0.5677	0.0269	446.5471	<.0001

Summary of predictor selection

Summary of Stepwise Selection							
Step	Effect Entered	Removed	DF	Number In	Score Chi-Square	Wald Chi-Square	Pr > ChiSq
1	price1_fr_w		1	1	3858.056		<.0001
2	age_fr_w		1	2	3007.7932		<.0001
3	init_pay_by_price1_f		1	3	1434.1863		<.0001
4	ident_card_age_fr_w		1	4	868.4661		<.0001
5	type_suite_fr_w		1	5	800.2294		<.0001
6	time_on_job_fr_w		1	6	554.571		<.0001
7	mobile_phone_fr_w		1	7	342.7936		<.0001
8	sex_fr_w		1	8	357.5026		<.0001
9	fam_state_fr_w		1	9	331.6358		<.0001
10	ident_type2_fr_w		1	10	358.4905		<.0001
11	weekend_fr_w		1	11	323.0631		<.0001
12	region_fr_w		1	12	299.3854		<.0001

Good Scorecard?

- Eye Ball Test
  - Point allocation logical, no flips (after scaling)
    - “Flips” occur for several reasons: low count, correlation.
  - Scorecard characteristics make sense
    - what went in, what did’t., does it cover all the major categories of information?
- Misclassification
- Strength
- Validation.

Scorecard Development

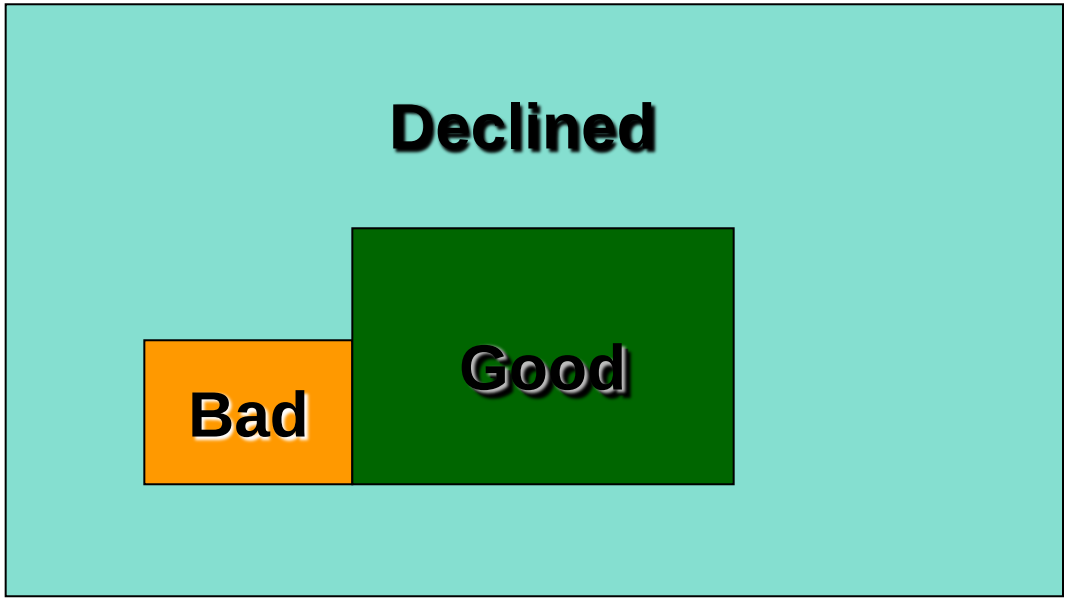
- Build more complex models and compare predictiveness – if difference not significant, then scorecard is OK
- Examine findings – is there a valid business reason?
- Build several ‘different’ scorecards

What Have I Just Done?

- "Designed" a scorecard
  - Used regression, with business considerations
  - Stable, represents strong major/independent information categories
  - Measurable strength and impact
  - Something a risk manager can buy and use.
- Used only known goods and bads (that is, approves)
- But need to apply scorecard on all applicants.

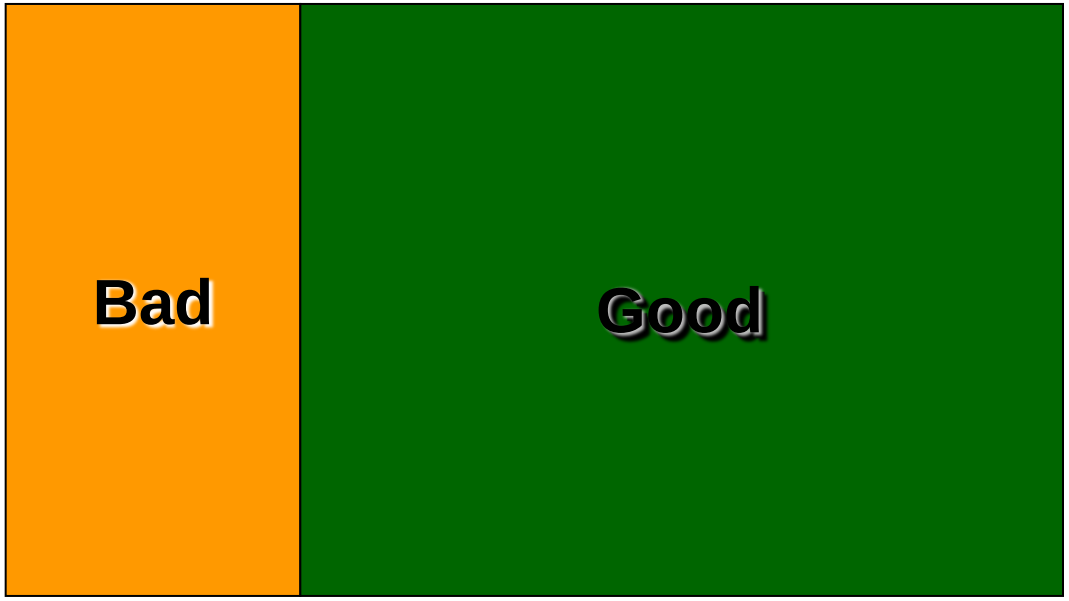
Reject Inference

- Everything to this point has been for known performance - e.g. approval rate is 60%, building a model for 100% of the population based on 60% sample is not accurate.



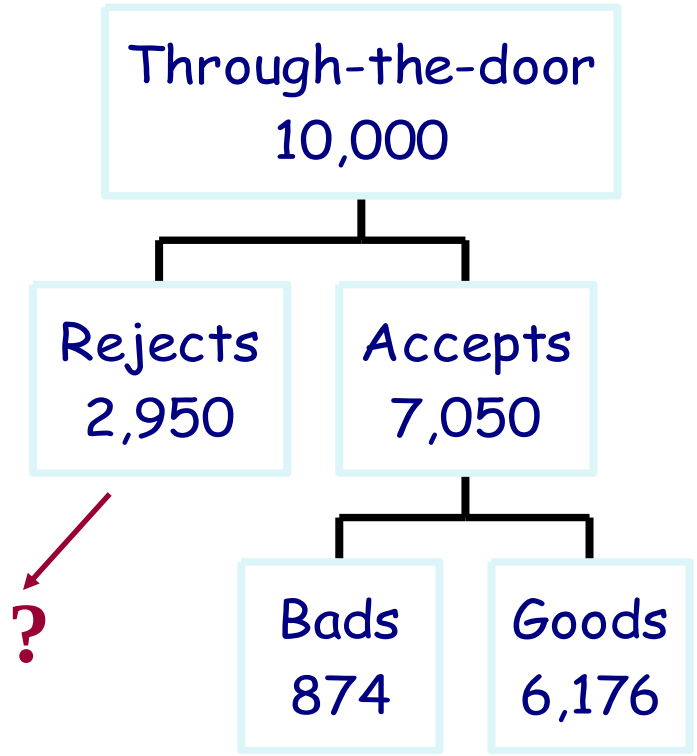
Reject Inference

- This is where you need to get to: so need to create a sample representative of the "through the door" or entire applicant pop performance - 100% approval rate.



- Inferring the behavior of declined applicants

The Known Good Bad Picture



Reject Inference

- Make the scorecard relevant
  - ignoring rejects distorts model
  - Influence of past decision making
- For decision making
  - Get population odds
  - Expected performance
  - Swap set.

		Old scorecard	
		Approve	Decline
New	Approve	A	B
Scorecard	Decline	C	D

A – is approved goods
B – is rejected goods
C – is approved bads
D – is rejected bads

Where?

- Medium/low approval rates
  - a 95% approval rate is close to “through the door”
- Manual adjudication environment
  - Incorporates experience/intuition based overriding
  - “cherry picking” distorts performance.

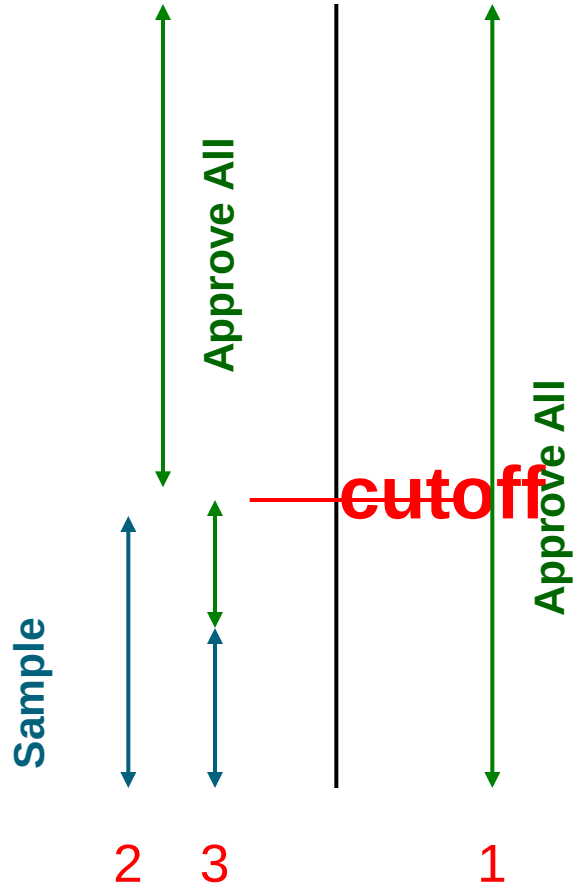
Reject Inference Techniques

- “True” Performance
- “Nearly True” Performance
- Statistical Inference
- Or ignore the problem
  - Assume accepts = total population
  - not recommended unless previous credit granting was random or scorecard was perfect (assume all rejects = bad).

“True” Performance

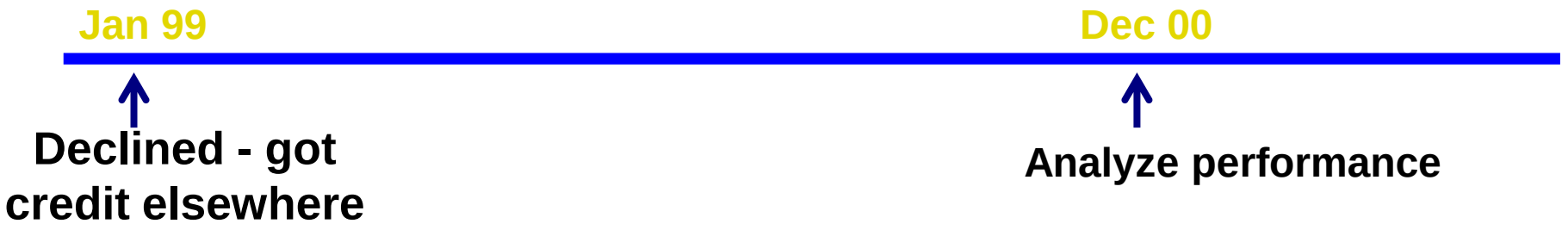
- Approve every applicant
- Or random sample
- Expensive

… but the only true way to determine performance of below cutoff applicants.



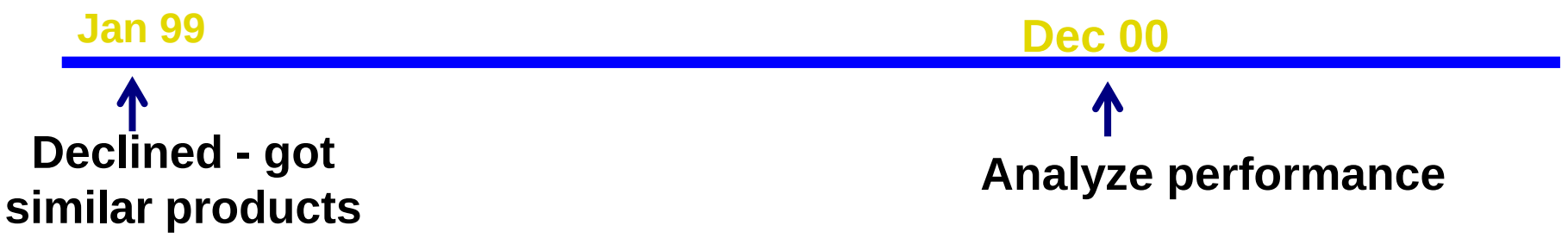
“Nearly True” Performance 1

- Bureau data
  - performance of declined apps on similar products with other companies
  - **legal issues**
  - difficult to implement in practice – timings, definitions, programming
    - Need consent to get bureau at any time
    - data - if u rejected them, they probably were rejected elsewhere
    - timings - performance window, sample window must be consistent
    - bad definition must be closely replicated
    - product must be similar - credit cards, unsecured line of credit with similar limit and conditions as you would have given
    - Experience - Programming effort is tremendous, depending on how detailed credit bureau reports are



“Nearly True” Performance 2

- In-house data
  - performance of declined apps on similar products, for example, credit cards/line of credit
  - timings, definitions my cause problems.
    - data - if u rejected them for a lower level product, they probably were rejected for higher one .. HOWEVER, in multiple product environments, scorecards are not always aligned and there is “ARBITRAGE”.
    - timings - performance window, sample window must be consistent
    - bad definition must be closely replicated
    - product must be similar - credit cards, unsecured line of credit with similar limit and conditions as you would have given



Bureau Score Migration

- Analyze bureau score migration of existing accounts with below cutoff scores
- Identify accounts whose scores migrate to ‘above cutoff’ within specified time frame

Reclassification

- Build an accept/reject model
- Score all rejects and designate worst as accepted ‘bad’
- Can use score or “serious derogatory” information to select accounts
- Analyze Accept/Reject vs. Good/Bad cross tabs
- Add to accepts and Re-model

Simple Augmentation

- Simple Augmentation
  - Build good/bad model
  - Score rejects – establish a p(bad) to assign class
  - Add to Accepts and re-model
- Simple
- Arbitrary cutoff to assign goods and bads
- Good/Bad model needs to be very good
- No adjustment for p(approve).

Augmentation 2

- Augmentation 2 *(Coffman, Chandler 1977)*
  - Build accept/reject model, obtain p(accept)
  - Build good/bad model
  - Adjust case weights of good/bad model to reflect probability of acceptance
- Recognizes need to adjust for p(approve).

Parceling

- Parceling (also called re-weighting)
  - score rejects with G/B model
  - split (randomly) rejects into proportional G and B groups.

Score	# Bad	# Good	% Bad	% Good	Reject	Rej - Bad	Rej - Good
0-99	24	10	70.3%	29.7%	342	240	102
100-199	54	196	21.6%	78.4%	654	141	513
200-299	43	331	11.5%	88.5%	345	40	305
300-399	32	510	5.9%	94.1%	471	28	443
400+	29	1,232	2.3%	97.7%	778	18	760

continued...

Parceling

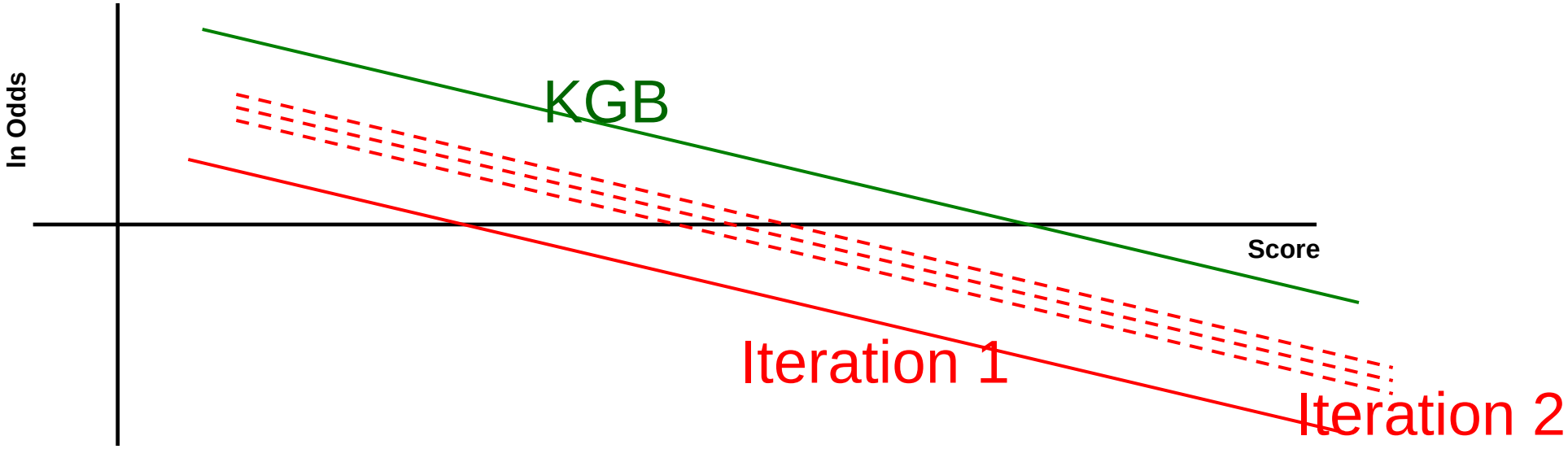
- But ..
  - Reject bad proportion cannot be the same as approved?
  - Allocate higher proportion of bads from reject
  - Rule of thumb: bad rate for rejects should be 2–4 times that of approves.
- Quick and simple
- Good/Bad model better be good
- May understate rejected bad rate.

Iterative Reclassification

- Iterative Reclassification *(McLachlan, 1975)*
  - Build good/bad model using accepts
  - Score rejects and assign class based on chosen p(bad) cutoff
  - Rebuild model with combined dataset
  - Score rejects and re-assign class
  - Repeat until parameter estimates (and p(bad)) converge.
- Can be modified for p(good) and p(bad) target assignment.

Iterative Reclassification

- can be done as a plot of ln (odds) versus score.



Fuzzy Augmentation

- Step 1: Classification
  - Build good/bad model
  - Score rejects with G/B model
  - Do not assign a reject to a class
  - Create 2 weighted cases for each reject, using p(good) and p(bad).

Fuzzy Augmentation

- Step 2: Augmentation
  - Combine rejects with accepts, adjusting for approval rate
  - For this, weigh rejects again: weight determines how much more frequent an actual case is compared to an inferred case in the augmented dataset
  - Freq of a 'Good' from rejects = p(good) x weight
- Step 3: Remodel.

EM users: This is in the EM RI node.

Freq= p(good) x (reject rate/approval rate) x (#accepts/#rejects)

 # rejects/accepts are proportional to actual population I.e. weighted, not raw counts

Fuzzy Augmentation

- No need for arbitrary classification cut-off
- Augmentation step: better approach for choosing the importance of rejects.

Nearest Neighbor (Clustering)

- Clustering
  - Create 2 sets of clusters: goods and bads
  - Run rejects through both clusters
  - Compare Euclidean distances to assign most likely performance
  - Combine accepts and rejects and re-model
- Measures are relative
- Adjustment for p(approve) can be added at augmentation step.
- Can also use Memory-based Reasoning.

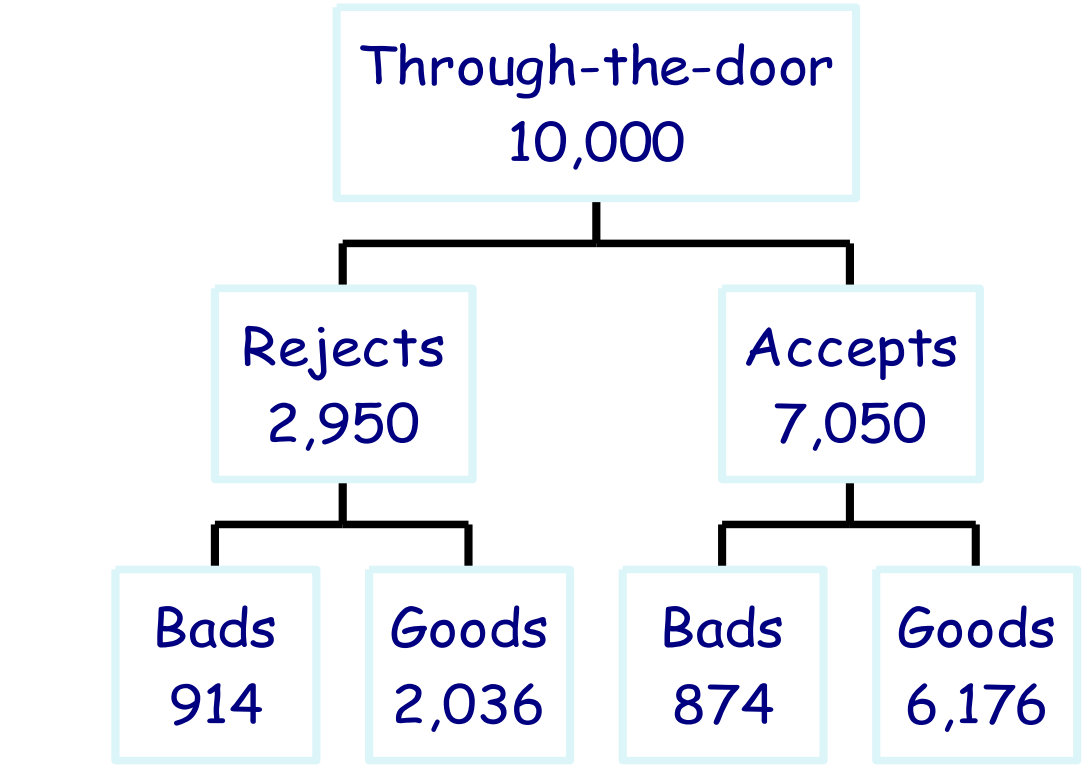
Other Techniques

- Heckman's Correction
  - http://ewe3.sas.com/techsup/download/stat/heckman.html
  - Heckman, James. "Sample Selection Bias as a Specification Error", Econometrica, Vol 47, No 1., January 1979, pp. 153-161.
  - Greene, William. "Sample Selection Bias as a Specification Error: Comment", {\sl Econometrica}, Vol. 49, No. 3, May 1981, pp. 795-798.
- Mixture Decomposition
  - B.S Everitt and D.J. Hand, Finite Mixture Distributions (London:Chapman & Hall, 1981)

Verification

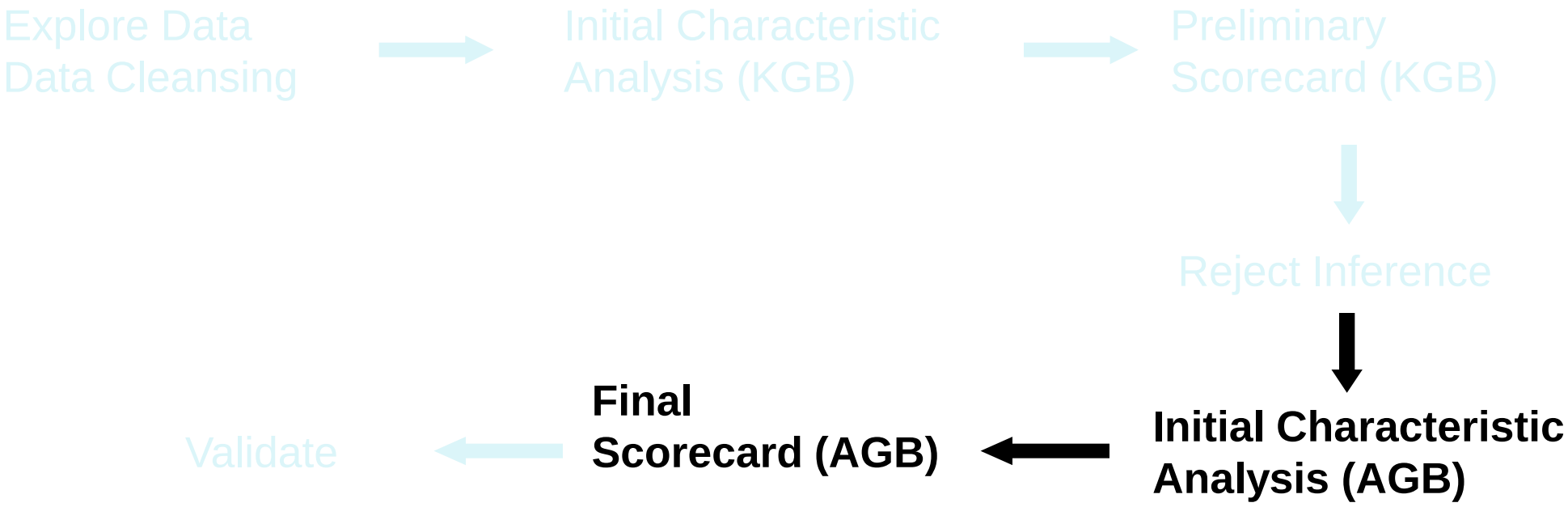
- Compare bad rates/odds for known versus inferred, and use rule of thumb.
- Review bad rates/weight of evidence of pre- versus post inference groupings.
- Create “fake rejects” and test.
  - assign some accepted accounts as rejects with an artificial cutoff and test methods.

Factoring – Post Inference



- After rejects have been inferred, we build the post-inference data sets for the final scorecard production.
- So the sample bias is solved and you can apply the scorecard on the entire population.

Process Flow



Final Scorecard

- Repeat Exploration, Initial Characteristics Analysis and Regression for “All Good Bad” data set
  - Scaling
  - Assessment
    - Misclassification
    - Strength.

Scorecard Scaling (conversion into points)

- Why scale?
  - Implementation software – batch versus on-line
    - Marketing uses (off line selection, build retention model, score and isolate account numbers) vs. online decision support and app processing software
  - Ease of understanding and interpretation
    - End user can deal with points easier than weights
  - Continuity
    - previous scorecards were grouped/scaled . .and you want to have the same format and scaling.
  - Legal requirements
    - legal requirements to identify characteristics and reasons for decline
  - Components
    - Odds at a score
    - Points to double the odds
    - Example: Odds of 20:1 at 200, and odds double every 20 points.

Scorecard Scaling

- This is the transformation from parameter estimates to scores.
- Result: get a score card with discrete points, related to each other and the final score related to odds.
- odds doubling every 20 points

Age	
18-24	10
25-29	15
30-37	25
38-45	28
46+	35
Time at Res	
0-6	12
7-18	25
19-36	28
37+	40
Region	
Major Urban	20
Minor Urban	25
Rural	15
Inq 6 mth	
0	40
1-3	30
4-5	15
6+	10

Score	Odds
200	20
201	23
202	25
203	26
.	
.	
220	40
.	
240	80

Scorecard Scaling

In general:

$$\text{Score} = A + B \log(\text{odds})$$
$$\text{Score} + \text{PDO} = A + B \log(2 * \text{odds})$$

- Offset A and Factor B are to be calculated
  - Odds = odds at which to fix a score
  - Score = score at point x
  - PDO = points to double the odds

Scorecard Scaling

- Solving for PDO:

PDO = B Log (2), therefore

B = PDO/log(2) ; A = Score – {B log (Odds)}

Example, Odds of 50:1 at 600 and 20 pdo

B = 20/log(2) = 28.8539

A = 600 – {28.8539 log (50)} = 487.123

Score = 487.123 + 28.8539 log (odds)

Or log (odds) = (-16.88239) + 0.03465*Score

Scorecard Scaling

- The points for each attribute are calculated by multiplying the Weight of Evidence of the attribute with the regression coefficient of the characteristic, then adding a fraction of the regression intercept, then multiplying this by -1 and by the factor and finally adding a fraction of the offset.

$$-(woe_i * \beta_i + \frac{a}{n}) * \underbrace{factor}_{B} + \underbrace{\frac{offset}{n}}_{A}$$

- The negative sign is there because we switch from bad/good in modeling (regression) to good/bad in scaling (high scores being better than low scores).

Scorecard Scaling

$$score = \log(odds) * factor + offset =$$

$$-(\sum_{i=1}^{n}(woe_i * \beta_i) + a) * factor + offset =$$

$$-(\sum_{i=1}^{n}(woe_i * \beta_i + \frac{a}{n})) * factor + offset =$$

$$\sum_{i=1}^{n}(-(woe_i * \beta_i + \frac{a}{n}) * factor + \frac{offset}{n})$$

- **β = is the regression coefficient**
- **WOE = weight of evidence for the attribute**
- **n = number of characteristics**
- **a = intercept**

Check Points Allocation

Age	Weight	Scorecard 1	Scorecard 2
Missing	-55.50	16	16
18-22	-108.41	12	12
23-26	-72.04	18	18
27-29	-3.95	26	14
30-35	70.77	35	38
35-44	122.04	43	44
44 +	165.51	51	52

- Scorecard 1 looks OK - logical distribution - as age increases, points increase according to weight
- But Scorecard 2 doesn't.
- Why?
  - Correlation? Quirk in the data? Grouping? Maybe weights were too close together and not enough differentiation - repeat grouping with more distinct groups, and repeat regression.

FICO score

FICO is a unified score, which you can get from your score by linear transformation. The aim is to compute these transformation coefficients for every scorecard, because then you can compare quality of portfolios. If your development data are old enough, so that you can observe 90DPD @12MOB, take random sample (30 000 observations) from them, if not, take older data and score them by your new scorecard. Make a table according to example below, compute FICO score for each category as linear transformation ln(G/B)->Fico, defined FICO=(x+7.58)/0.0157. Apply linear regression on median score and FICO.

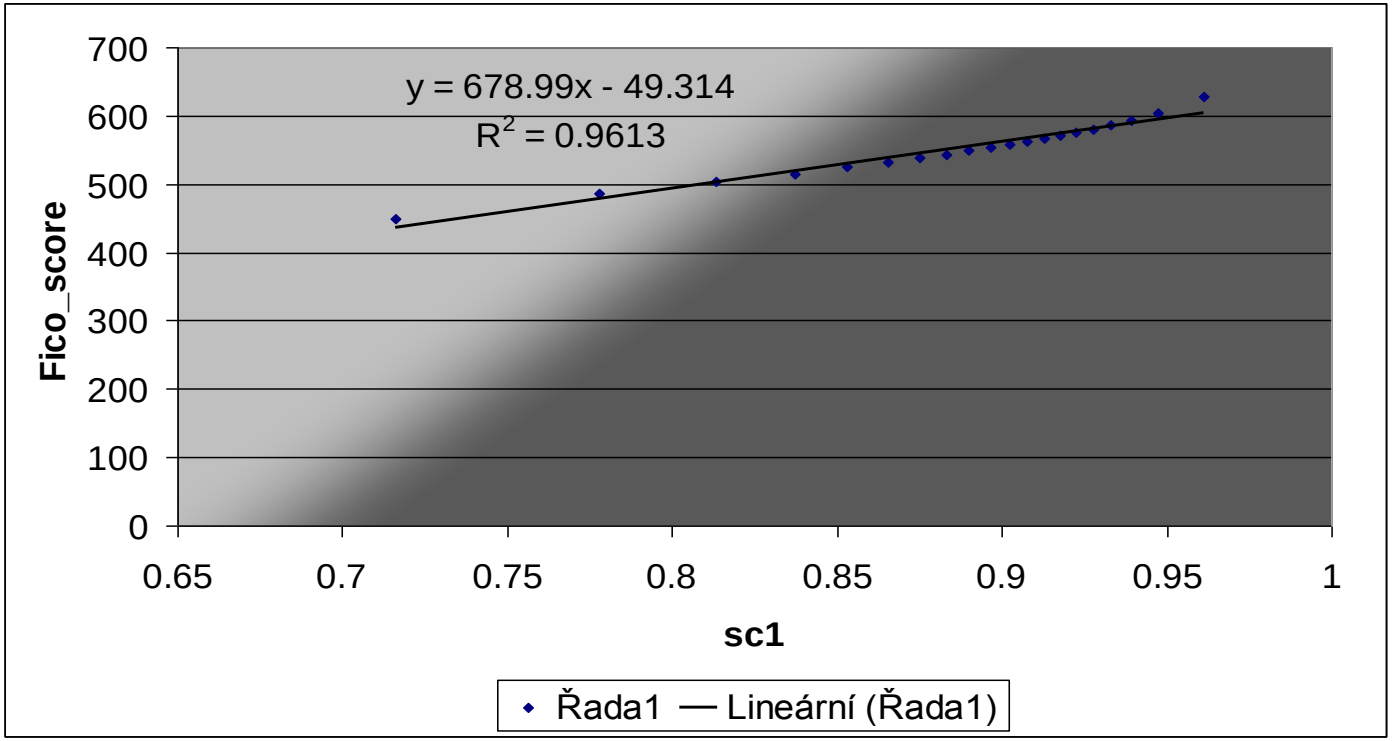
(-0.5319 + 7.58)/0.0157 = 449

(1497-943)/943 = 0.5874

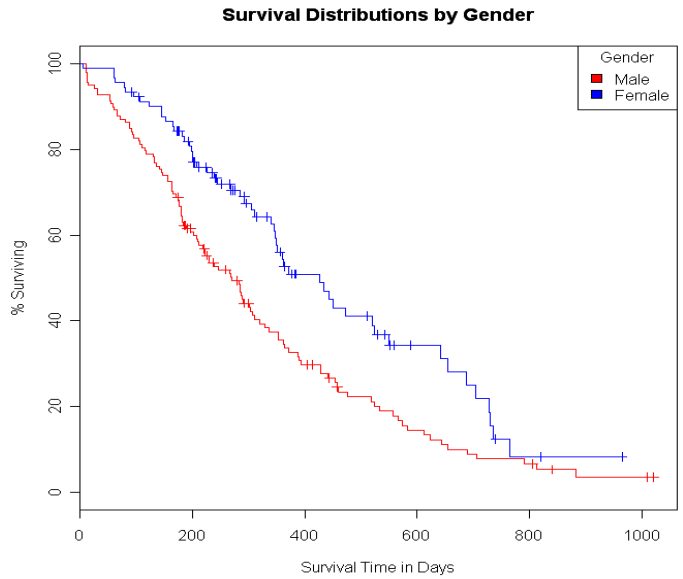
Median of the category	Lower bound of the score	Upper bound of the score	Numb @12 Mob	Ever 90@12 MOB	Good/Bad	ln(G/B)	Fico
0.716	0	0.752399981	1497	943	0.5874867	-0.5319	449
0.778	0.7524	0.7968	1504	733	1.0518418	0.050543	486
0.8132	0.7968	0.8268	1496	630	1.3746032	0.318165	503
0.8371	0.8268	0.8457	1508	564	1.6737589	0.515072	516
0.8532	0.8457	0.8596	1495	510	1.9313726	0.658231	525
0.8654	0.8596	0.8703	1500	474	2.164557	0.772216	532
0.875	0.8703	0.8792	1513	447	2.3847875	0.86911	538
0.8833	0.8792	0.8869	1489	414	2.5966184	0.95421	544
0.8901	0.8869	0.8934	1496	393	2.8066158	1.031979	549
0.8968	0.8934	0.8996	1521	378	3.0238095	1.106517	553
0.9024	0.8996	0.9051	1491	351	3.2478633	1.177997	558

FICO score

FICO transformation graph



9. Introduction to Survival Analysis



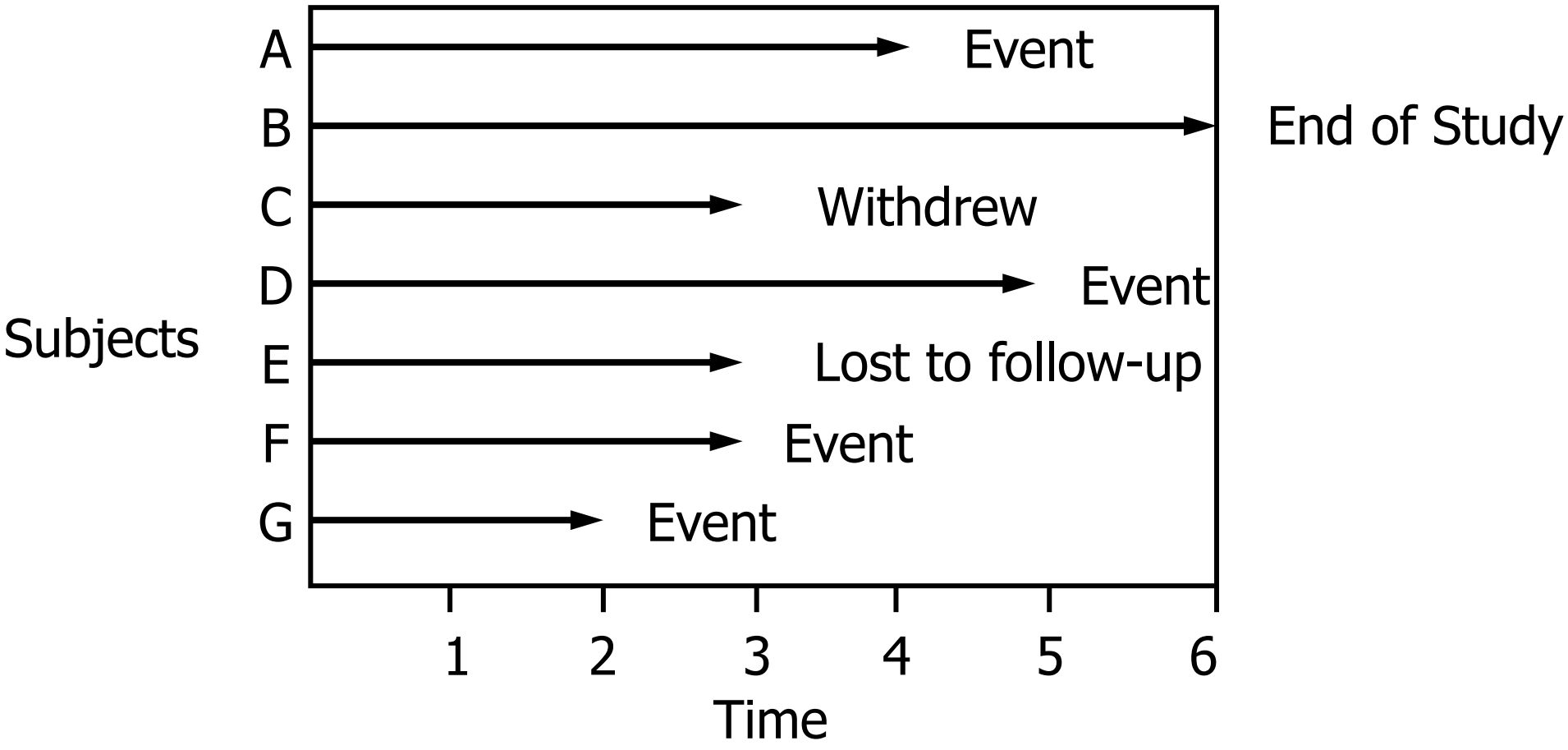
What Is Survival Analysis?

- *Survival analysis* is a class of statistical methods for which the outcome variable of interest is time until an event occurs.
- Time is measured from the beginning of follow-up until the event occurs or a reason occurs for the observation of time to end.

Examples of Survival Analysis

- Follow-up of patients undergoing surgery to measure how long they survived after the surgery
- Follow-up of leukemia patients in remission to measure how long they remain in remission
- Follow-up of clients to measure how long they stay non-defaulted

What Is Survival Analysis?



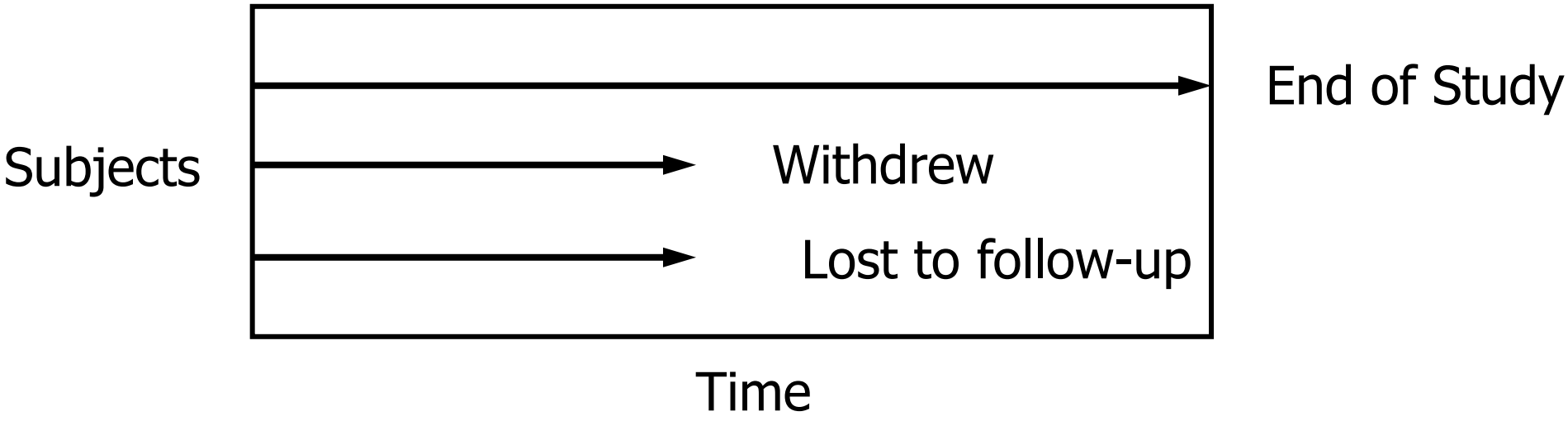
Data Structure

Subject	Survival Time	Status
A	4.0	1 (event)
B	6.0	0 (censored)
C	3.0	0
D	5.0	1
E	3.0	0
F	3.0	1
G	2.0	1

Problems with Conventional Methods

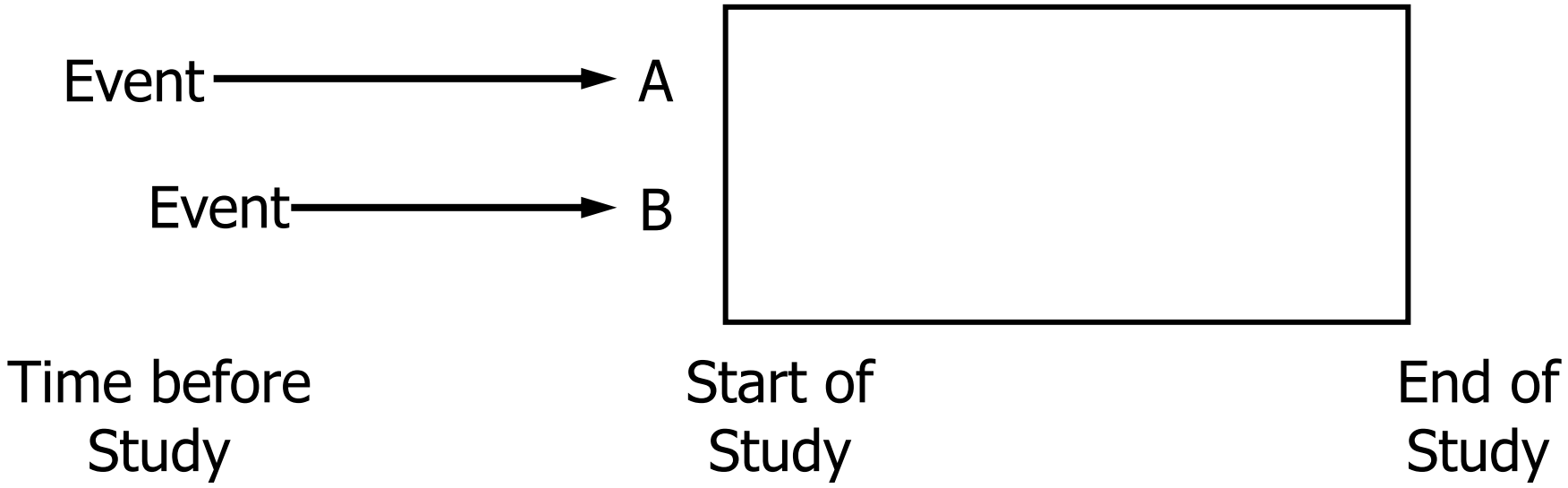
- Logistic regression
  - ignores information on the timing of events
  - cannot handle time-dependent covariates.
- Linear regression
  - cannot handle censored observations
  - cannot handle time-dependent covariates
  - is not appropriate because time to event can have unusual distribution.

Right-Censoring



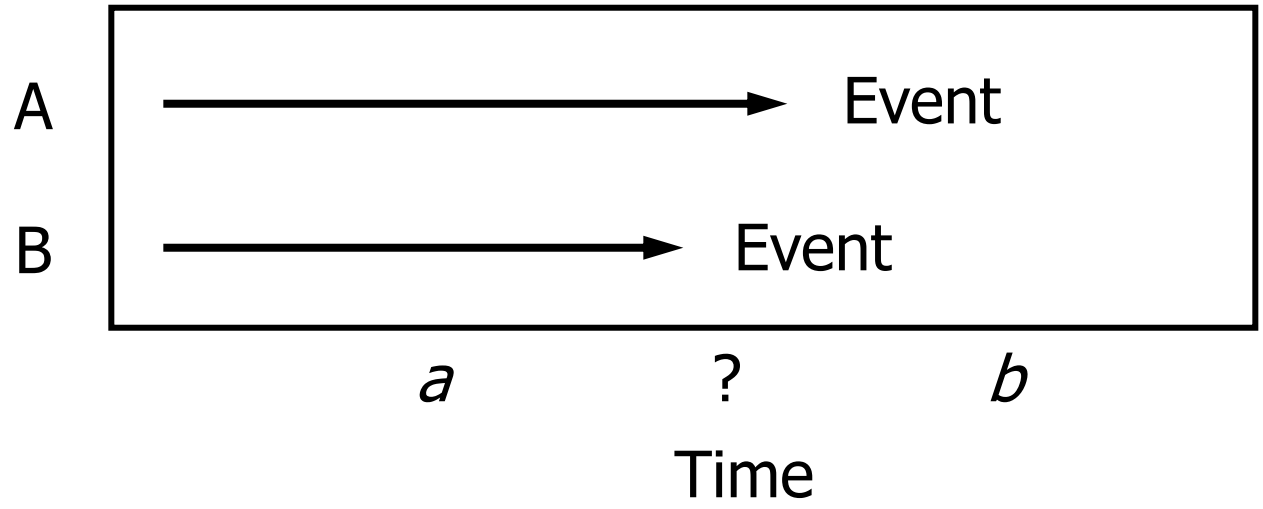
An observation is right-censored if the observation is terminated before the event occurs.

Left-Censoring



An observation is left-censored when the observation experiences the event before the start of the follow-up period.

Interval-Censoring



An observation is interval-censored if the only information you know about the survival time is that it is between the values *a* and *b*.

Types of Right-Censoring

- Type I subjects survived until end of the study. Censoring time is fixed.
- Type II subjects survived until end of the study. Censoring time occurs when a pre-specified number of events have occurred.
- Random observations are terminated for reasons that are not under the control of the investigator.

Uninformative Censoring

- Censoring is uninformative if it
  - occurs when the reasons for termination are unrelated to the risk of the event
  - assumes that subjects who are censored at time X should be representative of all those subjects with the same values of the predictor variables who survive to time X
  - does not bias the parameter estimates and statistical inference.

Informative Censoring

- Censoring is informative if it
  - occurs when the reasons for termination of the observation are related to the risk of the event
  - results in biased parameter estimates and inaccurate statistical inference about the survival experience.

Recommendations Regarding Informative Censoring

- When designing and conducting studies, reduce the amount of random censoring.
- Always analyze the pattern of censoring to see whether it is related to a subset of subjects.
- Include in your study any explanatory variables that may affect the rate of censoring.

Time Origin Recommendations

- Choose a time origin that marks the onset of continuous exposure to the risk of the event.
- Choose the time of randomization to treatment as the time origin in experimental studies.
- If there are several time origins available, consider controlling for the other time origins by including them as covariates.

Survival Analysis

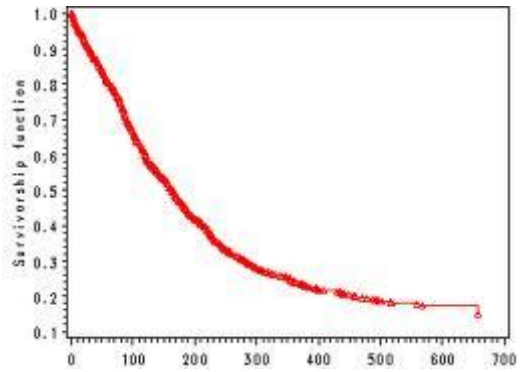
- The goals of survival analysis might be to
  - estimate and interpret survival and hazard functions from survival data
  - compare survival and hazard functions among different groups
  - assess the relationship of time-independent and time-dependent explanatory variables to survival time
  - predict the remaining time until the event.

Survival Function (funkce přežití)

- T … náhodná veličina označující délku přežití (čas do sledované události nebo cenzorování).
- δ… indikátor události (δ=1 pokud událost nastala, δ=0 pokud je pozorování cenzorované)
- S(t)… funkce přežití (survival function) $S(t) = P(T > t)$ vyjadřuje pravděpodobnost, že jedinec v čase t ještě žije.
- Platí:

$$S(0) = 1$$

$$\lim_{t \to \infty} S(t) = 0$$



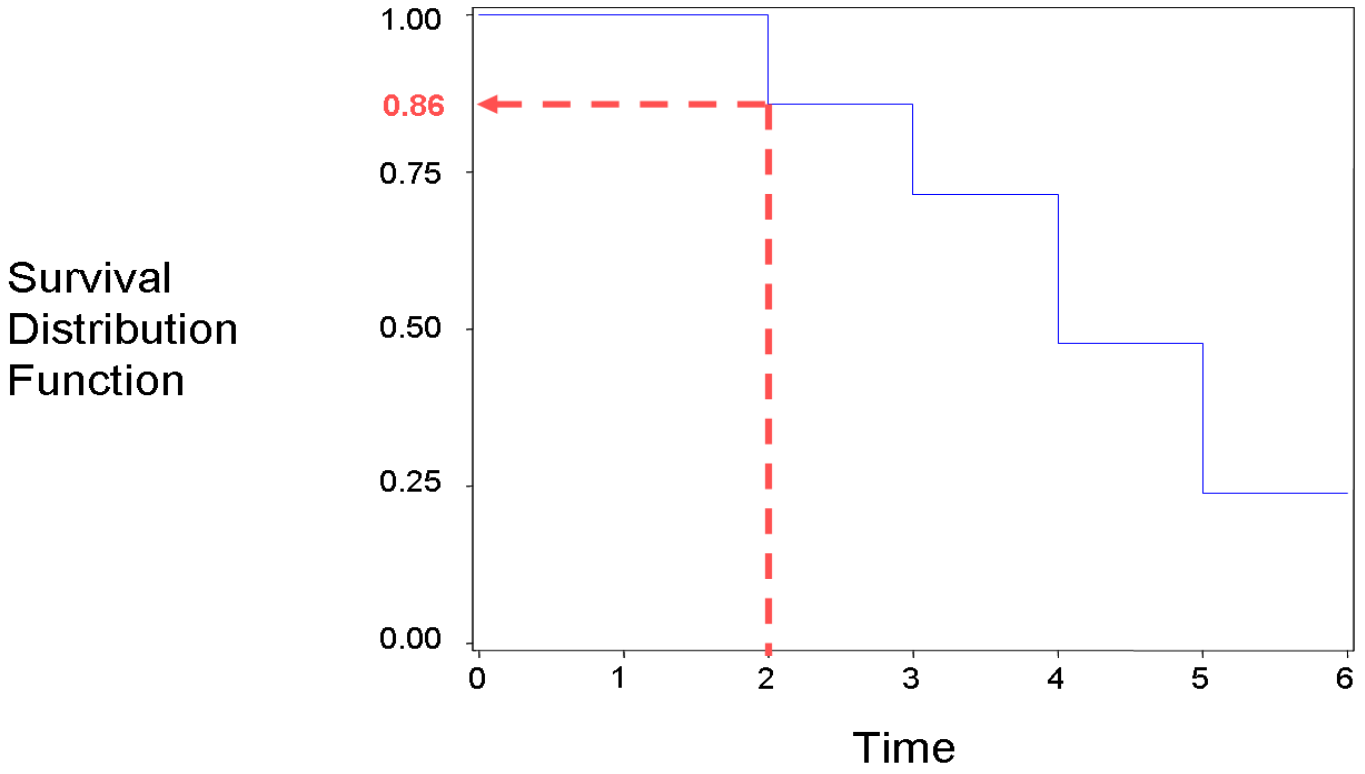
Kaplan-Meier estimation

$$\hat{S}(t_j) = \hat{S}(t_{j-1})(1 - d_j / n_j) = \prod_{i=1}^{j} \frac{n_i - d_i}{n_i}$$

n_j... počet objektů pozorovaných do doby t_j
d_j...počet objektů s událostí v době t_j

Time	Number Events	Number Censored	Number At Risk	Cumulative Survival
0	0	0	7	1.00
1	0	0	7	1.00
2	1	0	7	(7-1)/7=.86
3	1	+ 2	6 - 3	.86*5/6=.71
4	1	0	3	.71*2/3=.48
5	1	1	2	.48*1/2=.24
6	0	0	0	-------

Kaplan-Meier Curve



Další metody odhadu

The Breslow estimate of the survivor function is

$$\hat{S}(t_i) = \exp\left(-\sum_{j=1}^{i} \frac{d_j}{n_j}\right)$$

Note that the Breslow estimate is the exponentiation of the negative Nelson-Aalen estimate of the cumulative hazard function.

The Fleming-Harrington estimate (Fleming and Harrington; 1984) of the survivor function is

$$\hat{S}(t_i) = \exp\left(-\sum_{k=1}^{i} \sum_{j=0}^{d_k-1} \frac{1}{n_k - j}\right)$$

If the frequency values are not integers, the Fleming-Harrington estimate cannot be computed.

Life Table Method

- The *life table* method
  - is useful when there are a large number of observations
  - groups the event times into intervals
  - can produce estimates and plots of the hazard function.

Life Table Method

The life-table estimates are computed by counting the numbers of censored and uncensored observations that fall into each of the time intervals $[t_{i-1}, t_i)$, $i = 1, 2, \ldots, k+1$, where $t_0 = 0$ and $t_{k+1} = \infty$. Let n_i be the number of units entering the interval $[t_{i-1}, t_i)$, and let d_i be the number of events occurring in the interval. Let $b_i = t_i - t_{i-1}$, and let $n_i' = n_i - w_i/2$, where w_i is the number of units censored in the interval. The ***effective sample size*** of the interval $[t_{i-1}, t_i)$ is denoted by n_i'. Let t_{mi} denote the midpoint of $[t_{i-1}, t_i)$.

The conditional probability of an event in $[t_{i-1}, t_i)$ is estimated by

$$\hat{q}_i = \frac{d_i}{n_i'}$$

and its estimated standard error is

$$\hat{\sigma}(\hat{q}_i) = \sqrt{\frac{\hat{q}_i \hat{p}_i}{n_i'}}$$

where $\hat{p}_i = 1 - \hat{q}_i$.

The estimate of the survival function at t_i is

$$\hat{S}(t_i) = \begin{cases} 1 & i = 0 \\ \hat{S}(t_{i-1}) p_{i-1} & i > 0 \end{cases}$$

and its estimated standard error is

$$\hat{\sigma}\left(\hat{S}(t_i)\right) = \hat{S}(t_i) \sqrt{\sum_{j=1}^{i-1} \frac{\hat{q}_j}{n_j' \hat{p}_j}}$$

Differences between KM and Life Table Methods

- In the Kaplan-Meier method,
  - time interval boundaries are determined by the event times themselves
  - censored observations are assumed to be at risk for the whole event time period.
- In the life table method,
  - time interval boundaries are determined by the user
  - censored observations are censored at the midpoint of the time interval.

Standard error of KM estimate

- The corresponding estimate of the standard error is computed using Greenwood's formula (Kalbfleisch and Prentice; 1980) as

$$\hat{\sigma}(\hat{S}(t_j)) = \hat{S}(t_j)\sqrt{\sum_{i=1}^{j} \frac{d_i}{n_i(n_i - d_i)}}$$

Pointwise Confidence Limits

Pointwise confidence limits are computed for the survivor function, and for the density function and hazard function when the life-table method is used. Let α be specified by the ALPHA= option. Let $z_{\alpha/2}$ be the critical value for the standard normal distribution. That is, $\Phi(-z_{\alpha/2}) = \alpha/2$, where Φ is the cumulative distribution function of the standard normal random variable.

Survivor Function

When the computation of confidence limits for the survivor function $S(t)$ is based on the asymptotic normality of the survival estimator $\hat{S}(t)$, the approximate confidence interval might include impossible values outside the range [0,1] at extreme values of t. This problem can be avoided by applying the asymptotic normality to a transformation of $S(t)$ for which the range is unrestricted. In addition, certain transformed confidence intervals for $S(t)$ perform better than the usual linear confidence intervals (Borgan and Liestøl; 1990). The CONFTYPE= option enables you to pick one of the following transformations: the log-log function (Kalbfleisch and Prentice; 1980), the arcsine-square root function (Nair; 1984), the logit function (Meeker and Escobar; 1998), the log function, and the linear function.

Let g be the transformation that is being applied to the survivor function $S(t)$. By the delta method, the standard error of $g(\hat{S}(t))$ is estimated by

$$\tau(t) = \hat{\sigma}\left[g(\hat{S}(t))\right] = g'\left(\hat{S}(t)\right)\hat{\sigma}[\hat{S}(t)]$$

where g' is the first derivative of the function g. The 100(1–α)% confidence interval for $S(t)$ is given by

$$g^{-1}\left\{g[\hat{S}(t)] \pm z_{\frac{\alpha}{2}} g'[\hat{S}(t)]\hat{\sigma}[\hat{S}(t)]\right\}$$

where g^{-1} is the inverse function of g.

Pointwise Confidence Limits

Arcsine-Square Root Transformation

The estimated variance of $\sin^{-1}\left(\sqrt{\hat{S}(t)}\right)$ is $\hat{\tau}^2(t) = \frac{\sigma^2[\hat{S}(t)]}{4\hat{S}(t)[1-\hat{S}(t)]}$. The 100(1–$\alpha$)% confidence interval for $S(t)$ is given by

$$\sin^2\left\{\max\left[0, \sin^{-1}\left(\sqrt{\hat{S}(t)}\right) - z_{\frac{\alpha}{2}}\hat{\tau}(t)\right]\right\} \leq S(t) \leq \sin^2\left\{\min\left[\frac{\pi}{2}, \sin^{-1}\left(\sqrt{\hat{S}(t)}\right) + z_{\frac{\alpha}{2}}\hat{\tau}(t)\right]\right\}$$

Linear Transformation

This is the same as having no transformation in which g is the identity. The 100(1–α)% confidence interval for $S(t)$ is given by

$$\hat{S}(t) - z_{\frac{\alpha}{2}}\hat{\sigma}\left[\hat{S}(t)\right] \leq S(t) \leq \hat{S}(t) + z_{\frac{\alpha}{2}}\hat{\sigma}\left[\hat{S}(t)\right]$$

Log Transformation

The estimated variance of $\log(\hat{S}(t))$ is $\hat{\tau}^2(t) = \frac{\sigma^2(\hat{S}(t))}{\hat{S}^2(t)}$. The 100(1–$\alpha$)% confidence interval for $S(t)$ is given by

$$\hat{S}(t)\exp\left(-z_{\frac{\alpha}{2}}\hat{\tau}(t)\right) \leq S(t) \leq \hat{S}(t)\exp\left(z_{\frac{\alpha}{2}}\hat{\tau}(t)\right)$$

Pointwise Confidence Limits

Log-Log Transformation

The estimated variance of $\log(-\log(\hat{S}(t))$ is $\hat{\tau}^2(t) = \frac{\sigma^2[\hat{S}(t)]}{[\hat{S}(t)\log(\hat{S}(t))]^2}$. The 100(1–$\alpha$)% confidence interval for $S(t)$ is given by

$$[\hat{S}(t)]^{\exp\left(z_{\frac{\alpha}{2}}\hat{\tau}(t)\right)} \leq S(t) \leq [\hat{S}(t)]^{\exp\left(-z_{\frac{\alpha}{2}}\hat{\tau}(t)\right)}$$

Logit Transformation

The estimated variance of $\log\left(\frac{\hat{S}(t)}{1-\hat{S}(t)}\right)$ is $\hat{\tau}^2(t) = \frac{\sigma^2(\hat{S}(t))}{\hat{S}^2(t)[1-\hat{S}(t)]^2}$. The 100(1–$\alpha$)% confidence limits for $S(t)$ are given by

$$\frac{\hat{S}(t)}{\hat{S}(t)+\left[1-\hat{S}(t)\right]\exp\left(z_{\frac{\alpha}{2}}\hat{\tau}(t)\right)} \leq S(t) \leq \frac{\hat{S}(t)}{\hat{S}(t)+\left[1-\hat{S}(t)\right]\exp\left(-z_{\frac{\alpha}{2}}\hat{\tau}(t)\right)}$$

Simultaneous Confidence Intervals

- Confidence bands show with a given confidence level that the survival function falls within the interval for all time points.
- There are two approaches in SAS for constructing simultaneous confidence intervals.
- Equal precision (CONFBAND=EP) confidence intervals are proportional to the pointwise confidence intervals.
- Hall-Wellner (CONFBAND=HW) confidence intervals are not proportional to the pointwise confidence intervals.
- Transformations that are used to improve the pointwise confidence bands can be used to improve the simultaneous confidence bands.

Simultaneous Confidence Intervals

Let $a_L = \dfrac{n\sigma_S^2(t_L)}{1+n\sigma_S^2(t_L)}$ and $a_U = \dfrac{n\sigma_S^2(t_U)}{1+n\sigma_S^2(t_U)}$

Let $\{W^0(u), 0 \le u \le 1\}$ be a Brownian bridge.

Hall-Wellner Band

The 100(1–α)% HW band of Hall and Wellner (1980) is

$$\hat{S}(t) - h_\alpha(a_L, a_U) n^{-\frac{1}{2}} [1+n\sigma_S^2(t)]\hat{S}(t) \le S(t) \le \hat{S}(t) + h_\alpha(a_L, a_U) n^{-\frac{1}{2}} [1+n\sigma_S^2(t)]\hat{S}(t)$$

for all $t_L \le t \le t_U$, where $h_\alpha(a_L, a_U)$ is given by

$$\alpha = \Pr\{\sup_{a_L \le u \le a_U} |W^0(u)| > h_\alpha(a_L, a_U)\}$$

The critical values are computed from the results in Chung (1986).

Simultaneous Confidence Intervals

Note that the given confidence band has a formula similar to that of the (linear) pointwise confidence interval, where $h_\alpha(a_L, a_U)$ and $n^{-\frac{1}{2}}[1+n\sigma_S^2(t)]\hat{S}(t)$ in the former correspond to $z_{\frac{\alpha}{2}}$ and $\hat{\sigma}(\hat{S}(t))$ in the latter, respectively. You can obtain the other transformations (arcsine-square root, log-log, log, and logit) for the confidence bands by replacing $z_{\frac{\alpha}{2}}$ and $\hat{\tau}(t)$ in the corresponding pointwise confidence interval formula by $h_\alpha(a_L, a_U)$ and the following $\hat{\tau}(t)$, respectively.

- Arcsine-Square Root Transformation

$$\hat{\tau}(t) = \frac{1+n\sigma_S^2(t)}{2}\sqrt{\frac{S(t)}{n[1-S(t)]}}$$

- Log Transformation

$$\hat{\tau}(t) = \frac{1+n\sigma_S^2(t)}{\sqrt{n}}$$

- Log-Log Transformation

$$\hat{\tau}(t) = \frac{1+n\sigma_S^2(t)}{\sqrt{n}|\log[\hat{S}(t)]|}$$

- Logit Transformation

$$\hat{\tau}(t) = \frac{1+n\sigma_S^2(t)}{\sqrt{n}[1-\hat{S}(t)]}$$

Simultaneous Confidence Intervals

Equal-Precision Band

The 100(1–α)% EP band of Nair (1984) is

$$\hat{S}(t) - e_\alpha(a_L, a_U)\hat{S}(t)\sigma_S(t) \le S(t) \le \hat{S}(t) + e_\alpha(a_L, a_U)\hat{S}(t)\sigma_S(t)$$

for all $t_L \le t \le t_U$, where $e_\alpha(a_L, a_U)$ is given by

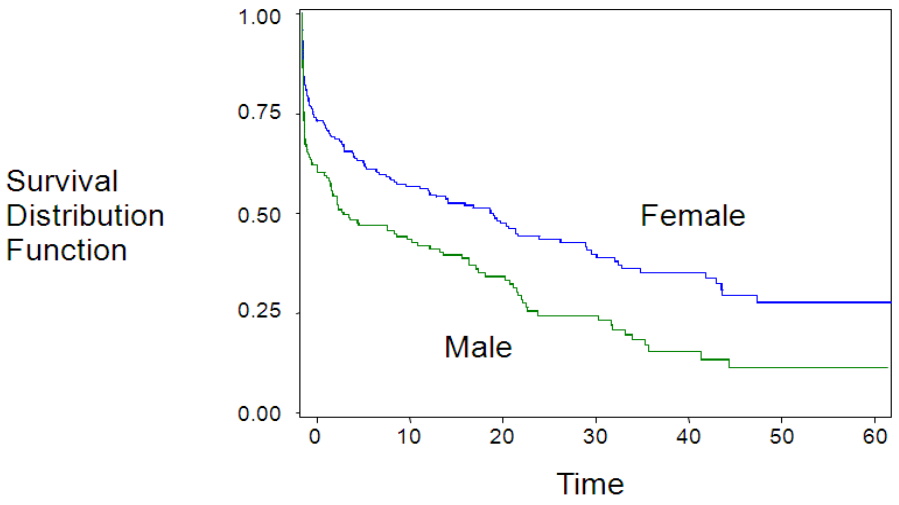
$$\alpha = \Pr\{\sup_{a_L \le u \le a_U} \frac{|W^0(u)|}{[u(1-u)]^{\frac{1}{2}}} > e_\alpha(a_L, a_U)\}$$

PROC LIFETEST uses the approximation of Miller and Siegmund (1982, Equation 8) to approximate the tail probability in which $e_\alpha(a_L, a_U)$ is obtained by solving x in

$$\frac{4x\phi(x)}{x} + \phi(x)\left(x - \frac{1}{x}\right)\log\left[\frac{a_U(1-a_L)}{a_L(1-a_U)}\right] = \alpha$$

where $\phi(x)$ is the standard normal density function evaluated at x. Note that the confidence bounds given are proportional to the pointwise confidence intervals. As a matter of fact, this confidence band and the (linear) pointwise confidence interval have the same formula except for the critical values ($z_{\frac{\alpha}{2}}$ for the pointwise confidence interval and $e_\alpha(a_L, a_U)$ for the band). You can obtain the other transformations (arcsine-square root, log-log, log, and logit) for the confidence bands by replacing $z_{\frac{\alpha}{2}}$ by $e_\alpha(a_L, a_U)$ in the formula of the pointwise confidence intervals.

Comparing Survival Functions



Let k be the number of groups. Let $S_i(t)$ be the underlying survivor function ith group, $i = 1, \ldots, k$. The null and alternative hypotheses to be tested are

$H_0 : S_1(t) = S_2(t) = \ldots = S_k(t)$ for all $t \leq \tau$

versus

H_1 : at least one of the $S_i(t)$'s is different for some $t \leq \tau$

respectively, where τ is the largest observed time.

Likelihood-Ratio Test

- The *likelihood-ratio test*
  - is a parametric test that assumes that the distribution of event times follows an exponential distribution
  - can be verified if the plot of the negative log of the survival function by time follows a linear trend with an origin of 0.

The likelihood ratio test statistic (Lawless; 1982) for test H_0 versus H_1 assumes that the data in the various samples are exponentially distributed and tests that the scale parameters are equal. The test statistic is computed as

$$\chi^2 = 2N \log\left(\frac{T}{N}\right) - 2\sum_{j=1}^{k} N_j \log\left(\frac{T_j}{N_j}\right)$$

where N_j is the total number of events in the jth stratum, $N = \sum_{j=1}^{k} N_j$, T_j is the total time on test in the jth stratum, and $T = \sum_{j=1}^{k} T_j$. The approximate probability value is computed by treating χ^2 as having a chi-square distribution with k–1 degrees of freedom.

Nonparametric Tests

Let $t_1 < t_2 < \ldots < t_D$ be the distinct event times in the pooled sample. At time t_i, let $W(t_i)$ be a positive weight function, and let n_{ij} and d_{ij} be the size of the risk set and the number of events in the jth sample, respectively. Let $n_i = \sum_{j=1}^{k} n_{ij}$, $d_i = \sum_{j=1}^{k} d_{ij}$, and $s_i = n_i - d_i$.

The choices of the weight function $W(t_i)$ are given in Table 49.3.

Table 49.3 Weight Functions for Various Tests

Test	$W(t_i)$
Log-rank	1.0
Wilcoxon	n_i
Tarone-Ware	$\sqrt{n_i}$
Peto-Peto	$\tilde{S}(t_i)$
Modified Peto-Peto	$\tilde{S}(t_i)\frac{n_i}{n_i+1}$
Harrington-Fleming (p,q)	$[\hat{S}(t_i)]^p[1-\hat{S}(t_i)]^q, p \geq 0, q \geq 0$

where $\hat{S}(t)$ is the product-limit estimate at t for the pooled sample, and $\tilde{S}(t)$ is a survivor function estimate close to $\hat{S}(t)$ given by

$$\tilde{S}(t) = \prod_{t_i \leq t} \left(1 - \frac{d_i}{n_i + 1}\right)$$

Log-Rank Test

- The *log-rank test*
  - tests whether the survival functions are statistically equivalent
  - is a large-sample chi-square test that uses the observed and expected cell counts across the event times
  - has maximum power when the ratio of hazards is constant over time
  - loses power in the presence of interactions.

Log-Rank Test for Two Groups

$$\frac{\left(\sum_{j=1}^{r}(d_{1j}-e_{1j})\right)^2}{\text{var}\left(\sum_{j=1}^{r}(d_{1j}-e_{1j})\right)}$$

where d_{1j} is the number of events that occur in group 1 at time j, and e_{1j} is the expected number of events in group 1 at time j.

Wilcoxon Test

- The *Wilcoxon test*
  - is also known as the Gehan test or the Breslow test
  - can be biased if the pattern of censoring is different between the groups
  - loses power in the presence of interactions.

Wilcoxon Test for Two Groups

$$\frac{\left(\sum_{j=1}^{r} n_j (d_{1j} - e_{1j})\right)^2}{\text{var}\left(\sum_{j=1}^{r} n_j (d_{1j} - e_{1j})\right)}$$

where n_j is the total number at risk at each time point.

Log-Rank versus Wilcoxon Test

- Log-rank test
  - is more sensitive than the Wilcoxon test to differences between groups in later points in time.
- Wilcoxon test
  - is more sensitive than the log-rank test to differences between groups that occur in early points in time.

New Tests in SAS®9

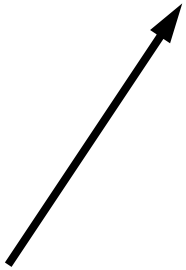
- Tarone-Ware test uses a weight equal to the square root of the number at risk. This gives more weight to differences between the observed and expected number of events at time points where there is the most data.
- Peto-Peto and Modified Peto-Peto tests use weights that depend of the observed survival experience of the combined sample. The principle advantage of these tests is that they do not depend on the censoring experience of the groups.
- Harrington-Fleming test incorporates features of both the log-rank and Peto-Peto tests.

Stratified Tests

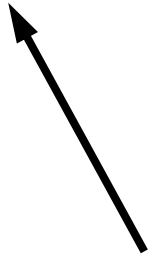
- Stratified tests are used when you want to compare survival functions across *k* populations while controlling for other covariates.
- They are different than the *k*-sample tests which only compare survival functions across *k* populations.
- Stratified tests are available in SAS® 9 with the use of the GROUP= option in the STRATA statement.

Syntax for Stratified Tests

STRATA variable1 / GROUP variable2 TEST=(list);



Distinct values represent the *m* strata



Distinct values represent the *k* populations

Multiple Comparison Methods

- Bonferroni correction to the raw *p*-values
- Dunnett's two-tailed comparisons of the control group with all other groups
- Scheffe's multiple-comparison adjustment
- Sidák correction to the raw *p*-values
- Paired comparisons based on the studentized maximum modulus test
- Tukey's studentized range test
- Adjusted *p*-values from the simulated distribution

Specification of Comparisons

- DIFF=ALL requests all paired comparisons.
- DIFF=CONTROL <(*'string'* <...*'string'*>)> requests comparisons of the control curve with all other curves.
- To specify the control curve, you specify the quoted strings of formatted values that represent the curve in parentheses.

Unstratified Tests

The rank statistics (Klein and Moeschberger; 1997, Section 7.3) for testing H_0 versus H_1 have the form of a k-vector $\mathbf{v} = (v_1, v_2, \ldots, v_k)'$ with

$$v_j = \sum_{i=1}^{D} W(t_i)\left\{d_{ij} - \frac{n_{ij}d_i}{n_i}\right\}$$

and the estimated covariance matrix, $\mathbf{V} = (V_{jl})$, is given by

$$V_{jl} = \sum_{i=1}^{D} W^2(t_i)\left\{\frac{d_i s_i (n_i n_{il}\delta_{jl} - n_{ij}n_{il})}{n_i^2(n_i - 1)}\right\}$$

where δ_{jl} is 1 if $j = l$ and 0 otherwise. The term v_j can be interpreted as a weighted sum of observed minus expected numbers of failure under the null hypothesis of identical survival curves. The overall test statistic for homogeneity is $\mathbf{v}'\mathbf{V}^{-}\mathbf{v}$, where $\mathbf{V}^{-}$ denotes a generalized inverse of $\mathbf{V}$. This statistic is treated as having a chi-square distribution with degrees of freedom equal to the rank of $\mathbf{V}$ for the purposes of computing an approximate probability level.

Stratified Tests

Suppose the test is to be stratified on M levels of a set of STRATA variables. Based only on the data of the sth stratum ($s = 1 \ldots M$), let $\mathbf{v}_s$ be the test statistic (Klein and Moeschberger; 1997, Section 7.5) for the sth stratum, and let $\mathbf{V}_s$ be its covariance matrix. Let

$$\mathbf{v} = \sum_{s=1}^{M} \mathbf{v}_s$$

$$\mathbf{V} = \sum_{s=1}^{M} \mathbf{V}_s$$

A global test statistic is constructed as

$$\chi^2 = \mathbf{v}'\mathbf{V}\mathbf{v}$$

Under the null hypothesis, the test statistic has a χ^2 distribution with the same degrees of freedom as the individual test for each stratum.

Multiple-Comparison Adjustments

Let χ^2_r denote a chi-squared random variable with r degrees of freedom. Denote ϕ and Φ as the density function and the cumulative distribution function of a standard normal distribution, respectively. Let m be the number of comparisons; that is,

$$m = \begin{cases} \frac{k(k-1)}{2} & \text{DIFF} = \text{ALL} \\ k-1 & \text{DIFF} = \text{CONTROL} \end{cases}$$

For a two-sided test comparing the survival of the jth group with that of lth group, $1 \le j \ne l \le r$, the test statistic is

$$z^2_{jl} = \frac{(v_j - v_l)^2}{V_{jj} + V_{ll} - 2V_{jl}}$$

and the raw ***p***-value is

$$p = \Pr(\chi^2_1 > z^2_{jl})$$

Adjusted ***p***-values for various multiple-comparison adjustments are computed as follows:

Bonferroni Adjustment

$$p = \min\{1, m\Pr(\chi_1^2 > z_{jl}^2)\}$$

Dunnett-Hsu Adjustment

With the first group being the control, let $\mathbf{C} = (c_{ij})$ be the $(r-1) \times r$ matrix of contrasts; that is,

$$c_{ij} = \begin{cases} 1 & i = 1, \ldots, r-1, j = 2, \ldots, r \\ -1 & j = i+1, i = 2, \ldots, r \\ 0 & \text{otherwise} \end{cases}$$

Let $\mathbf{\Sigma} \equiv (\sigma_{ij})$ and $\mathbf{R} \equiv (r_{ij})$ be covariance and correlation matrices of $\mathbf{Cv}$, respectively; that is,

$$\mathbf{\Sigma} = \mathbf{CVC}'$$

and

$$r_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}}$$

The factor-analytic covariance approximation of Hsu (1992) is to find $\lambda_1, \ldots, \lambda_{r-1}$ such that

$$\mathbf{R} = \mathbf{D} + \boldsymbol{\lambda}\boldsymbol{\lambda}'$$

where $\mathbf{D}$ is a diagonal matrix with the jth diagonal element being $1 - \lambda_j$ and $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{r-1})'$. The adjusted ***p***-value is

$$p = 1 - \int_{-\infty}^{\infty} \phi(y) \prod_{i=1}^{r-1} \left[\Phi\left(\frac{\lambda_i y + z_{jl}}{\sqrt{1-\lambda_i^2}} \right) - \Phi\left(\frac{\lambda_i y - z_{jl}}{\sqrt{1-\lambda_i^2}} \right) \right] dy$$

which can be obtained in a DATA step as

$$p = \text{PROBMC}(\text{"DUNNETT2"}, z_{ij}, ., ., r-1, \lambda_1, \ldots, \lambda_{r-1}).$$

Scheffé Adjustment

$$p = \Pr(\chi^2_{r-1} > z^2_{jl})$$

Šidák Adjustment

$$p = 1 - \{1 - \Pr(\chi^2_1 > z^2_{jl})\}^m$$

SMM Adjustment

$$p = 1 - [2\Phi(z_{jl}) - 1]^m$$

which can also be evaluated in a DATA step as

$$p = 1 - \text{PROBMC}(\text{"MAXMOD"}, z_{jl}, ., ., m).$$

Tukey Adjustment

$$p = 1 - \int_{-\infty}^{\infty} r\phi(y)[\Phi(y) - \Phi(y - \sqrt{2}z_{jl})]^{r-1} dy$$

which can also be evaluated in a DATA step as

$$p = 1 - \text{PROBMC}(\text{"RANGE"}, \sqrt{2}z_{jl}, ., ., r).$$

LIFETEST Procedure

- General form of the LIFETEST procedure:

```
PROC LIFETEST DATA=SAS-data-set <options>;
    TIME variable <*censor(list)>;
    STRATA variable <(list)> <...variable <(list)>>
            </options>;
    TEST variables;
RUN;
```

• The simplest use of PROC LIFETEST is to request the nonparametric estimates of the survivor function for a sample of survival times. In such a case, only the PROC LIFETEST statement and the TIME statement are required. You can use the STRATA statement to divide the data into various strata. A separate survivor function is then estimated for each stratum, and tests of the homogeneity of strata are performed.

Hazard Function (riziková funkce)

- The hazard function
  - is the instantaneous risk or potential that an event will occur at time *t*, given that the individual has survived up to time *t*
  - takes the form number of events per interval of time
  - is a rate, not a probability, that ranges from zero to infinity.

Hazard Function

Conditional Probability

$$h(t) = \lim_{\Delta t \to 0} \frac{P(t \le T < t + \Delta t \mid T \ge t)}{\Delta t}$$

Interval of time

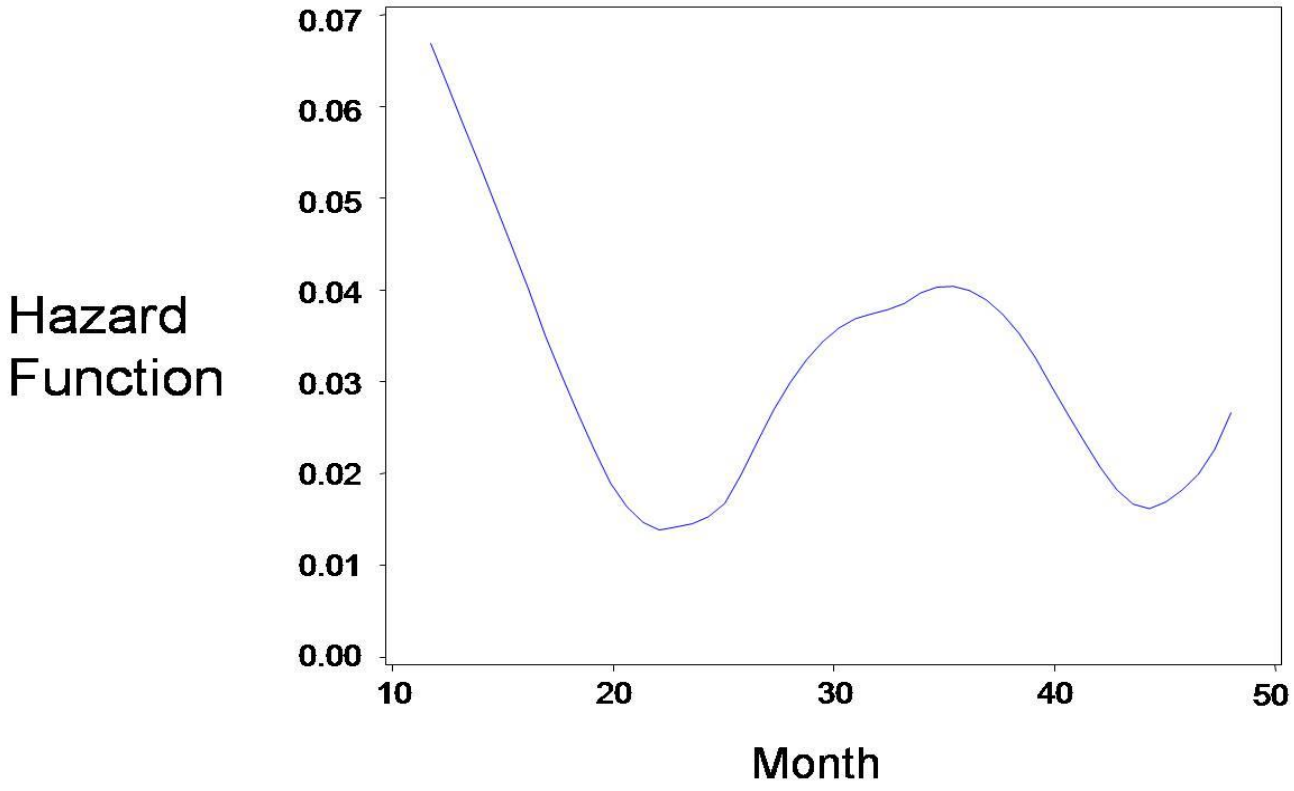
Instantaneous risk or potential
(okamžité riziko/potenciál)

Platí:

$$h(t) = \frac{f(t)}{S(t)} = -\frac{\partial}{\partial t} \ln(S(t))$$ kde $f(t)$ je hustota náhodné veličiny *T*.

$S(t) = \exp(-H(t))$ kde $H(t) = \int_0^t h(x)dx$ je tzv. kumulativní riziková funkce.

Hazard Function



10. Cox model

Survival Models

- Models in survival analysis
  - are written in terms of the hazard function
  - assess the relationship of predictor variables to survival time
  - can be parametric or nonparametric models.

Parametric versus Nonparametric Models

- Parametric models require that
  - the distribution of survival time is known
  - the hazard function is completely specified except for the values of the unknown parameters.
- Examples include the **Weibull model**, the **exponential model**, and the **log-normal model**.

Parametric versus Nonparametric Models

- Properties of nonparametric models are
  - the distribution of survival time is unknown
  - the hazard function is unspecified.
- An example is the **Cox proportional hazards model**.

Cox Proportional Hazards Model

$$h_i(t) = h_0(t)e^{\{\beta_1 X_{i1} + \ldots + \beta_k X_{ik}\}}$$

Baseline Hazard function – involves time but not predictor variables

Linear function of a set of predictor variables – does **not** involve time

Popularity of the Cox Model

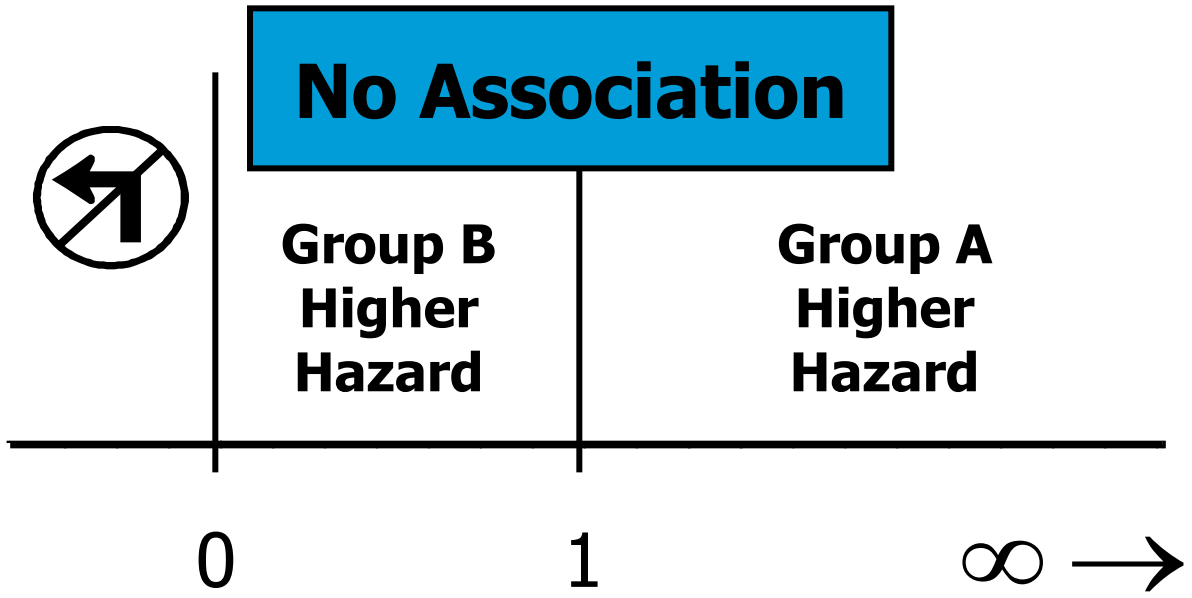
- The Cox proportional hazards model
  - provides the primary information desired from a survival analysis, hazard ratios and adjusted survival curves, with a minimum number of assumptions
  - is a robust model where the regression coefficients closely approximate the results from the correct parametric model.

Measure of Effect

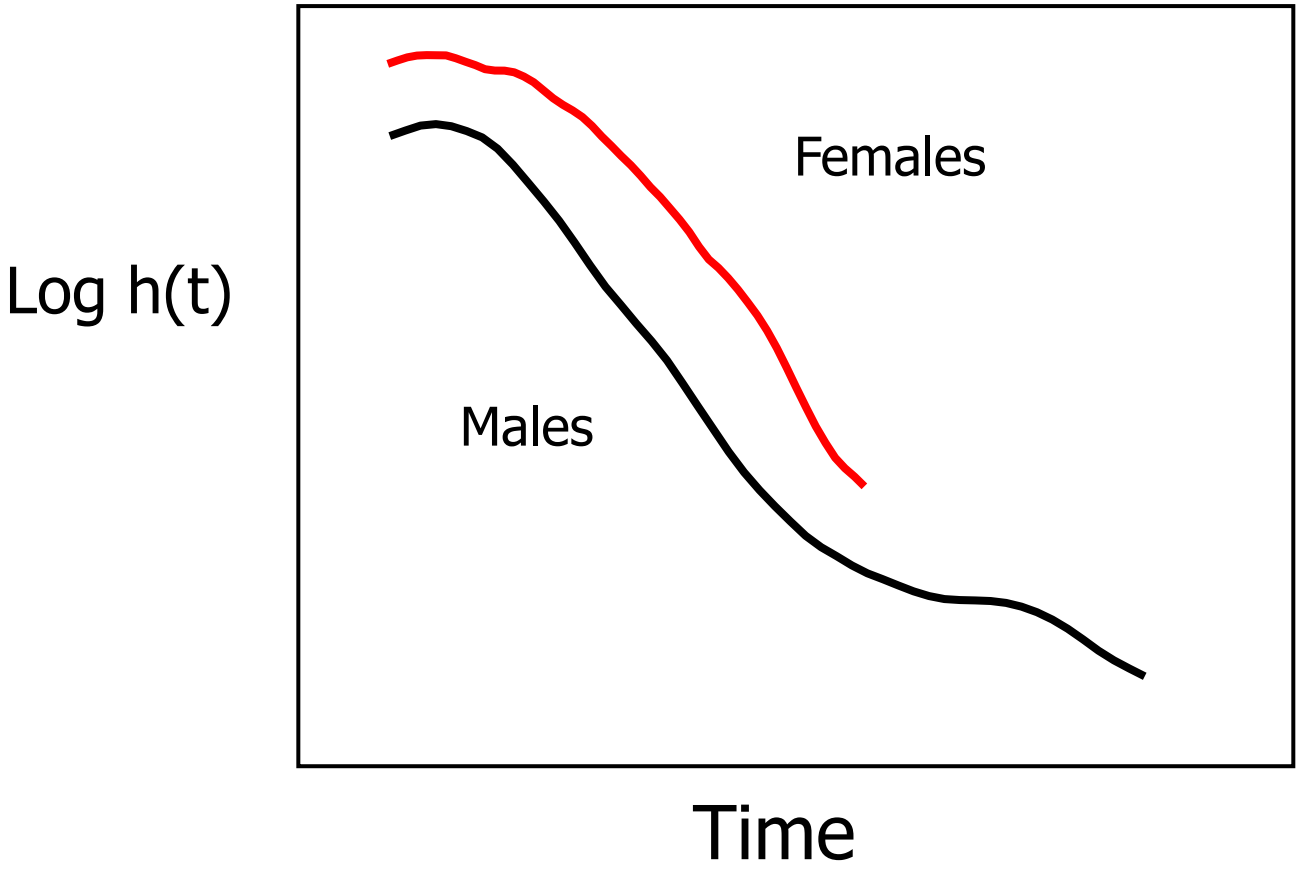
$$\text{Hazard ratio} = \frac{\text{hazard in group A}}{\text{hazard in group B}}$$

$$= e^{\hat{\beta}_i (X_{iA} - X_{iB})}$$

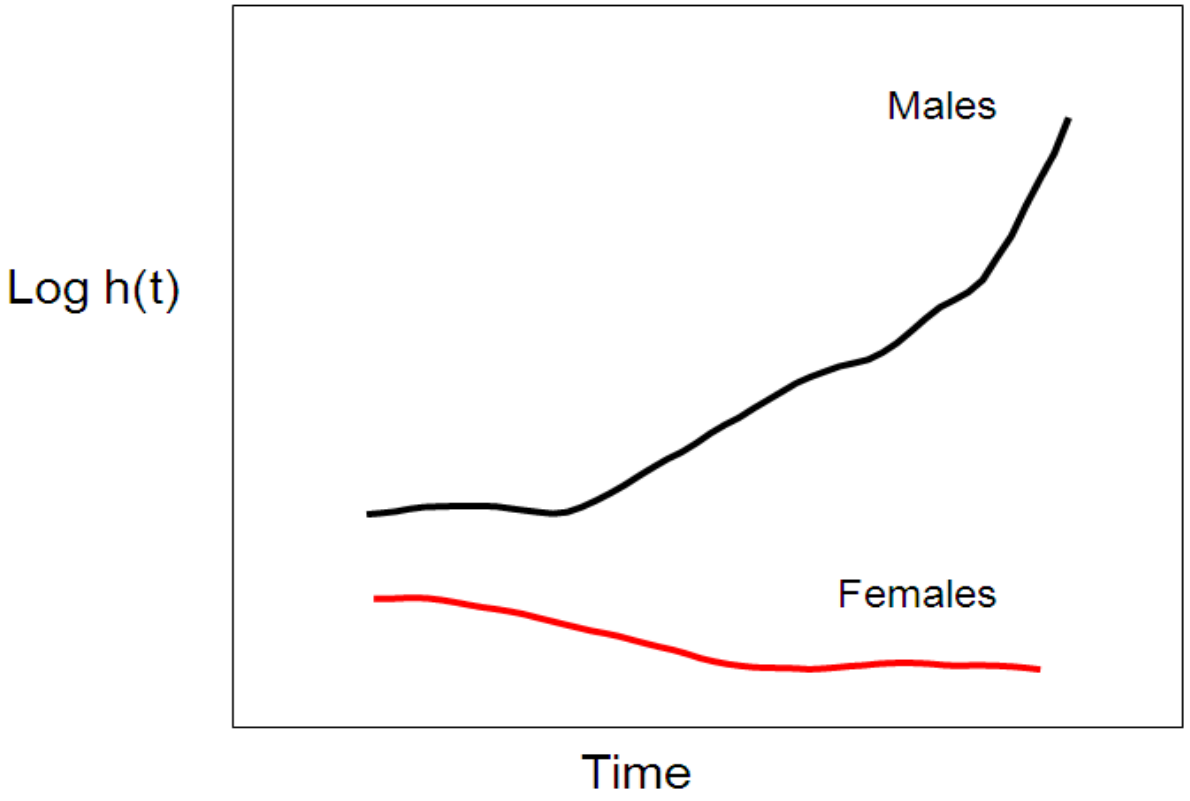
Properties of the Hazard Ratio



Proportional Hazards Assumption



Nonproportional Hazards



Cox model in credit scoring

Credit-scoring systems were built to answer the question, "How likely is a credit applicant to default by a given time in the future?" The methodology is to take a sample of previous customers and classify them into good or bad depending on their repayment performance over a given fixed period. Poor performance just before the end of this fixed period means that customer is classified as bad; poor performance just after the end of the period does not matter and the customer is classified as good. This arbitrary division can lead to less-thanrobust scoring systems. Also, if one wants to move from credit scoring to profit scoring, then it matters when a customer defaults. One asks not if an applicant will default but when will they default. This is a more difficult question to answer because there are lots of answers, not just the yes or no of the "if" question, but it is the question that survival analysis tools address when modeling the lifetime of equipment, constructions, and humans.

Zdroj: Thomas, Edelman, Crook – Credit scoring and its application.

Cox model in credit scoring

Using survival analysis to answer the "when" question has several advantages over standard credit scoring. For example,

- it deals easily with censored data, where customers cease to be borrowers (either by paying back the loan, death, changing lender) before they default;
- it avoids the instability caused by having to choose a fixed period to measure satisfactory performance;
- estimating when there is a default is a major step toward calculating the profitability of an applicant;
- these estimates will give a forecast of the default levels as a function of time, which is useful in debt provisioning;
- this approach may make it easier to incorporate estimates of changes in the economic climate into the scoring system.

Zdroj: Thomas, Edelman, Crook – Credit scoring and its application.

Cox model in credit scoring

Let *T* be the time until a loan defaults. Then there are three standard ways to describe the randomness of *T in survival analysis (Collett 1994): S(t), f(t)* and *h(t).*

Two of the commonest lifetime distributions are the negative exponential, which with parameters λ has $S(t) = e^{-\lambda t}$, $f(t) = \lambda e^{-\lambda t}$, and $h(t) = \lambda$, and the Weibull distribution, which with scale λ and shape k has $S(t) = e^{-(\lambda t)^k}$, $f(t) = k\lambda t^{k-1} e^{-(\lambda t)^k}$, and $h(t) = k\lambda^k t^{k-1}$. The former has no aging effect in that the default rate stays the same over time; the latter is more likely to default early on if $k < 1$, is more likely to default late on if $k > 1$, and becomes the negative exponential distribution if $k = 1$.

Zdroj: Thomas, Edelman, Crook – Credit scoring and its application.

Cox model in credit scoring

In standard credit scoring, one assumes that the application characteristics affect the probability of default. Similarly, in this survival analysis approach, we want models that allow these characteristics to affect the probability of when a customer defaults. Two models have found favor in connecting explanatory variables to failure times in survival analysis:

- proportional hazard models
- accelerated life models.

If $x = (x_1, \ldots, x_p)$ *are the application* (explanatory) characteristics, then an accelerated life model assumes that

$$S(t) = S_0(e^{\mathbf{w}\cdot\mathbf{x}}t) \quad \text{or} \quad h(t) = e^{\mathbf{w}\cdot\mathbf{x}}h_0(e^{\mathbf{w}\cdot\mathbf{x}}t)$$

where h_o and S_o are baseline functions, so the x can speed up or slow down the aging of the account. The proportional hazard assumes that

$$h(t) = e^{\mathbf{w}\cdot\mathbf{x}}h_0(t)$$

so the application variables x have a multiplier effect on the baseline hazard.

Zdroj: Thomas, Edelman, Crook – Credit scoring and its application.

Cox model in credit scoring

Cox (1972) pointed out that in proportional hazards one can estimate the weights *w* without knowing $h_0(t)$ using the ordering of the failure times and the censored times. If t_i and x_i are the failure (or censored) times and the application variables for each of the items under test, then the conditional probability that customer *i* defaults at time t_i given that *R(i)* are the customers still operating just before t_i is given by

$$\frac{\exp\{\mathbf{w}\cdot\mathbf{x}_i\}h_0(t)}{\sum_{k\in R(i)}\exp\{\mathbf{w}\cdot\mathbf{x}_k\}h_0(t)}=\frac{\exp\{\mathbf{w}\cdot\mathbf{x}_i\}}{\sum_{k\in R(i)}\exp\{\mathbf{w}\cdot\mathbf{x}_k\}}$$

which is independent of h_0.

Zdroj: Thomas, Edelman, Crook – Credit scoring and its application.

PHREG Procedure

```
PROC PHREG DATA=SAS-data-set <options>;
      CLASS variable <(options)><...variable <(options)>><|options>;
      MODEL  response<*censor(list)>=variables </options>;
      STRATA variable<(list)><…variable<(list)>> </options>;
     CONTRAST <'label>effect values<,.., effect values> <|options>;
      ASSESS keyword <|options>;
      HAZARDRATIO <'label> variable <|options>;
      <label:> TEST equation1 <,..., equationk> < |options>;
      WEIGHT variable</option;
      BASELINE <OUT=SAS-data-set><COVARIATES=
                    SAS-data-set><keyword=name…></options>;
      OUTPUT <OUT=SAS-data-set> <keyword=name…
                  keyword=name> </options>;
     programming statements;
RUN;
```

11. Měření kvality (síly) modelu, validace modelu.

How Good Is the Scorecard?

- And which one is the best?

- Combination of statistical measures and business objectives
  - Misclassification (Confusion) matrix
  - Scorecard strength measures

Misclassification

- Confusion matrix

		Predicted	
		Good	**Bad**
Actual	**Good**	True Positive	False Negative
	Bad	False Positive	True Negative

"Good"/"Bad" is above/below chosen cutoff.

  - Accuracy
    - (TP+TN)/total
  - Error rate
    - (FP+FN)/total
  - Sensitivity ; Specificity
    - (TP)/Actual Positives ; (TN)/Actual Negatives
  - Positive ; Negative predicted value
    - TP/predicted positives ; TN/predicted negatives

Want to max accuracy and min error rate.

Misclassification

- Confusion matrix
  - Acceptance of bads (FP)
  - Acceptance of goods (TP)
  - Decline Goods (FN)
  - Decline Bads (TN)

		Predicted	
		Good	**Bad**
Actual	**Good**	True Positive	False Negative
	Bad	False Positive	True Negative

Want to min rejection of goods and max rejection of bads.

Misclassification

- Approval rate: bad rate relationship
- Objective:
  - Minimize the rejection of goods or acceptance of bads
  - Best option for desired bad rate, approval rate
- Compare scorecards and cutoff choices.

"I'd rather approve some bads than reject good customers" vs "the cost of approving bads is too high, we can deal with PR".
Generate these stats for different cutoff rate choices and compare with base I.e. current approval and bad rates.
If several models are being compared, generate these for same bad rate or approval rate. I.e. choose different cutoffs to get same bad rate.

Misclassification: Oversampling

- Need to adjust for oversampling if have not done so before this step
- Sensitivity/specificity unaffected by oversampling
- Multiply cell counts by sample weights (π_0 and π_1)

		Predicted	
		Good	**Bad**
Actual	**Good**	n*(True P_s/Actual P_s)* π_1	n*(1-Sens)* π_1
	Bad	n*(1 - Spec)* π_0	n*(Spec)* π_0

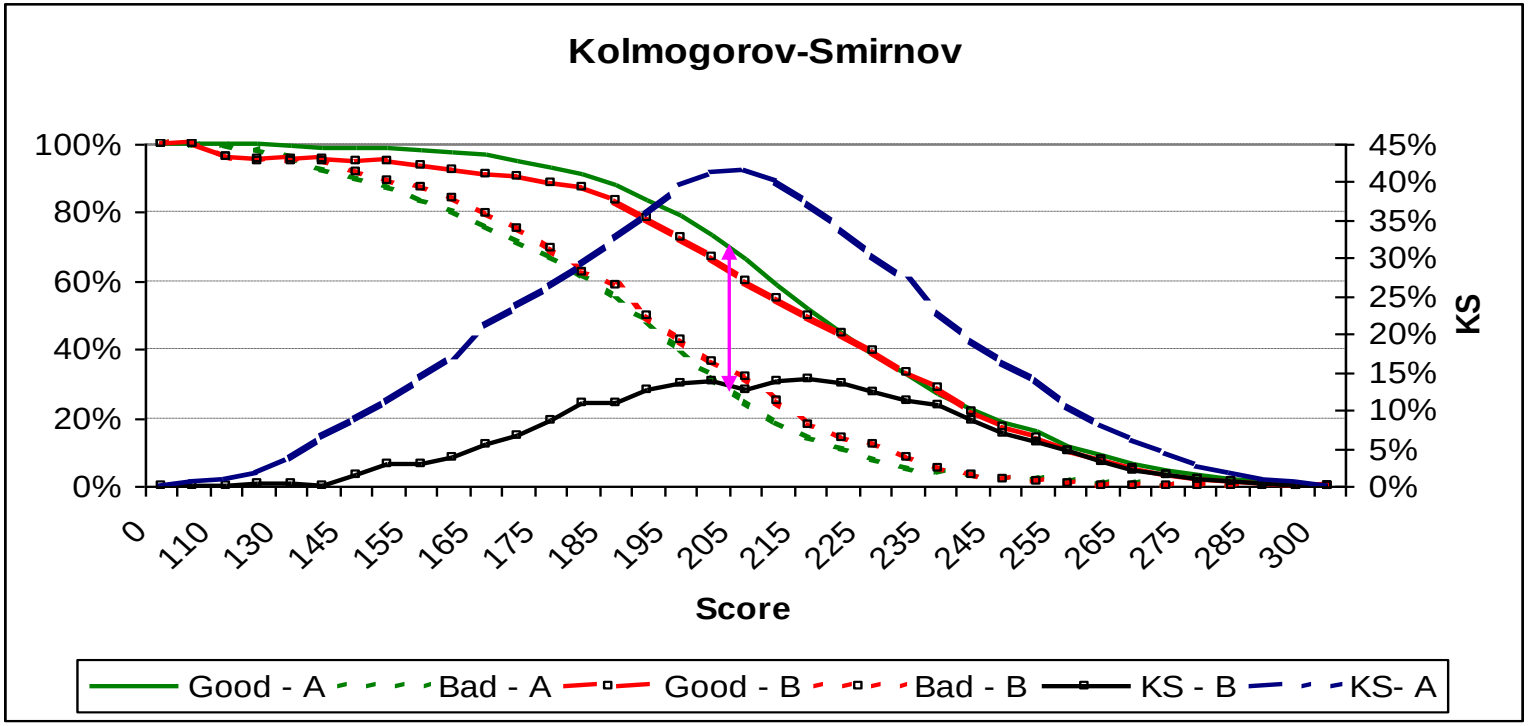
Scorecard Strength

- Akaike's Information Criterion (AIC)
- Schwartz Bayesian Criterion (SBC)
  - -(score test statistic) + penalty term
    - Penalty term = (k + 1). Ln(n)
    - k = number of variables
    - n = sample size

Penalise for adding parameters to the model … Smaller values are better.

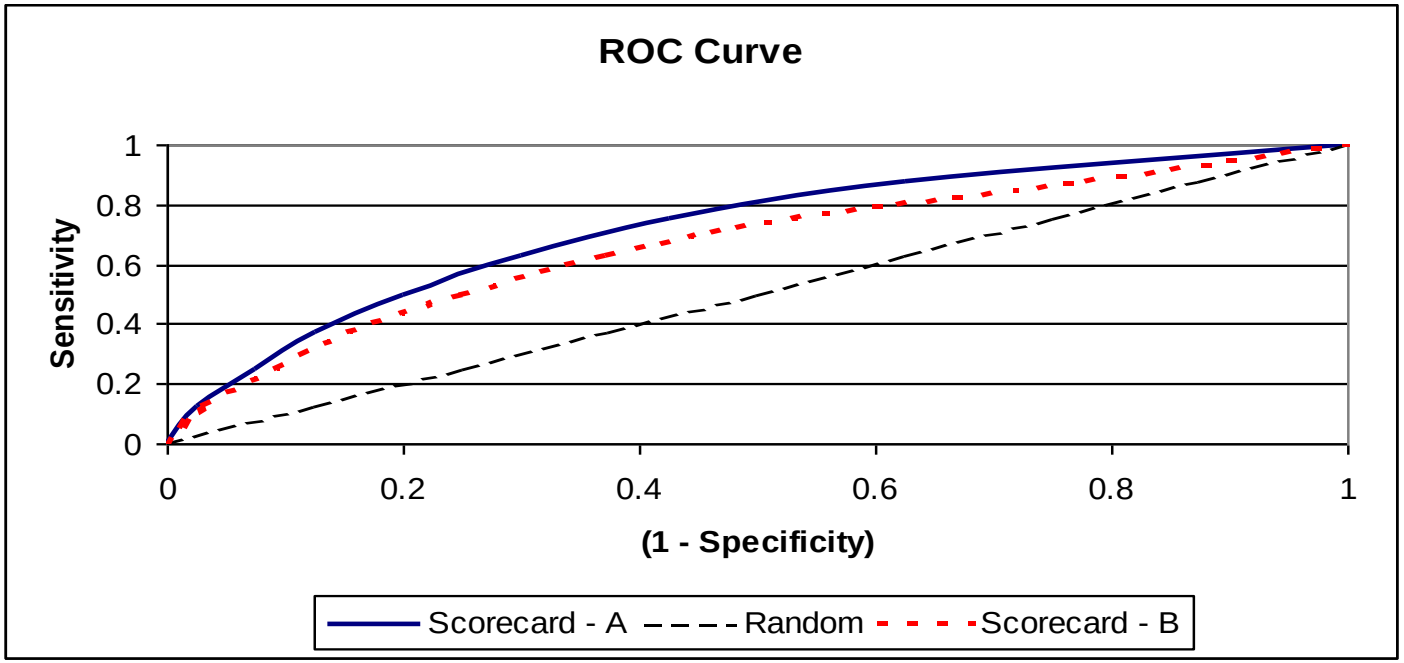
KS Statistic

- Max difference between cumulative distributions of goods and bads across score ranges



Scorecard Strength

- C - Statistic
  - Area under the ROC curve, Wilcoxon-Mann-Whitney test.



You may be wondering where the name "Receiver Operating Characteristic" came from. ROC analysis is part of a field called "Signal Detection Theory" developed during World War II for the analysis of radar images. Radar operators had to decide whether a blip on the screen represented an enemy target, a friendly ship, or just noise. Signal detection theory measures the ability of radar receiver operators to make these important distinctions. Their ability to do so was called the Receiver Operating Characteristics. It was not until the 1970's that signal detection theory was recognized as useful for interpreting medical test results.

Scorecard Strength

- **Gains Chart:** Cumulative Positive Predicted Value versus Distribution of Predicted Positives (*depth*)
- **Lift/concentration curve:** Sensitivity versus Depth
  - Lift = Positive predicted value / % positives in the sample
- Misclassification costs (losses assigned to false +positive and –positive)
- Bayes Rule (minimizes expected cost)
- Cost ratio (at what cutoff do I break even based on prior bad rate I.e. if bad odds are 9:1, you need a cutoff where 1 bad and 9 goods)
- Somers' D, Gamma, Tau-a

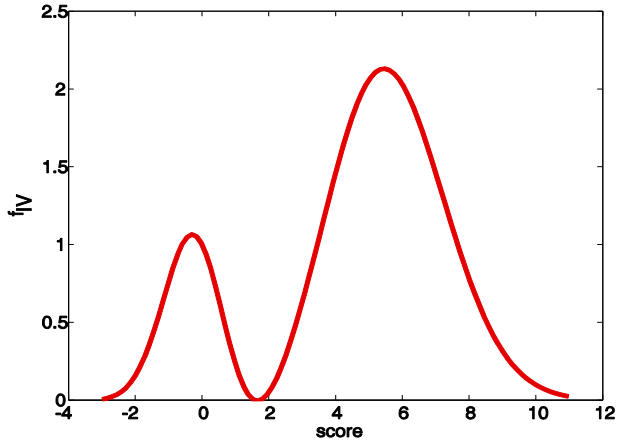
Information value

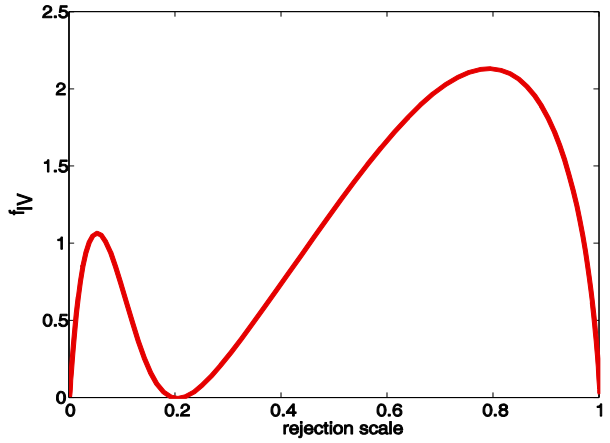
- The special case of Kullback-Leibler divergence given by:

$$I_{val} = \int_{-\infty}^{\infty} f_{IV}(x)dx \quad \text{where} \quad f_{IV}(x) = (f_1(x) - f_0(x)) \ln\left(\frac{f_1(x)}{f_0(x)}\right)$$

f_0, f_1 are densities of scores of bad and good clients.

The example of $f_{IV}(x)$ for 10% of bad clients with $f_0 \sim N(0,1)$ and 90% of good clients with $f_1 \sim N(4,2)$





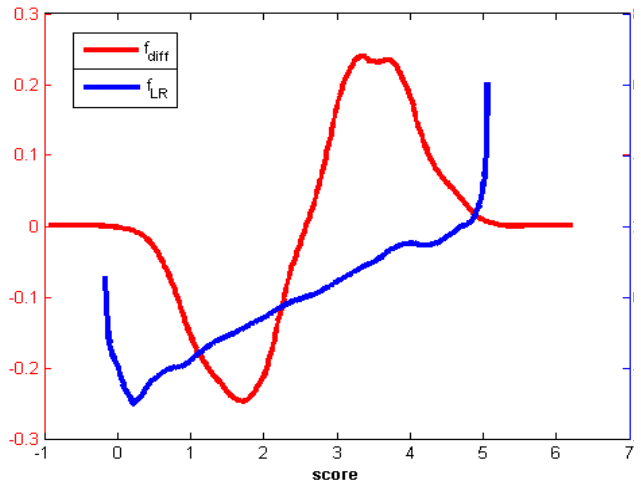
Information value

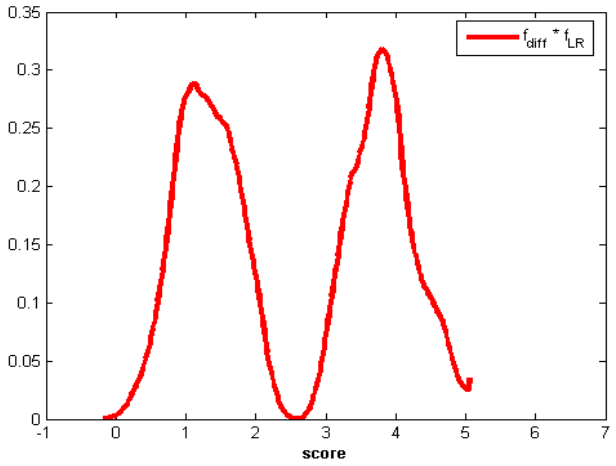
- Informační hodnota (I_{val}) – spojitý případ (Divergence):

$$I_{val} = \int_{-\infty}^{\infty} \left(f_{GOOD}(x) - f_{BAD}(x)\right) \ln\left(\frac{f_{GOOD}(x)}{f_{BAD}(x)}\right) dx$$

$$f_{diff}(x) = f_{GOOD}(x) - f_{BAD}(x)$$

$$f_{LR}(x) = \ln\left(\frac{f_{GOOD}(x)}{f_{BAD}(x)}\right)$$

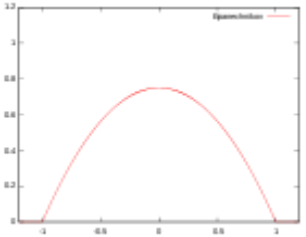




Information value

➢ Informační hodnota (I_{val}) – diskretizovaný spojitý případ:

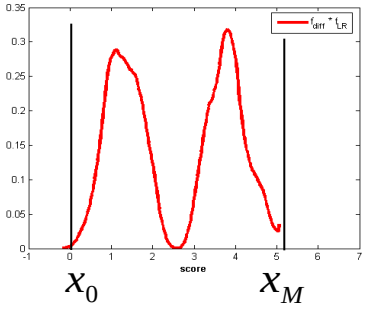
• Nahradíme hustotu jejím jádrovým odhadem a spočteme integrál numericky (např. pomocí složeného lichoběžníkového pravidla).



• S použitím Epanečnikova jádra, $K(x) = \frac{3}{4}\left(1-x^2\right)\cdot I\left(x \in [-1,1]\right)$,
a optimální šířky vyhlazovacího okna $h_{OS,k}$ dostaneme

$$\tilde{f}_{IV}(x) = \left(\tilde{f}_{GOOD}(x, h_{OS,2}) - \tilde{f}_{BAD}(x, h_{OS,2})\right)\ln\left(\frac{\tilde{f}_{GOOD}(x, h_{OS,2})}{\tilde{f}_{BAD}(x, h_{OS,2})}\right)$$

• Pro daných M+1 bodů $x_0, \ldots, x_M$ dostáváme



$$I_{val} = \frac{x_M - x_0}{2M}\left(\tilde{f}_{IV}(x_0) + 2\sum_{i=1}^{M-1}\tilde{f}_{IV}(x_i) + \tilde{f}_{IV}(x_M)\right)$$

Information value

- 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 $g_i > 0, b_i > 0 \;\; \forall i$

• 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

Information value

- Informační hodnota pro 2 příklady scoringových modelů:

SC 1:

decile	# cleints	# bad clients	#good	% bad [1]	% good [2]	[3] = [2] - [1]	[4] = [2] / [1]	[5] = ln[4]	[6] = [3] * [5]	cum. [6]
1	100	35	65	35,0%	7,2%	-0,28	0,21	-1,58	0,44	0,44
2	100	16	84	16,0%	9,3%	-0,07	0,58	-0,54	0,04	0,47
3	100	8	92	8,0%	10,2%	0,02	1,28	0,25	0,01	0,48
4	100	8	92	8,0%	10,2%	0,02	1,28	0,25	0,01	0,49
5	100	7	93	7,0%	10,3%	0,03	1,48	0,39	0,01	0,50
6	100	6	94	6,0%	10,4%	0,04	1,74	0,55	0,02	0,52
7	100	6	94	6,0%	10,4%	0,04	1,74	0,55	0,02	0,55
8	100	5	95	5,0%	10,6%	0,06	2,11	0,75	0,04	0,59
9	100	5	95	5,0%	10,6%	0,06	2,11	0,75	0,04	0,63
10	100	4	96	4,0%	10,7%	0,07	2,67	0,98	0,07	0,70
All	1000	100	900					Info. Value	0,70	

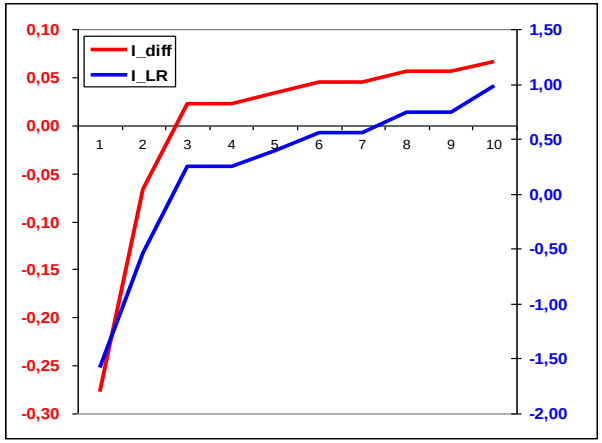
SC 2:

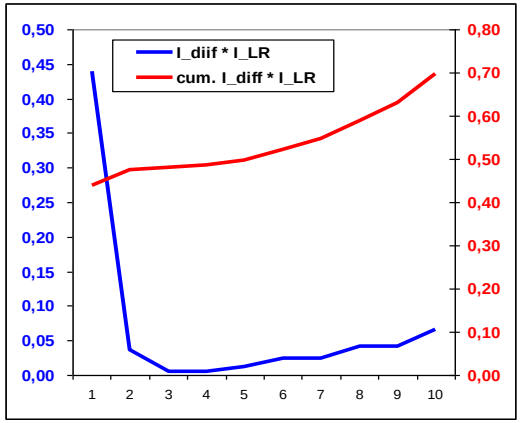
decile	# cleints	# bad clients	#good	% bad [1]	% good [2]	[3] = [2] - [1]	[4] = [2] / [1]	[5] = ln[4]	[6] = [3] * [5]	cum. [6]
1	100	20	80	20,0%	8,9%	-0,11	0,44	-0,81	0,09	0,09
2	100	18	82	18,0%	9,1%	-0,09	0,51	-0,68	0,06	0,15
3	100	17	83	17,0%	9,2%	-0,08	0,54	-0,61	0,05	0,20
4	100	15	85	15,0%	9,4%	-0,06	0,63	-0,46	0,03	0,22
5	100	12	88	12,0%	9,8%	-0,02	0,81	-0,20	0,00	0,23
6	100	6	94	6,0%	10,4%	0,04	1,74	0,55	0,02	0,25
7	100	4	96	4,0%	10,7%	0,07	2,67	0,98	0,07	0,32
8	100	3	97	3,0%	10,8%	0,08	3,59	1,28	0,10	0,42
9	100	3	97	3,0%	10,8%	0,08	3,59	1,28	0,10	0,52
10	100	2	98	2,0%	10,9%	0,09	5,44	1,69	0,15	0,67
All	1000	100	900					Info. Value	0,67	

Information value

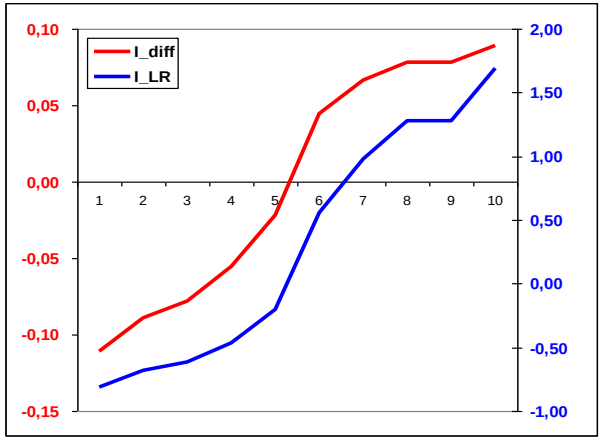
- Označíme-li $I_{diff_i} = \left(\frac{g_i}{n} - \frac{b_i}{m} \right)$ $I_{LR_i} = \ln\left(\frac{g_i m}{b_i n} \right)$, dostáváme:

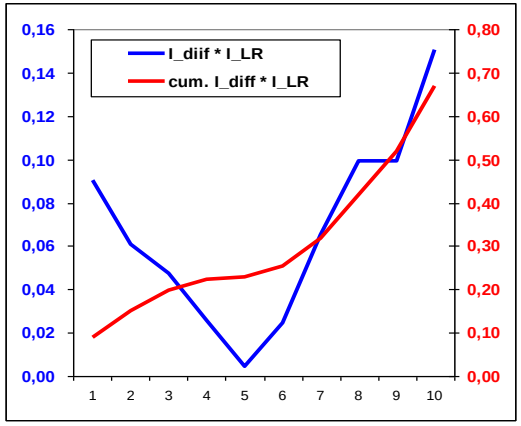
- **SC 1:**





- **SC 2:**





K-S	= 0.34
Gini	= 0.42
$Lift_{20\%}$	= 2.55
$Lift_{50\%}$	= 1.48
I_{val}	= 0.70
$I_{val20\%}$	= 0.47
$I_{val50\%}$	= 0.50
K-S	= 0.36
Gini	= 0.42
$Lift_{20\%}$	= 1.90
$Lift_{50\%}$	= 1.64
I_{val}	= 0.67
$I_{val20\%}$	= 0.15
$I_{val50\%}$	= 0.23

I_{val} for normally distributed scores

- Assume that the scores of good and bad clients are normally distributed, i.e. we can write their densities as

$$f_1(x) = \frac{1}{\sigma_g \sqrt{2\pi}} e^{-\frac{(x-\mu_g)^2}{2\sigma_g^2}} \qquad f_0(x) = \frac{1}{\sigma_b \sqrt{2\pi}} e^{-\frac{(x-\mu_b)^2}{2\sigma_b^2}}$$

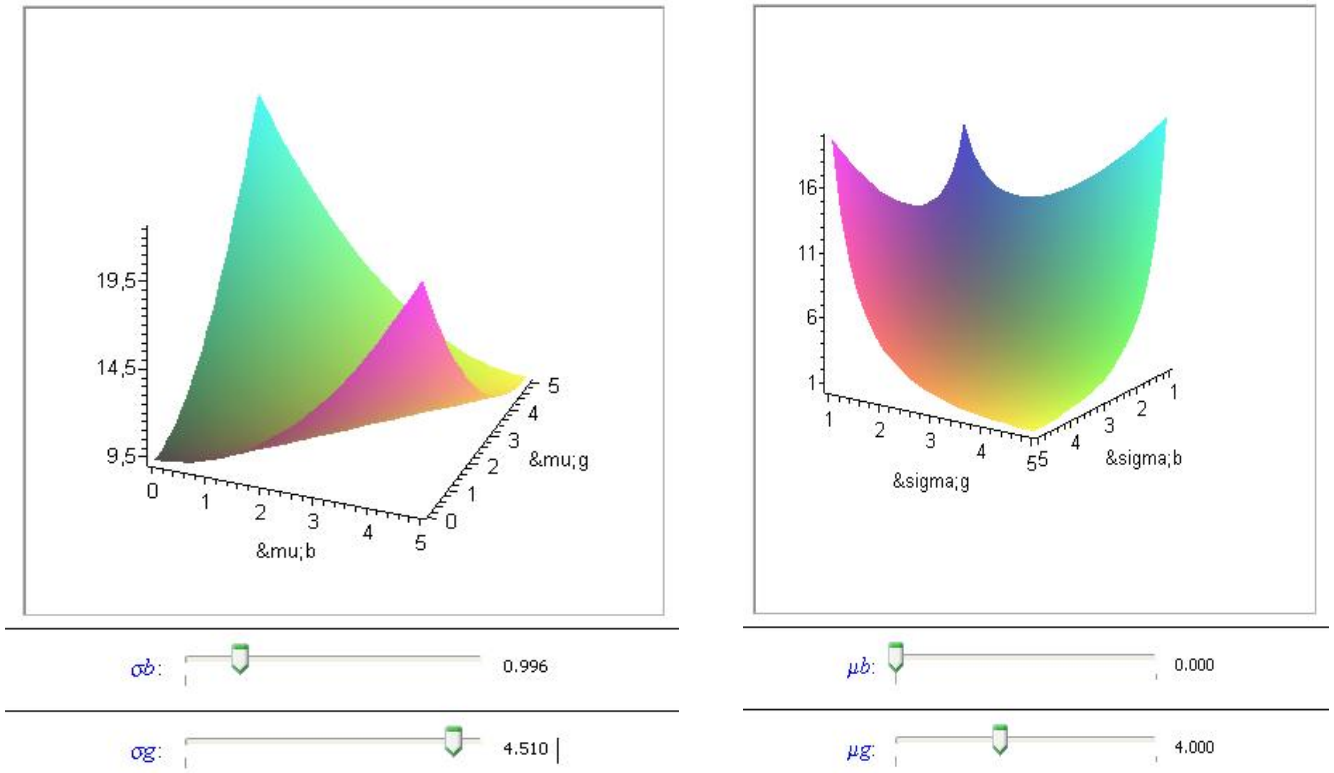
- Assume that standard deviations are equal to a common value σ:

$$I_{val} = D^2 \qquad \text{where} \quad D = \frac{\mu_g - \mu_b}{\sigma}$$

- Generally (i.e. without assumption of equality of standard deviations):

$$I_{val} = (A+1)D^{*2} + A - 1, \quad A = \frac{1}{2}\left(\frac{\sigma_b^2}{\sigma_g^2} + \frac{\sigma_g^2}{\sigma_b^2}\right) \qquad \text{where} \quad D^* = \frac{\mu_g - \mu_b}{\sqrt{\sigma_g^2 + \sigma_b^2}}$$

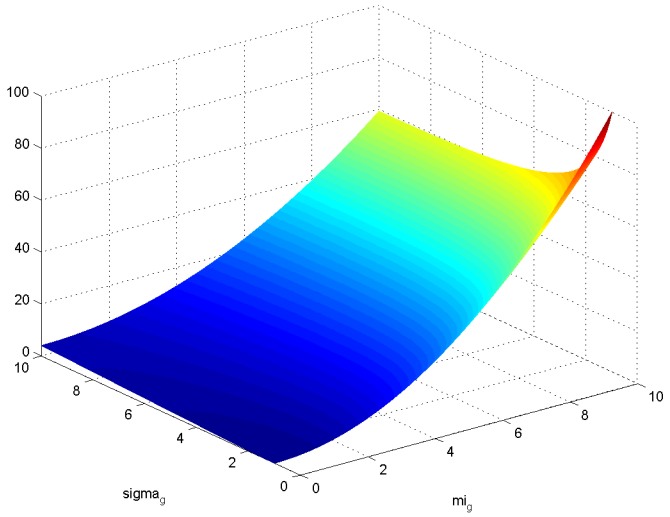
I_{val} for normally distributed scores

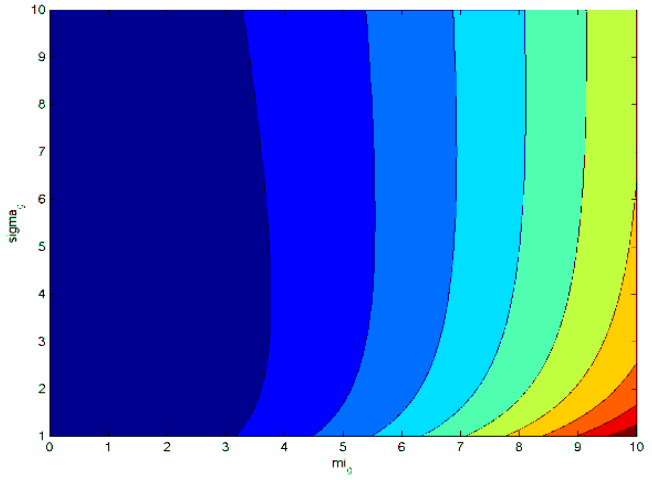


- We can see a quadratic dependence on difference of means.
- I_{val} takes quite high values when both variances are approximately equal and smaller or equal to 1, and it grows to infinity if ratio of the variances tends to infinity or is nearby zero.

I_{val} for normally distributed scores

- **I_{val}:** $\mu_b = 0$, $\sigma_b^2 = 1$





- Velmi silná závislost na μ_g . Navíc hodnota I_{val} míří velmi rychle k nekonečnu pokud se σ_g^2 blíží nule.

Empirical estimate of I_{val}

The main idea of this approach is to replace unknown densities by their empirical estimates. Let's have n score values, of which n_0 score values s_{0_i}, $i = 1, \ldots, n_0$ for bad clients and n_1 score values s_{1_j}, $j = 1, \ldots, n_1$ for good clients and denote L (resp. H) as the minimum (resp. maximum) of all values. Let's divide the interval $[L, H]$ up to r equal subintervals $[q_0, q_1], (q_1, q_2], \ldots, (q_{r-1}, q_r]$, where $q_0 = L, q_r = H$. Set

$$n_{0_j} = \sum_{i=1}^{n_0} I\left(s_{0_i} \in (q_{j-1}, q_j]\right)$$
$$n_{1_j} = \sum_{i=1}^{n_1} I\left(s_{1_i} \in (q_{j-1}, q_j]\right), \quad j = 1, \ldots, r$$

observed counts of bad or good clients in each interval. Then the empirical Information value is calculated by

$$\widehat{I}_{val,DEC} = \sum_{j=1}^{r} \left(\frac{n_{1_j}}{n_1} - \frac{n_{0_j}}{n_0}\right) \ln\left(\frac{n_{1_j} n_0}{n_{0_j} n_1}\right).$$

Empirical estimate of I_{val}

- However in practice, there could occur computational problems. The Information value index becomes infinite in cases when some of n_{oj} or n_{ij} are equal to 0. When this arises there are numerous practical procedures for preserving finite results. For example one can replace the zero entry of numbers of goods or bads by a minimum constant of say 0.0001. Choosing of the number of bins is also very important. In the literature and also in many applications in credit scoring, the value **r=10** is preferred.

Empirical estimate with supervised interval selection

- We want to avoid zero values of n_{0j} or n_{1j} .
- We propose to require to have at least k, where k is a positive integer, observations of scores of both good and bad client in each interval.
- Set

$$q_0 = L - 1$$
$$q_i = \widehat{F_0^{-1}}\left(\frac{k \cdot i}{n_0}\right), i = 1, \ldots, \lfloor \frac{n_0}{k} \rfloor$$
$$q_{\lfloor \frac{n_0}{k} \rfloor + 1} = H,$$

where $\widehat{F_0^{-1}}(\cdot)$ is the empirical quantile function appropriate to the empirical cumulative distribution function of scores of bad clients.

Empirical estimate with supervised interval selection

- Usage of quantile function of scores of bad clients is motivated by the assumption, that number of bad clients is less than number of good clients.
- If n_o is not divisible by k, it is necessary to adjust our intervals, because we obtain number of scores of bad clients in the last interval, which is less than k. In this case, we have to merge the last two intervals.
- Furthermore we need to ensure, that the number of scores of good clients is as required in each interval
- To do so, we compute n_{1j} for all actual intervals. If we obtain $n_{1j} < k$ for j^{th} interval, we merge this interval with its neighbor on the right side.
- This can be done for all intervals except the last one. If we have $n_{1j} < k$ for the last interval, than we have to merge it with its neighbor on the left side, i.e. we merge the last two intervals.

Empirical estimate with supervised interval selection

- Very important is the choice of *k*. If we choose too small value, we get overestimted value of the Information value, and vice versa. As a reasonable compromise seems to be adjusted square root of number of bad clients given by

$$k = \lceil \sqrt{n_0} \rceil$$

- The estimate of the Information value is given by

$$\widehat{I}_{val,ESIS} = \sum_{j=1}^{r} \left(\frac{n_{1_j}}{n_1} - \frac{n_{0_j}}{n_0} \right) \ln \left(\frac{n_{1_j} n_0}{n_{0_j} n_1} \right)$$

where n_{oj} and n_{1j} correspond to observed counts of good and bad clients in intervals created according to the described procedure.

Simulation results

- Consider n clients, $100p_B$% of bad clients with $f_0 : N(\mu_0, \sigma_0)$ and $100(1-p_B)$% of good clients with $f_1 : N(\mu_1, \sigma_1)$.
- Because of normality we know $I_{val} = \left(\frac{\mu_1 - \mu_0}{\sigma} \right)^2$.
- Consider following values of parameters:
  - n = 100 000 , n = 1000
  - $\mu_0 = 0$
  - $\sigma_0 = \sigma_1 = 1$
  - $\mu_1 = 0.5, 1, 1.5$
  - $p_B = 0.02, 0.05, 0.1, 0.2$

Simulation results

1) Scores of bad and good clients were generated according to given parameters.

2) Estimates $\hat{I}_{val,DEC}$, $\hat{I}_{val,KERN}$, $\hat{I}_{val,ESIS}$ were computed.

3) Square errors were computed.

4) Steps 1)-3) were repeated one thousand times.

5) MSE was computed.

Simulation results

n=100000, $\mu_1 - \mu_0$ = 0.5				
MSE	p_B			
	0.02	0.05	0.1	0.2
IV_decil	0,000546	0,000310	0,000224	0,000168
IV_kern	0,000487	0,000232	0,000131	0,000076
IV_esis	0,000910	0,000384	0,000218	0,000127

n=1000, $\mu_1 - \mu_0$ = 0.5				
MSE	p_B			
	0.02	0.05	0.1	0.2
IV_decil	0,025574	0,040061	0,026536	0,009074
IV_kern	0,038634	0,017547	0,009281	0,004737
IV_esis	0,038331	0,021980	0,016280	0,008028

n=100000, $\mu_1 - \mu_0$ = 1.0				
MSE	p_B			
	0.02	0.05	0.1	0.2
IV_decil	0,006286	0,004909	0,004096	0,002832
IV_kern	0,003396	0,001697	0,001064	0,000646
IV_esis	0,002146	0,000973	0,000477	0,000568

n=1000, $\mu_1 - \mu_0$= 1.0				
MSE	p_B			
	0.02	0.05	0.1	0.2
IV_decil	0,186663	0,084572	0,043097	0,029788
IV_kern	0,117382	0,072381	0,045344	0,032131
IV_esis	0,150881	0,071088	0,036503	0,023609

n=100000, $\mu_1 - \mu_0$ = 1.5				
MSE	p_B			
	0.02	0.05	0.1	0.2
IV_decil	0,056577	0,048415	0,034814	0,020166
IV_kern	0,019561	0,010789	0,006796	0,004862
IV_esis	0,013045	0,008134	0,007565	0,027943

n=1000, $\mu_1 - \mu_0$ = 1.5				
MSE	p_B			
	0.02	0.05	0.1	0.2
IV_decil	1,663859	1,037778	0,535180	0,200792
IV_kern	0,529367	0,349783	0,266912	0,196856
IV_esis	0,609193	0,352151	0,172931	0,194676

- worst
- average
- best performance

Adjusted empirical estimate with supervised interval selection (AESIS)

- Je zřejmé, že volba parametru *k* je zcela zásadní. Otázkou tedy je:
  - Je volba $k = \lceil \sqrt{n_0} \rceil$ optimální (vzhledem k MSE)?
  - Jaký vliv na optimální k má n_0?
  - A jaký vliv, pokud vůbec, má rozdíl středních hodnot $\mu_1 - \mu_0$?

Simulation results

- Consider 10000 clients, $100p_B$% of bad clients with $f_0 : N(\mu_0,1)$ and $100(1-p_B)$% of good clients with $f_1 : N(\mu_1,1)$. Set $\mu_0 = 0$ and consider $\mu_1 = 0.5, 1 \text{ and } 1.5$, $p_B = 0.02, 0.05, 0.1 \text{ and } 0.2$

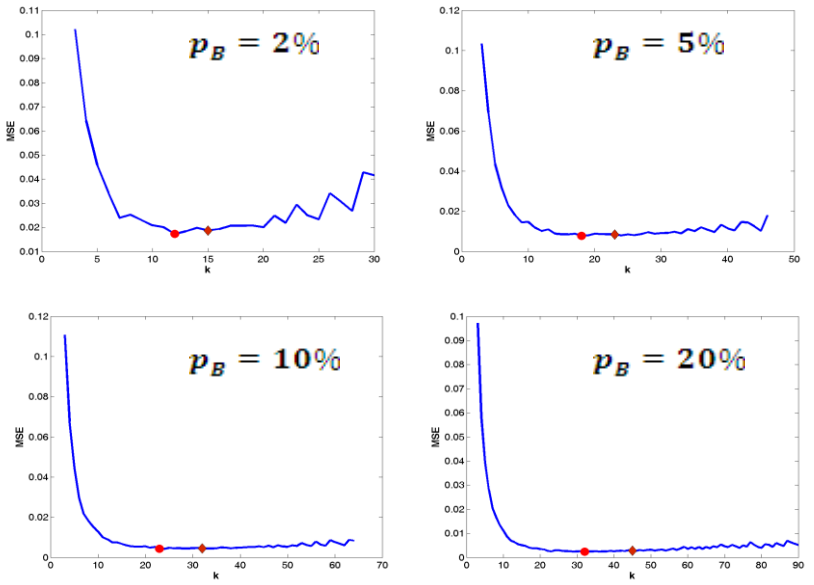
$$MSE = E((\hat{I}_{val} - I_{val})^2) \Rightarrow k_{MSE} = \underset{k}{argmin}\ MSE.$$

k_{MSE}		p_B			
		0.02	0.05	0.1	0.2
$\lvert\mu_1 - \mu_0$	0.5	29	42	62	84
	1	12	18	23	32
	1.5	6	9	8	9
$k = \lceil\sqrt{n_0}\rceil$		15	23	32	45

avg. # of bins		p_B			
		0.02	0.05	0.1	0.2
$\lvert\mu_1 - \mu_0$	0.5	8,00	13,00	18,00	24,90
	1	18,00	28,80	42,76	51,88
	1.5	33,62	50,20	95,96	127,67

Simulation results

- **Dependence of MSE on *k*, $\mu_1 - \mu_0 = 1$.**



➢ The highlighted circles correspond to values of *k*, where minimal value of the *MSE* is obtained. The diamonds correspond to values of *k* given by $k = \lceil \sqrt{n_0} \rceil$.

$$k = \left\lceil \frac{\frac{2}{3}\sqrt{p_B \cdot n} + 2}{|\widehat{\mu_1} - \widehat{\mu_0}|^{1.4}} \right\rceil$$

$k = \left\lceil \frac{\frac{2}{3}\sqrt{p_B \cdot n} + 2}{\lvert\widehat{\mu_1} - \widehat{\mu_0}\rvert^{1.4}} \right\rceil$		p_B			
		0.02	0.05	0.1	0.2
$\mu_1 - \mu_0$	0.5	31	45	61	84
	1	12	17	24	32
	1.5	7	10	14	19

k_{MSE}		p_B			
		0.02	0.05	0.1	0.2
$\mu_1 - \mu_0$	0.5	29	42	62	84
	1	12	18	23	32
	1.5	6	9	8	9

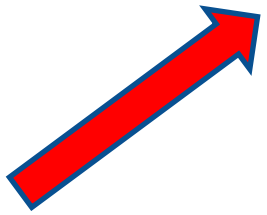
where $\widehat{\mu_1}$ and $\widehat{\mu_0}$ are suitable estimates of means of scores of good and bad clients, $p_B = \frac{n_0}{n}$ is the proportion of bad clients.

$$\hat{I}_{val,AESIS}$$

ESIS.1

- Algorithm for the modified ESIS:

1) $\mathbf{q} = []$
2) $q_{j1} = F_1^{-1}\left(\frac{k}{n_1}\right) \qquad q_{j0} = F_0^{-1}\left(\frac{k}{n_0}\right)$
3) $s_{\max} = \max(q_{j1}, q_{j0})$
4) Add $s_{\max}$ to the sequence, i.e. $\mathbf{q} = [\mathbf{q}, s_{\max}]$
5) Erase all scores $\leq s_{\max}$
6) While n_0 and n_1 are greater than 2*k, repeat step 2) – 5)
7) $\mathbf{q} = [\min(score) - 1, \mathbf{q}, \max(score)]$

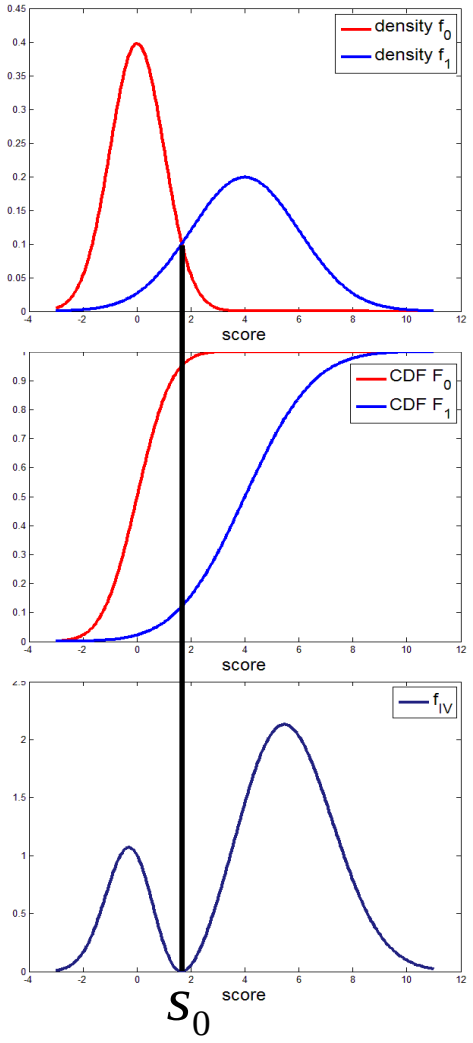


$\hat{I}_{val,ESIS1}$

where $k = \lceil \sqrt{n_0} \rceil$

ESIS.2

- U původního ESIS často dochází ke slučování vypočtených intervalů ve druhé fázi algoritmu.
- Pro výpočet se používá jen $F_0^{-1}(.)$.
- Aby byla splněna podmínka $n_1 > k$, je zřejmě nutné, aby hranice prvního intervalu byla větší než $F_1^{-1}\left(\frac{k}{n_1}\right)$.
- To vede k myšlence použít ke konstrukci intervalů nejprve $F_1^{-1}(.)$ a následně, od nějaké hodnoty skóre $F_0^{-1}(.)$.
- Jako vhodná hodnota skóre pro tento účel se jeví hodnota s_o, ve které se protínají hustoty skóre, rozdíl distribučních funkcí skóre nabývá své maximální hodnoty a také platí, že funkce f_{IV} nabývá nulové hodnoty.



Point of intersection of densities

||

Point of maximal difference of CDFs

||

Point of zero value of f_{IV}

ESIS.2

- Algorithm for the modified ESIS:

1) $s_0 = \arg\max_s |F_1 - F_0|$

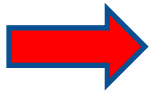
2) $q_{1_j} = F_1^{-1}\left(\frac{j \cdot k}{n_1}\right),\ j = 1, \ldots, \left\lfloor \frac{n_1}{k} \cdot F_1(s_0) \right\rfloor$

3) $q_{0_j} = F_0^{-1}\left(\frac{j \cdot k}{n_0}\right),\ j = \left\lceil \frac{n_0}{k} \cdot F_0(s_0) \right\rceil, \ldots, \left\lfloor \frac{n_0}{k} \right\rfloor - 1$

4) $\mathbf{q} = [\min(score) - 1, \mathbf{q}_1, \mathbf{q}_0, \max(score) + 1]$

5) Merge intervals given by $\mathbf{q}_1$ where number of bads is less than k.

6) Merge intervals given by $\mathbf{q}_0$ where number of goods is less than k.



$\hat{I}_{val,ESIS\,2}$

where $k = \lceil \sqrt{n_0} \rceil$

AESIS.2 – Simulation results

- Consider 1000, 10000 and 100000 clients, $100p_B$% of bad clients with $f_0 : N(\mu_0,1)$ and $100(1-p_B)$% of good clients with $f_1 : N(\mu_1,1)$. Set $\mu_0 = 0$, $\mu_1 = 0.5, 1$ and 1.5 and consider $p_B = 0.02, 0.05, 0.1$ and 0.2.

$$MSE = E((\hat{I}_{val} - I_{val})^2) \Rightarrow k_{MSE} = \underset{k}{argmin}\ MSE.$$

$n = 1000$

k_{MSE}		p_B = 0.02	0.05	0.1	0.2
$\mu_1 - \mu_0$	0.5	15	19	22	45
	1	3	8	11	16
	1.5	2	3	6	7
$k = \lceil \sqrt{n_0} \rceil$		5	8	10	15

$n = 10000$

k_{MSE}		p_B = 0.02	0.05	0.1	0.2
$\mu_1 - \mu_0$	0.5	29	51	77	112
	1	15	24	28	45
	1.5	6	11	11	14
$k = \lceil \sqrt{n_0} \rceil$		15	23	32	45

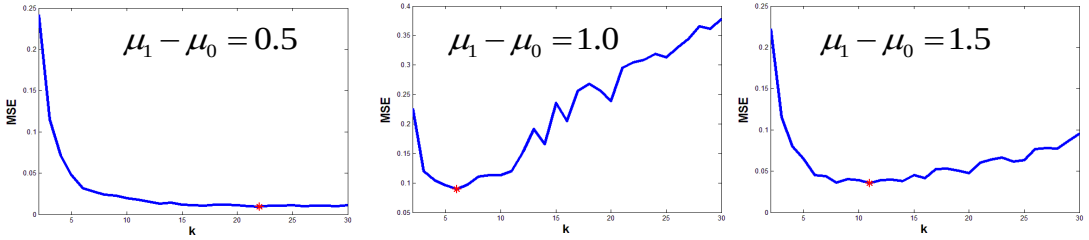
$n = 100000$

k_{MSE}		p_B = 0.02	0.05	0.1	0.2
$\mu_1 - \mu_0$	0.5	118	198	298	371
	1	50	61	106	141
	1.5	17	28	32	48
$k = \lceil \sqrt{n_0} \rceil$		5	8	10	15

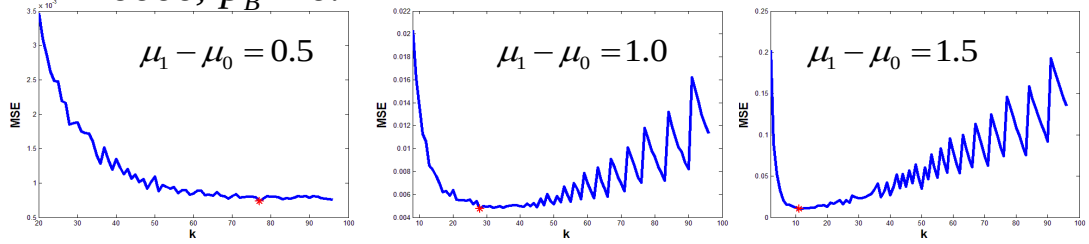
Simulation results

- **Dependence of MSE on *k*.**

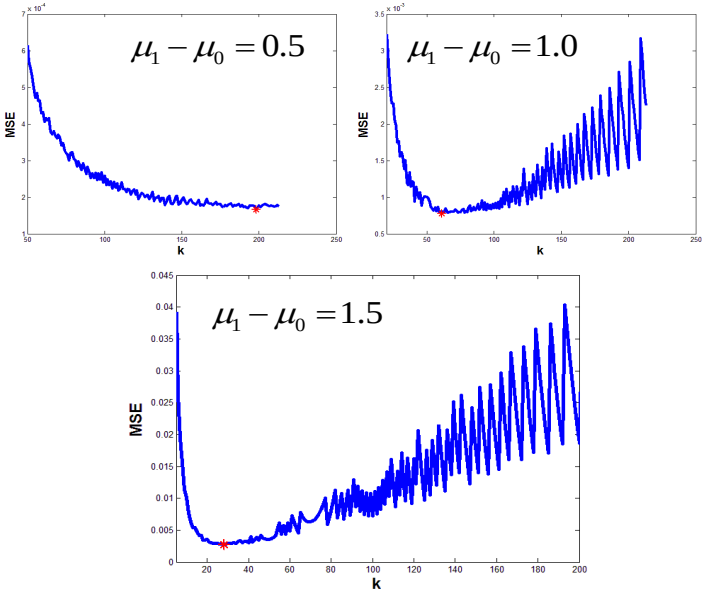
$n = 1000,\ p_B = 0.2$



$n = 10000,\ p_B = 0.2$



$n = 100000,\ p_B = 0.05$



$$k = \left\lceil \frac{\sqrt{p_B \cdot n}}{|\hat{\mu}_1 - \hat{\mu}_0|^{\sqrt{2}}} \right\rceil$$

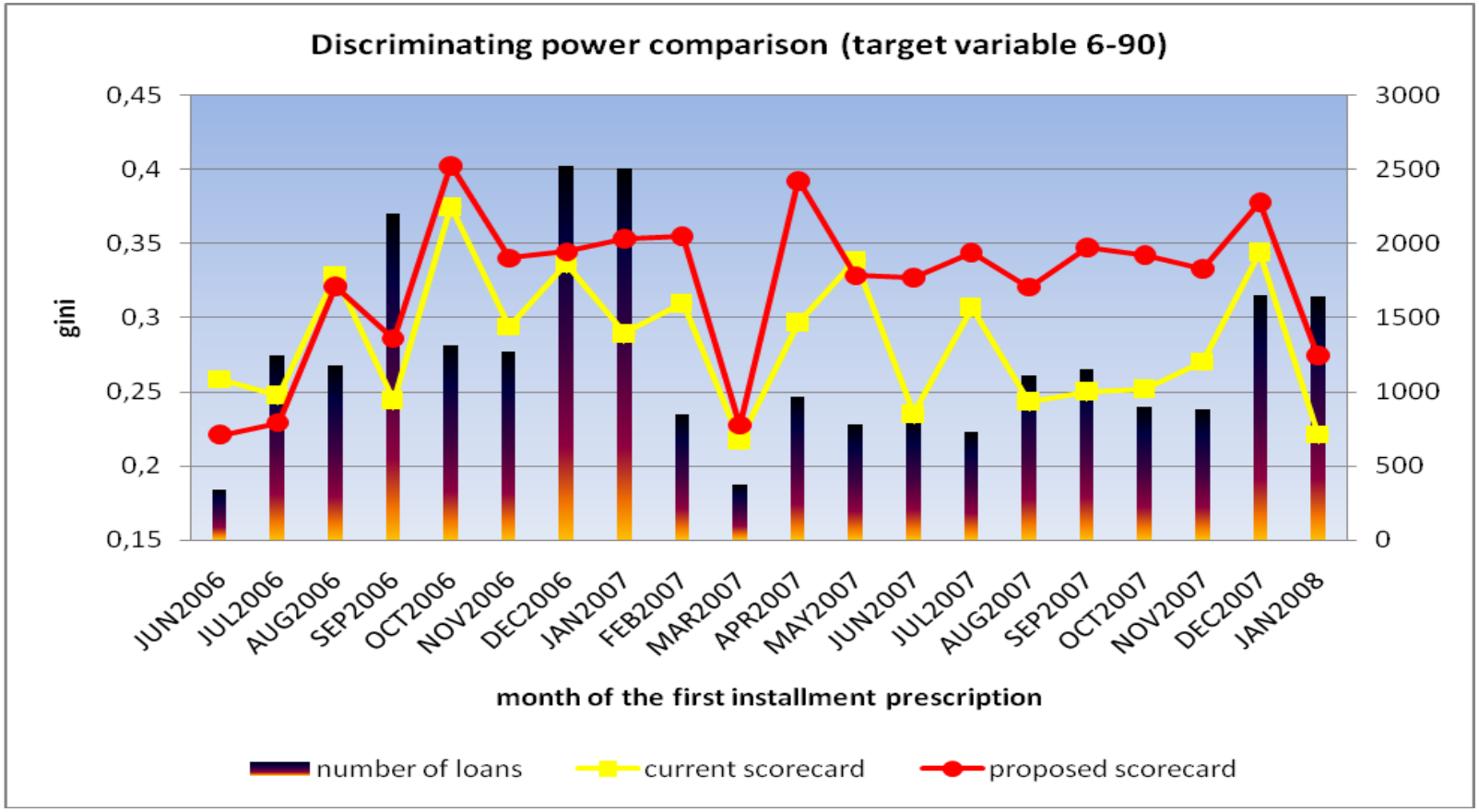
$$\hat{I}_{val,AESIS\,2}$$

$n = 10000$

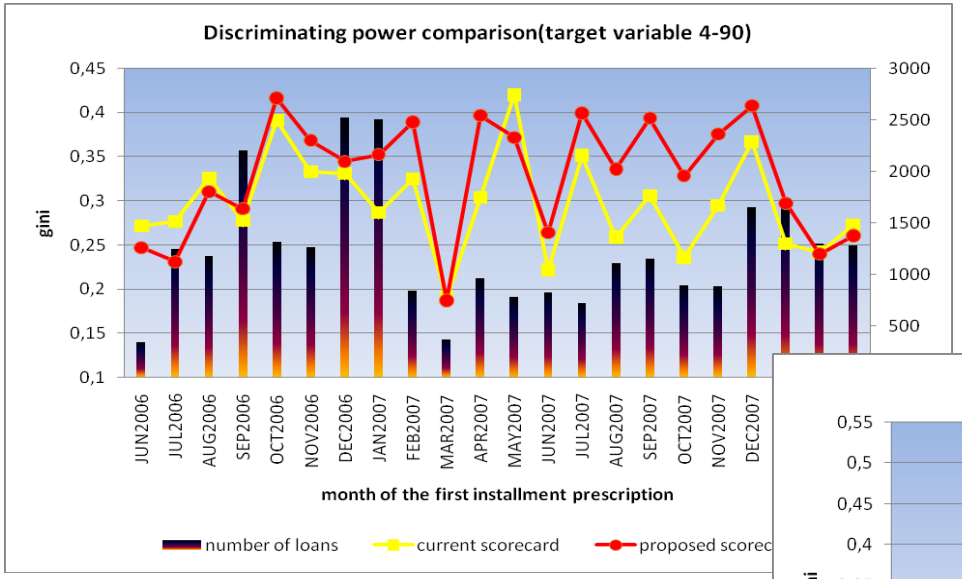
$k = \left\lceil \frac{\sqrt{p_B \cdot n}}{\lvert\hat{\mu}_1 - \hat{\mu}_0\rvert^{\sqrt{2}}} \right\rceil$		p_B			
		0.02	0.05	0.1	0.2
$\mu_1 - \mu_0$	0.5	38	60	85	120
	1	15	23	32	45
	1.5	8	13	18	26

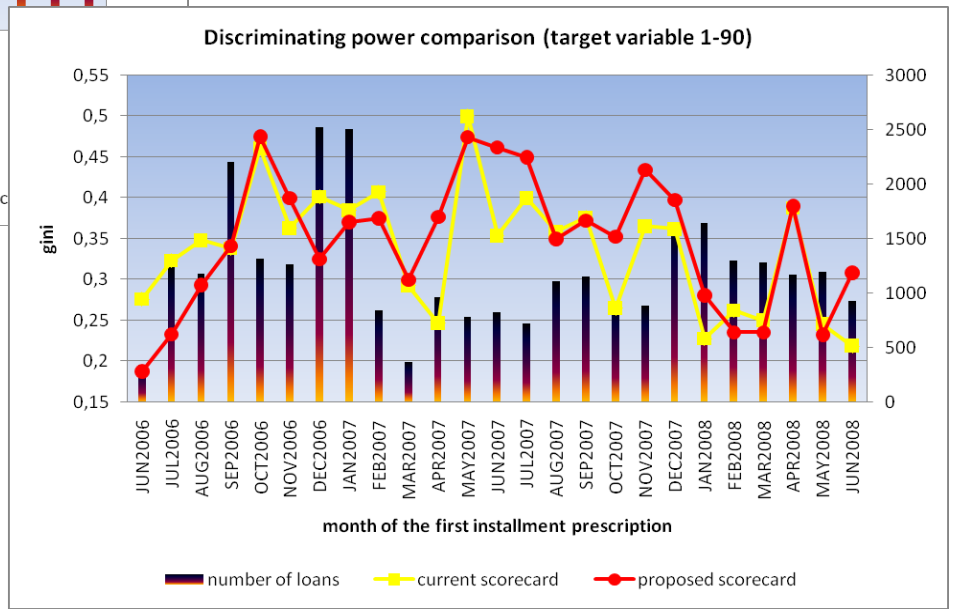
k_{MSE}		p_B			
		0.02	0.05	0.1	0.2
$\mu_1 - \mu_0$	0.5	29	51	77	112
	1	15	24	28	45
	1.5	6	11	11	14

Scorecard Strength

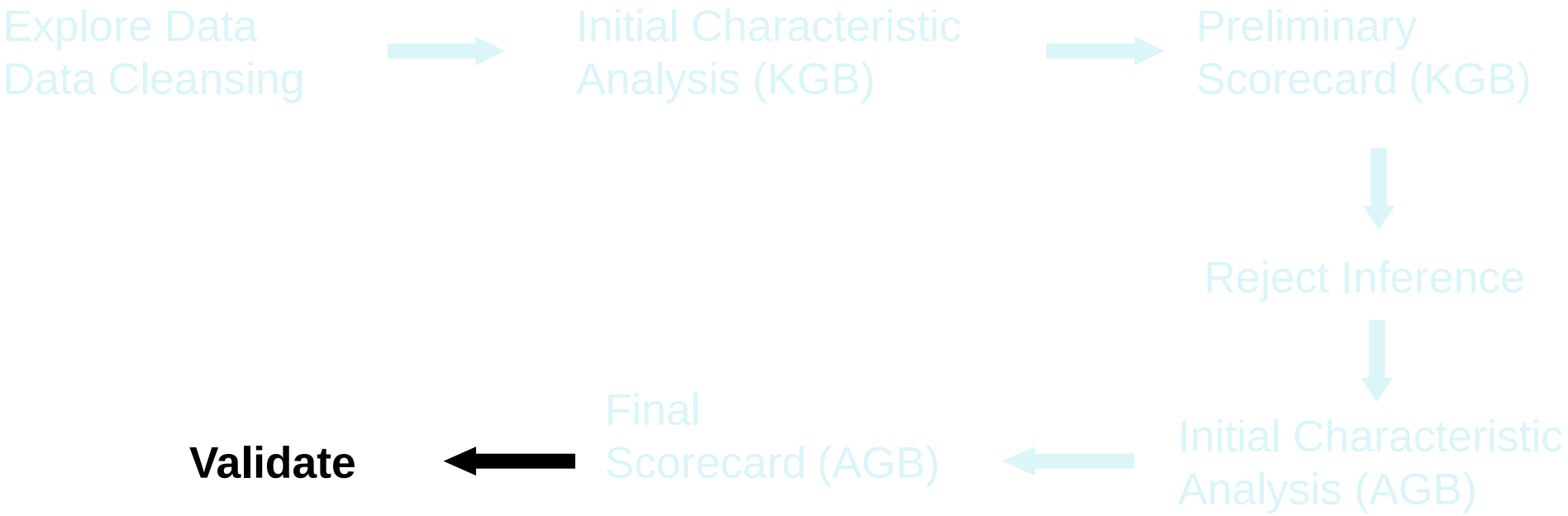


Scorecard Strength





Process Flow



Validation

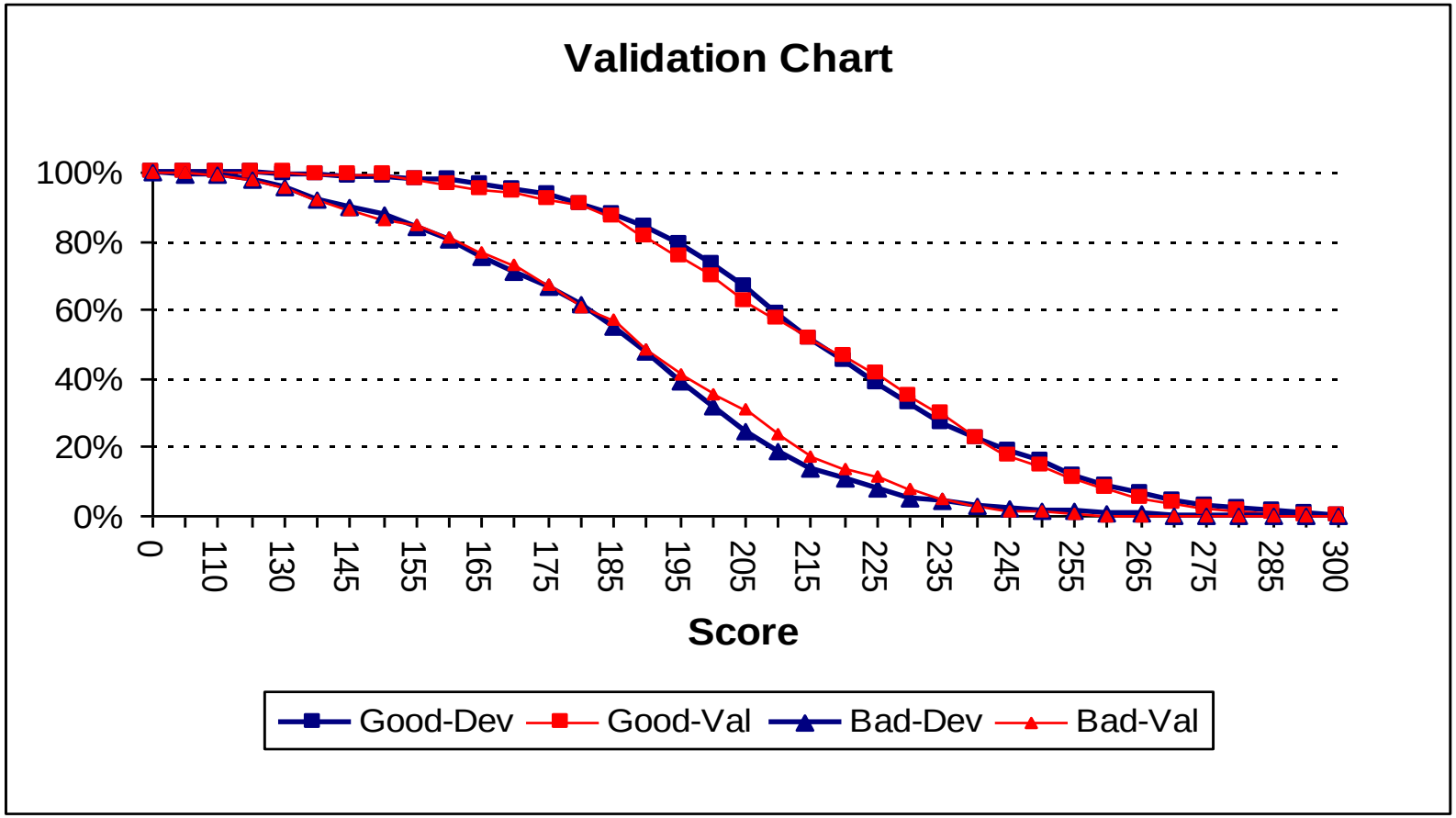
- Why?
  - Confirm that the model is robust and applicable on the subject population
  - Holdout sample
    - 70/30, 80/20 or random samples of 50–80%
- 2 Methods
  - Compare statistics for development versus validation
  - Compare distributions of goods and bads for development versus validation.

Validation – Comparing Statistics

Fit Statistic	Label	Training	Validation	Test
AIC	Akaike's Information Criterion	6214.0279153	.	.
ASE	Average Squared Error	0.0301553132	0.0309774947	.
AVERR	Average Error Function	0.1312675287	0.1355474611	.
DFE	Degrees of Freedom for Error	23609	.	.
DFM	Model Degrees of Freedom	7	.	.
DFT	Total Degrees of Freedom	23616	.	.
DIV	Divisor for ASE	47232	45768	.
ERR	Error Function	6200.0279153	6203.7361993	.
FPE	Final Prediction Error	0.0301731951	.	.
MAX	Maximum Absolute Error	0.9962871546	0.9959395534	.
MSE	Mean Square Error	0.0301642541	0.0309774947	.
NOBS	Sum of Frequencies	23616	22884	.
_NW_	Number of Estimate Weights	7	.	.
RASE	Root Average Sum of Squares	0.1736528525	0.1760042464	.
RFPE	Root Final Prediction Error	0.1737043324	.	.
RMSE	Root Mean Squared Error	0.1736785944	0.1760042464	.
SBC	Schwarz's Bayesian Criterion	6270.5156734	.	.
SSE	Sum of Squared Errors	1424.295752	1417.777979	.
SUMW	Sum of Case Weights Times Freq	47232	45768	.
MISC	Misclassification Rate	0.0320121951	0.0325117986	.
PROF	Total Profit for GB	3430000	2730000	.
APROF	Average Profit for GB	145.24051491	119.29732564	.

If stats are similar, then scorecard is validated.

Validation – Compare Distributions



Valid if no significant difference.

Validation

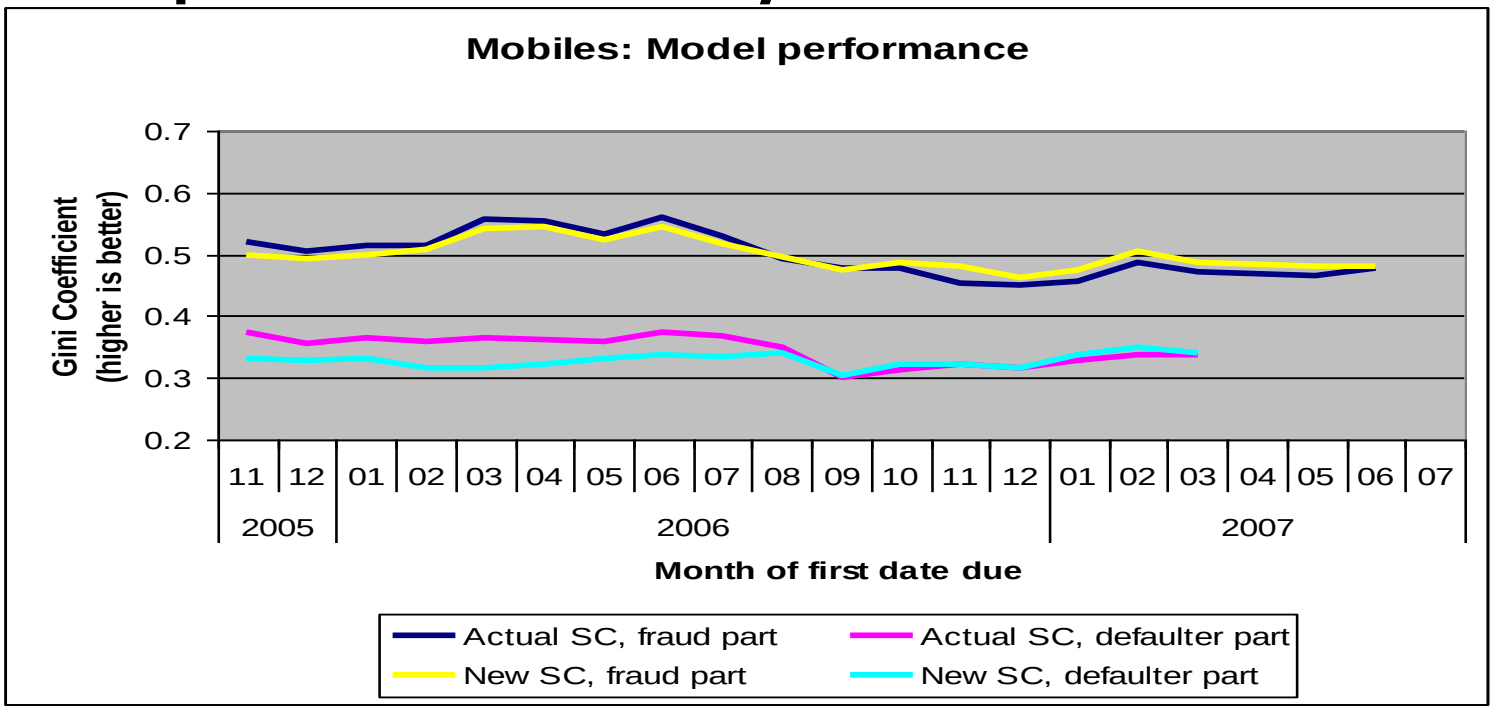
- Common reasons for not validating
  - Characteristics with large score ranges,
  - Concentration of a certain type of attribute in one sample (for example, not random sampling),
  - small sample sizes

Validation

Comparison with the old scorecard

Month by month comparison of performance of the old and the new scorecard, both for development and hold-out sample – on given segment

Comparison of performance month by month



Validation

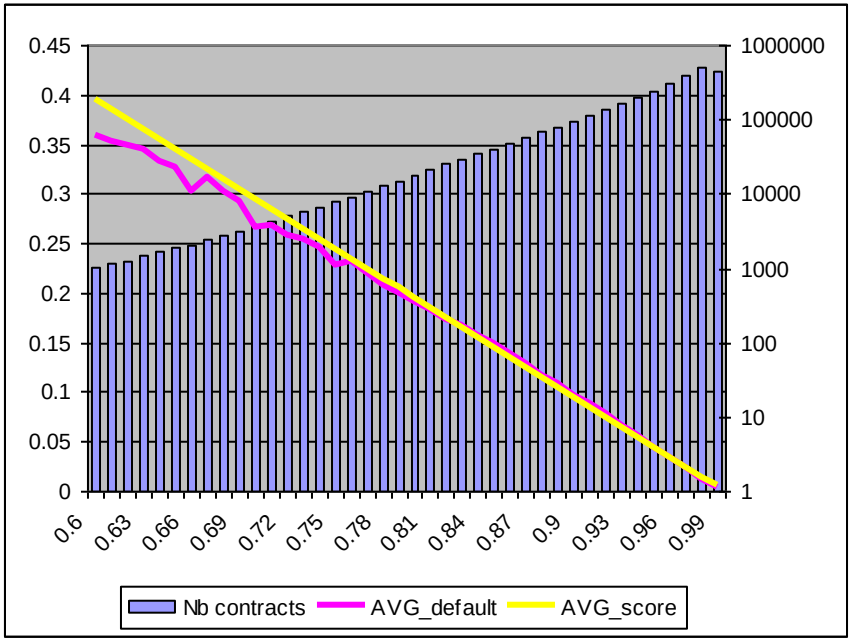
Power on fresh data

Use fresh data and compute "softer" good bad definitions (e.g. 1_30, 1_60 instead of 1_90). Measure power of the scorecard on development sample according these definitions and compare it with performance on the fresh data.

Comparison with real default

Month by month comparison of average predicted pd by the new scorecard and real default, for both development and hold-out samples. Diagonal test - score on x-axis and real default on y-axis – graph of average default should be ideally monotonous (the higher the score, the lower the default)

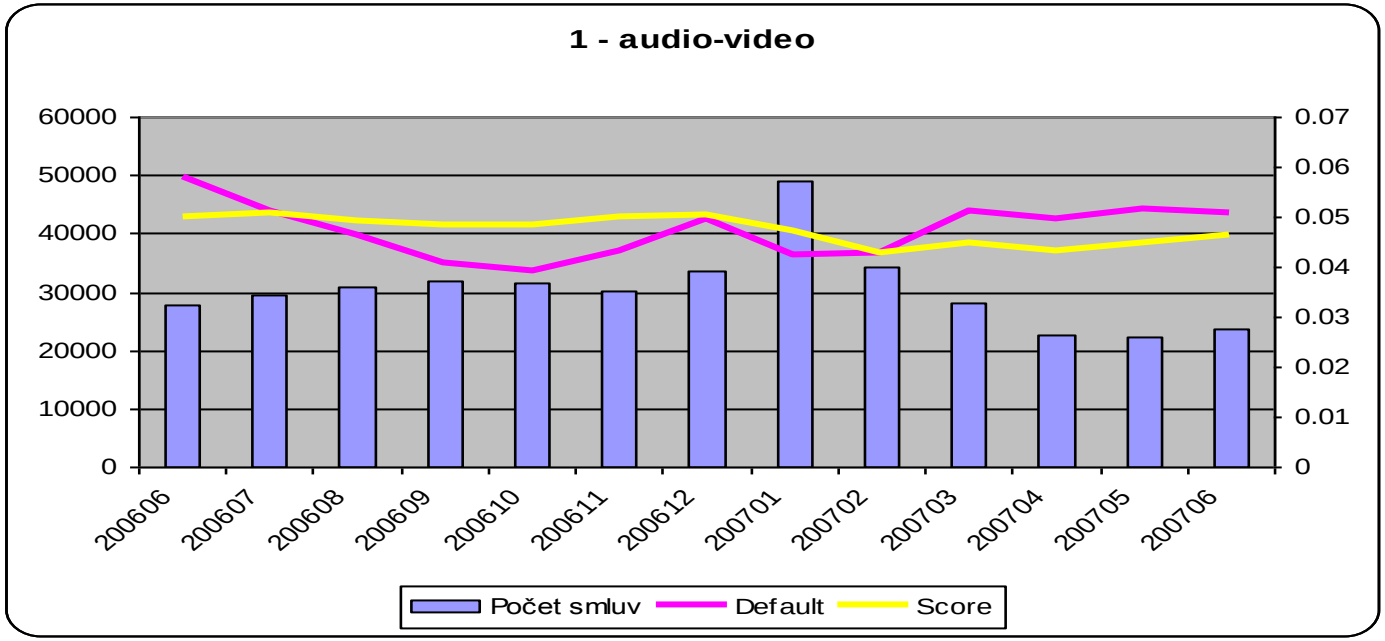
Graph of diagonal test



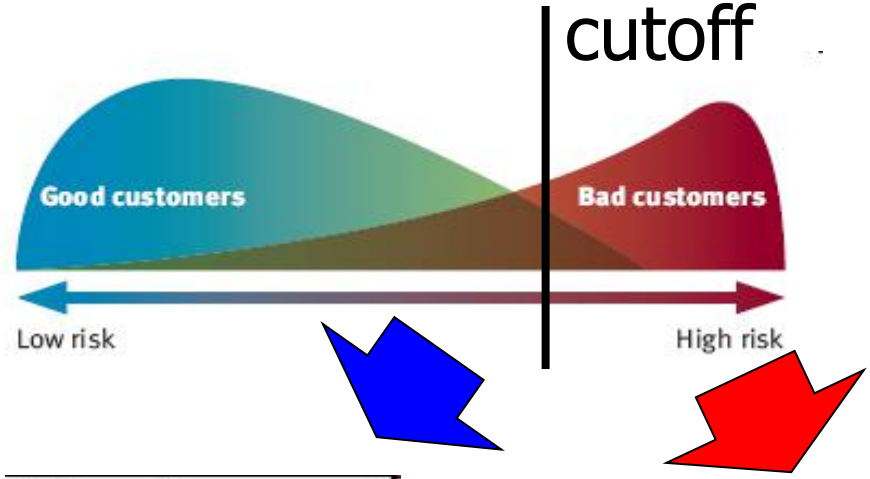
Validation

Comparison with real default

Graph of predicted pd versus the real default



12. Cutoff, RAROA, Monitoring



Možné zamítací škály – cutoff

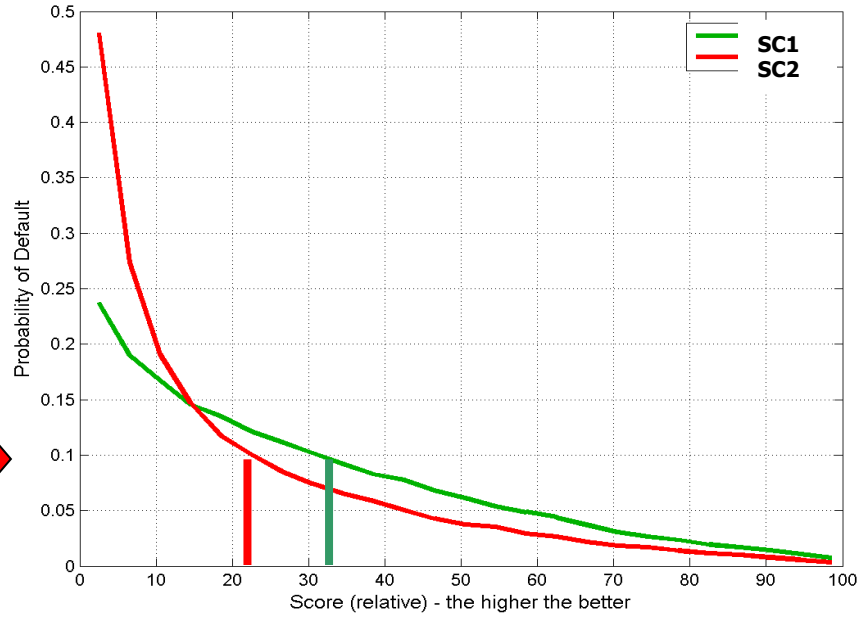
- cutoff hodnota určuje mez, při které je žádost o úvěr schválena/zamítnuta

- Je možné použít tyto zamítací škály:
  - **PD – Pravděpodobnost Defaultu (Probability of Default)**
  - **KRN - Kreditní Rizikové Náklady (CRE – Credit Risk Expenses)**
  - **Marže (Margin)**
  - **RAROA**
  - **…**

Cutoff na škále PD

cutoff = 0.1 (tj. zamítám všechny s pravděpodobností defaultu větší než 10 %)

Cut off



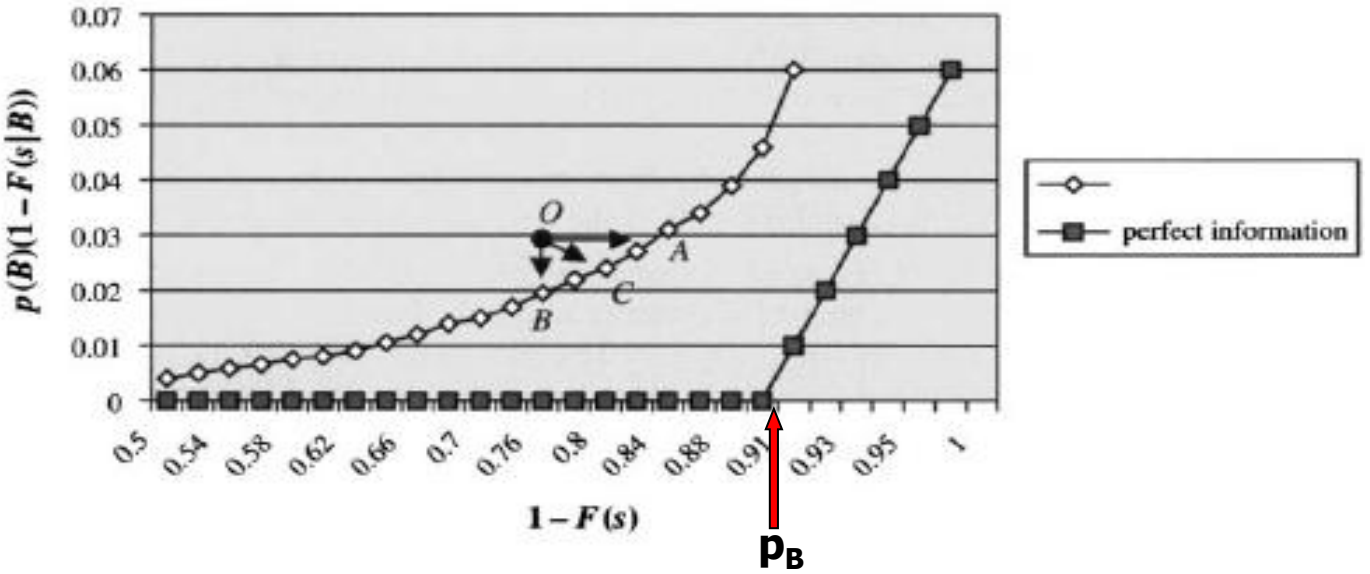
- Pro SC1 je reject rate 22 %.
- Pro SC2 je reject rate 33 %.

Strategická křivka (Strategy curve)

Bad acceptance rate $= p_B\left(1 - F\left(s|B\right)\right)$

Acceptance rate $= 1 - F(s)$

Actual bad rate $= \dfrac{p_B\left(1 - F\left(s|B\right)\right)}{1 - F(s)}$



Při zavádění nové scoringové funkce typicky dochází k tomu, že stávající nastavení schvalovacího procesu (nastavení cutoff) je reprezentováno bodem *O* , který leží nad novou strategickou křivkou. Otázkou pak je směr, kterým se chceme vydat při stanovení nového cutoff. Pokud se posuneme do bodu *A*, potom zachováme poměr schválených špatných klientů, ale současně zvýšíme celkový poměr schválených klientů. Při posunu do bodu *B* schválíme stejný poměr klientů, ale snížíme poměr schválených špatných klientů a tedy i poměr špatných klientů (bad rate). Posunem do bodu *C* zachováme bad rate při současném zvýšení poměru schválených klientů.

Nastavení cutoff maximalizující zisk (profit)

Profit - náhodná veličina definovaná jako:

$$R = \begin{cases} 0, & \text{je – li úvěr zamítnut} \\ L, & \text{je – li úvěr schválen a stane se dobrým} \\ -D, & \text{je – li úvěr schválen a stane se špatným} \end{cases}$$

Označme p_G a p_B proporci dobrých a špatných klientů v populaci. $q(G|s)$ ($q(B|s)$) označuje podmíněnou pravděpodobnost, že klient mající skóre s bude dobrý (špatný), přičemž $q(G|s) + q(B|s) = 1$. Nechť $p(s)$ je proporce populace se skóre s.

Střední hodnota profitu při schválení klientů se skóre s:

$$E\{R|s\} = Lq(G|s) - D(1 - q(G|s)) = (L + D)q(G|s) - D$$

Tedy k maximalizaci profitu je třeba schválit ty klienty, jejichž skóre splňuje podmínku:

$$q(G|s) \geq \frac{D}{D+L}$$

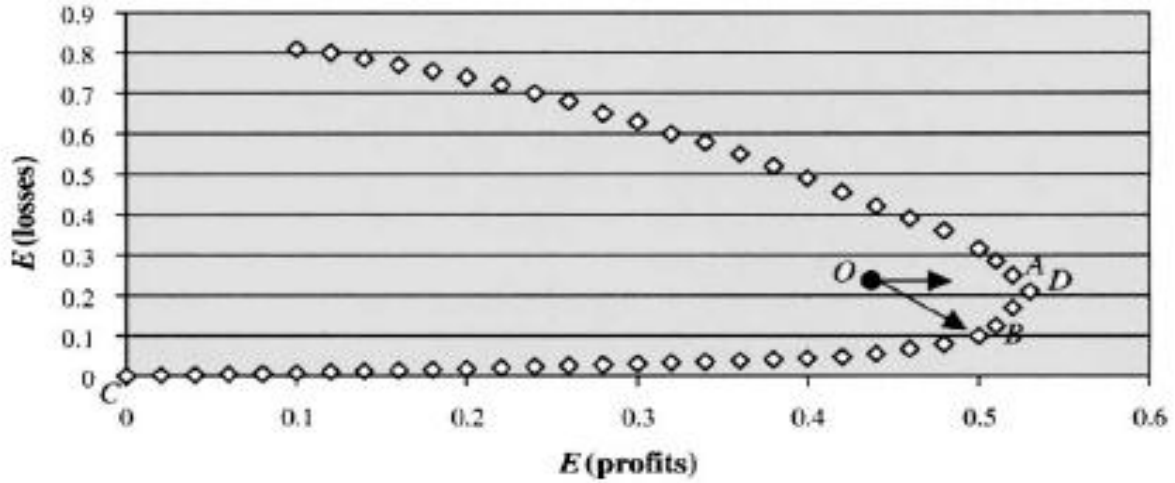
Nastavení cutoff maximalizující profit

Nechť A označuje množinu skóre, kde je splněna předchozí podmínka. Pak je střední hodnota zisku (profitu) na jednoho klienta dána vztahem:

$$E^*\{R\} = \sum_{s \in A} ((L + D)q(G|s) - D)p(s).$$

Pokud L a D navíc závisí na skóre s, je situace ještě o něco složitější. Více viz Thomas et al. (2002).

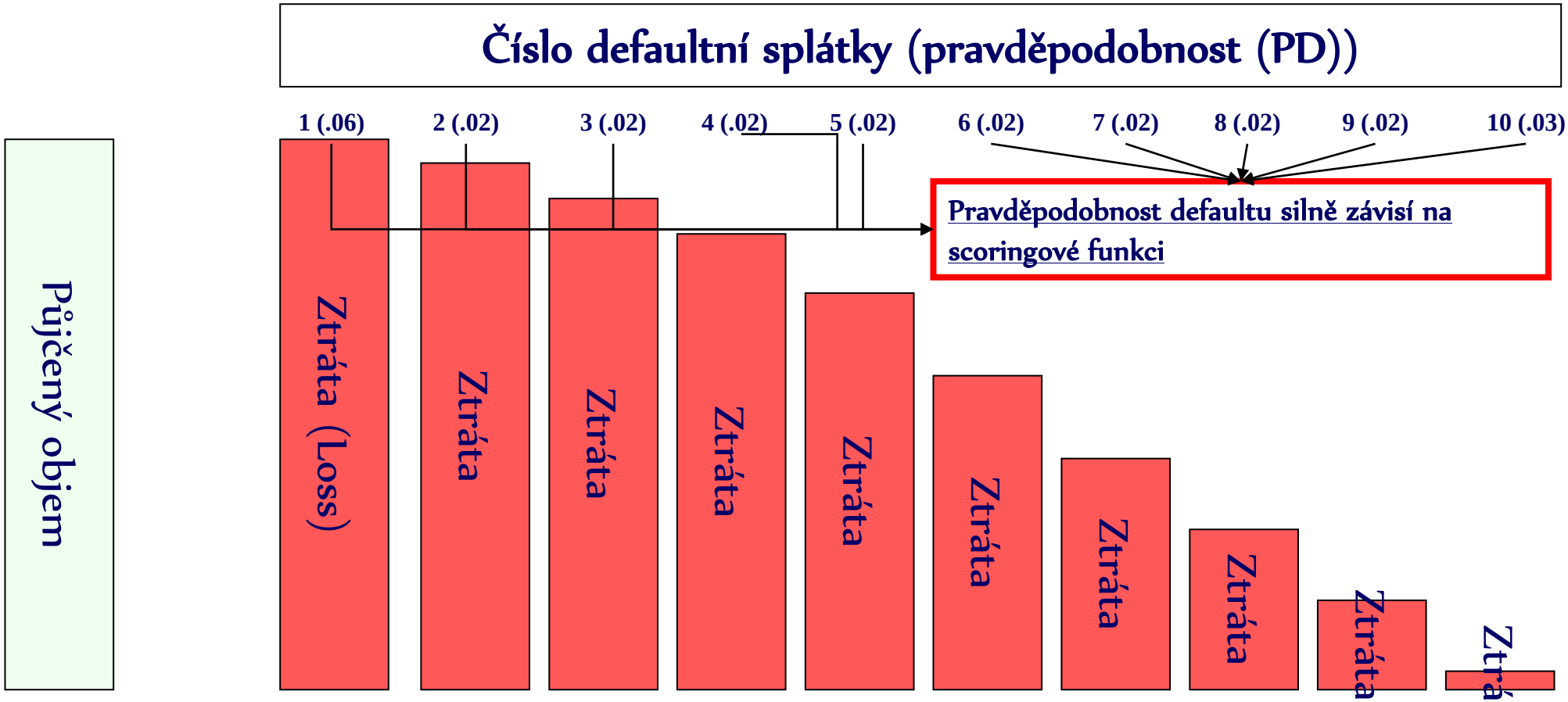
Nastavení cutoff maximalizující profit



Body na spodní části křivky odpovídají vyšším cutoff hodnotám, a tedy i menšímu počtu přijatých špatných klientů, zatímco body na horní části křivky odpovídají menším hodnotám cutoff, tj. vyššímu počtu přijatých špatných klientů. Efektivní hranicí je tedy spodní část křivky od bodu C do bodu D.

Jestliže aktuální nastavení schvalovacího procesu odpovídá bodu *O*, opět máme možnost posunu na křivku odpovídající nové scoringové funkci. První možností je zachování poměru schválených špatných klientů, tj. posun do bodu *A*. Druhou možností je zachování celkového poměru schválených klientů, tj. posun do bodu *B*. Je zřejmé, že posun do bodu *A* není vhodná volba, protože tento bod neleží na efektivní hranici a lze snadno dosáhnout stejného očekávaného zisku při nižší očekávané ztrátě.

Definice KRN (CRE)



CRE = ((1-Recovery) * SUM(PD * Loss))/(Expected Average Volume)

Profit = (Interest rate – CRE)*Expected Average Volume

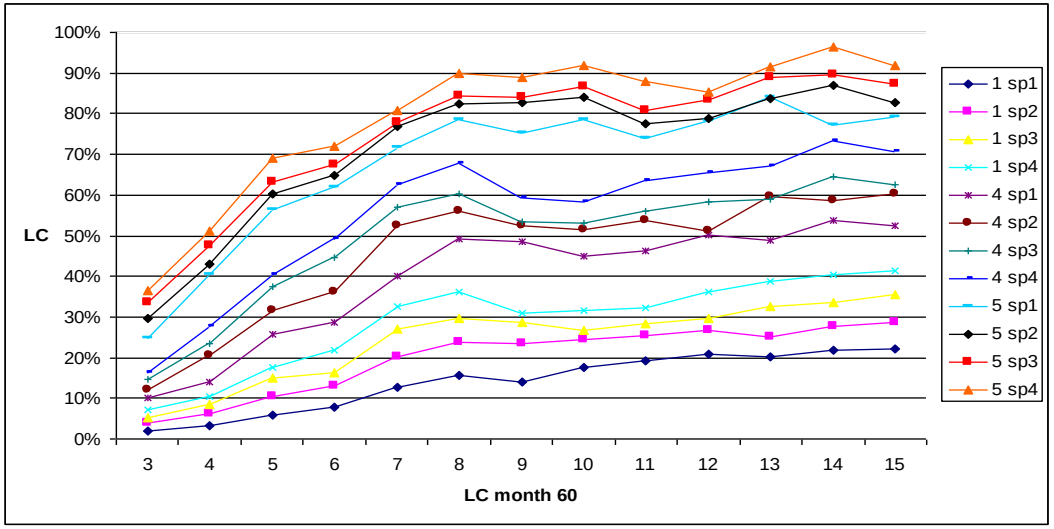
Úroková míra

Očekávaný průměrný objem úvěru

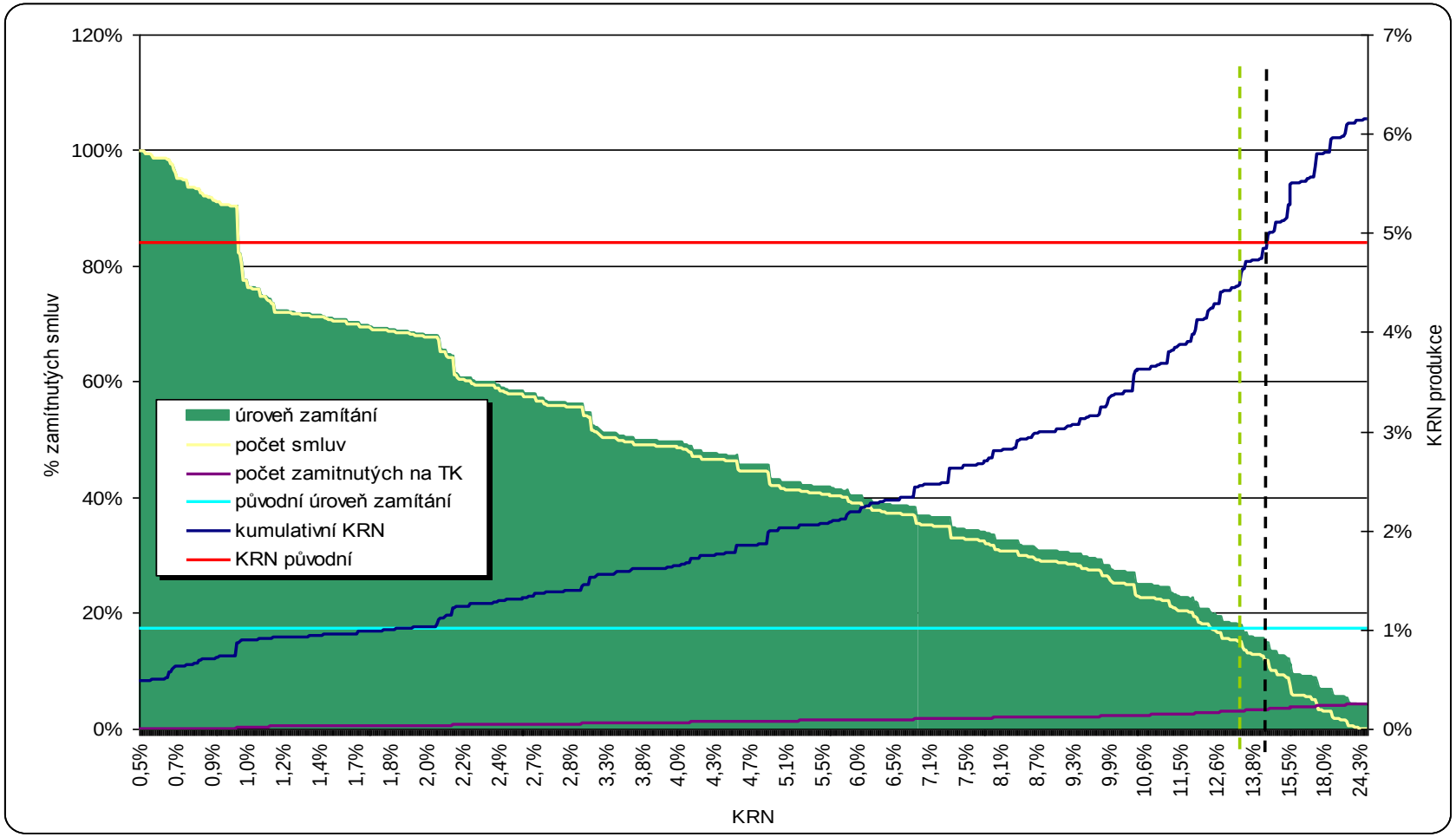
Recovery (=Late collection(LC))

Číslo defaultní splátky	score			
	band1	band2	band3	band4
1.	**20%**	**25%**	**30%**	**35%**
2.-4.	**50%**	**55%**	**60%**	**65%**
5. +	**75%**	**80%**	**85%**	**90%**

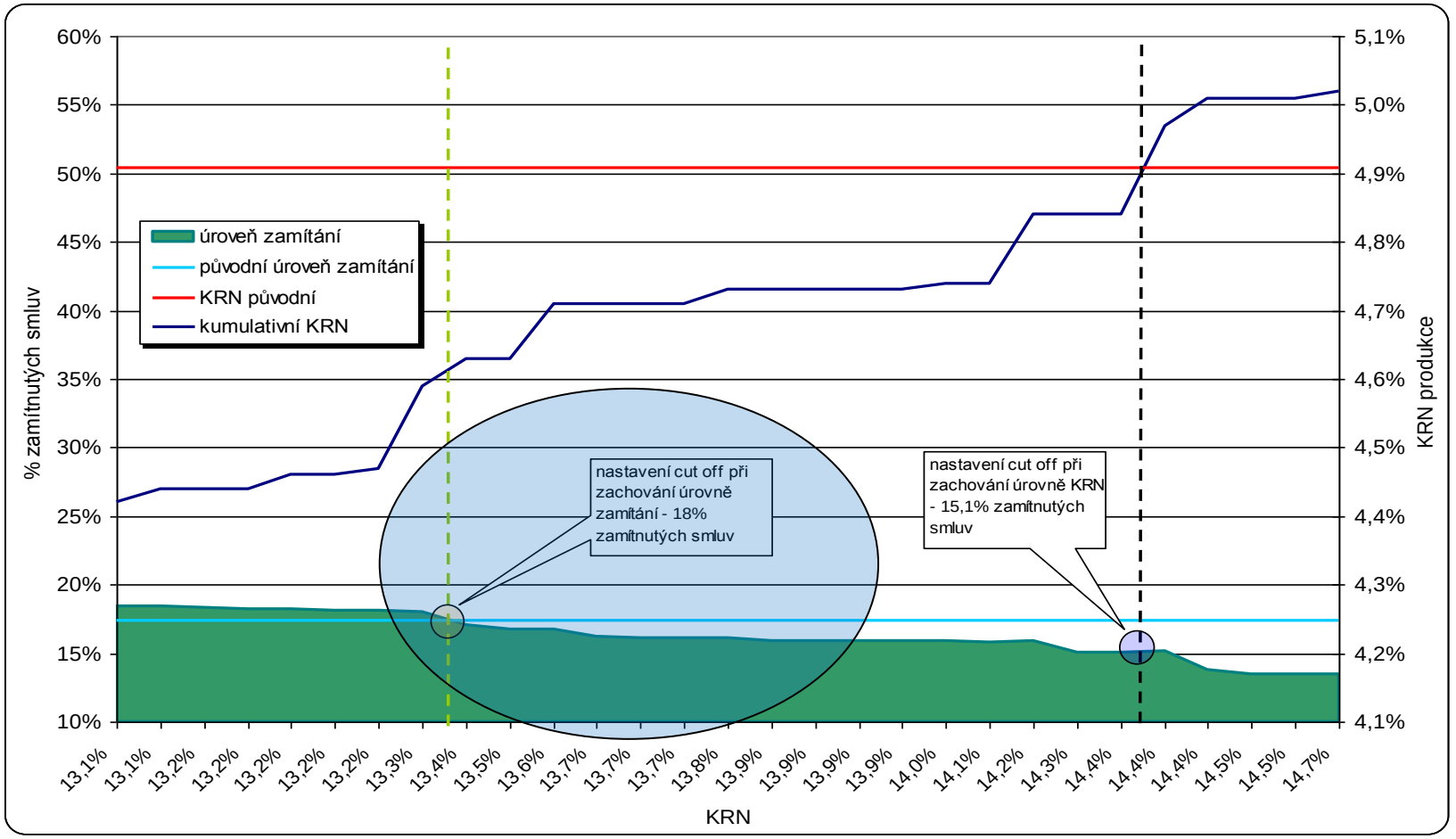
odhad



Cutoff na škále KRN



Cutoff na škále KRN



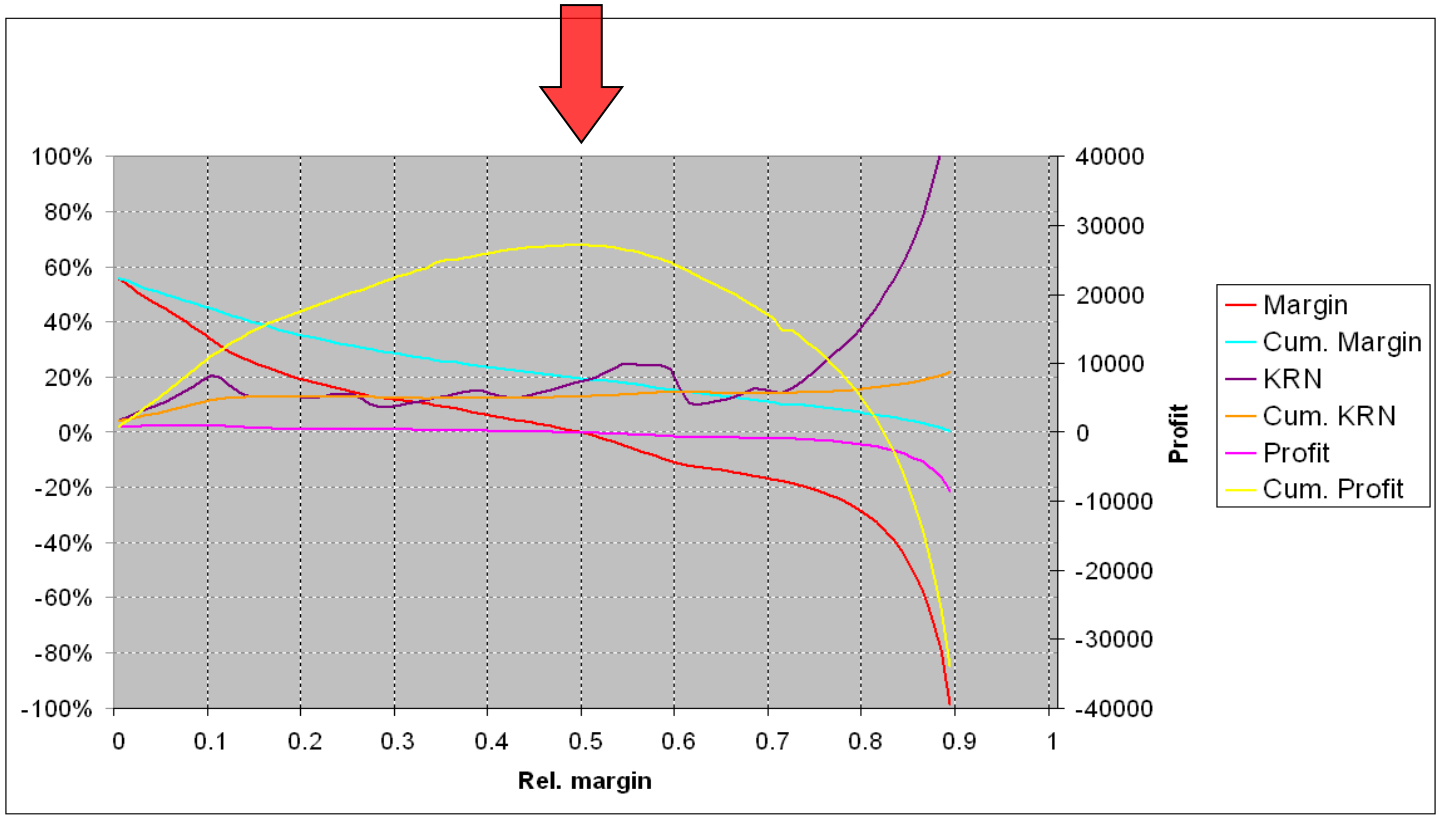
(Očekávaná) Marže

(Očekávaná) Marže = Úroková míra (vč. poplatků) – KRN – OPEX

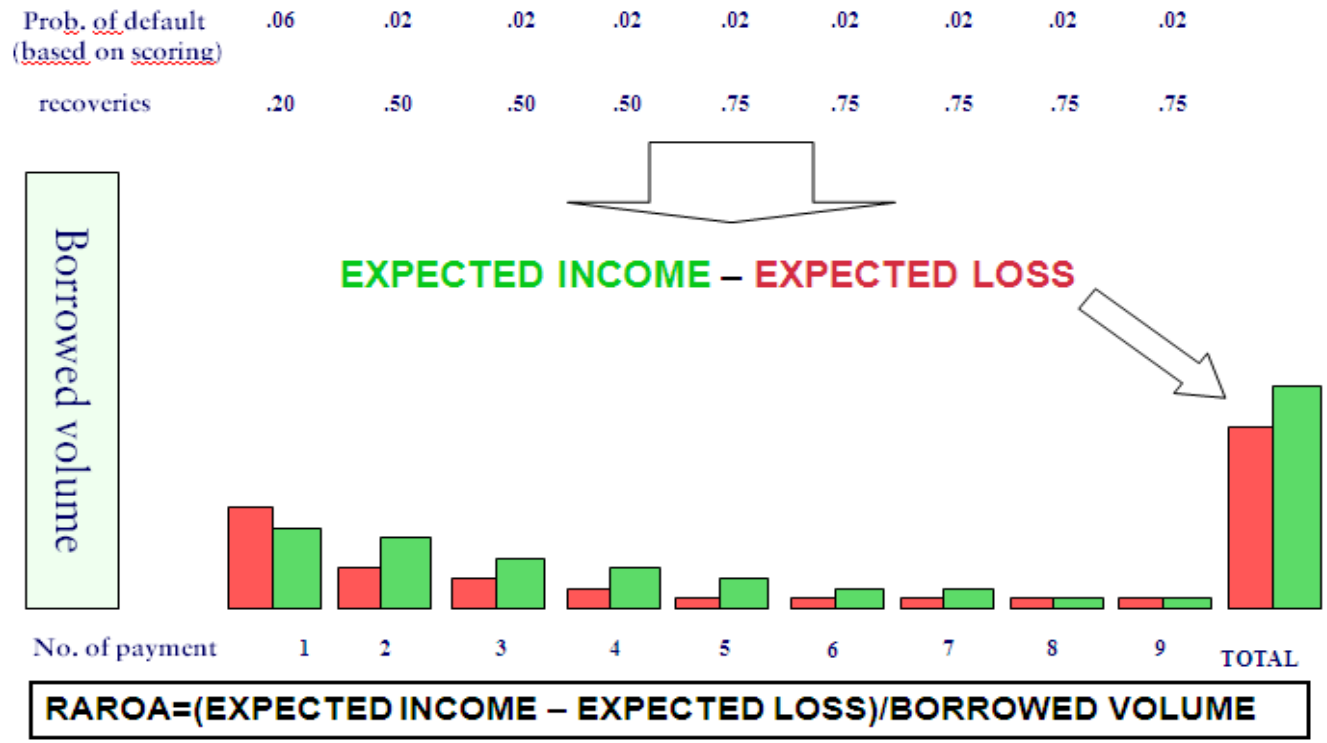
- ***Úroková míra***
  - *Efektivní míra ideálního finančního toku (-výše úvěru-poplatky; anuita; anuita; … ; anuita).*
- ***KRN***
  - *Viz výše.*
- ***OPEX***
  - *Cena peněz.*
  - *Režijní náklady, variabilní náklady, podpora prodejní sítě.*
  - *Náklady na administrátory – vlastní zaměstnance zajišťující zpracování úvěru.*

Marže (Margin)

- Optimální cutoff: *marže=0*



RAROA
(Risk Adjusted Return On Assets)



RAROA

- $t \ldots$ pořadí splátky úvěru, 0 je okamžik poskytnutí úvěru
- $T \ldots$ počet splátek
- $x(t) \ldots$ nesplacená část úvěru podle splátkového plánu v čase $t, \ldots t = 0, \ldots, T$. $x(0)$ je výše úvěru, $x(T) = 0$.
- $u(t) \ldots$ úroková část anuity t, $t = 1, \ldots, T$.
- $j(t) \ldots$ část anuity odpovídající splátce jistiny t, $t = 1, \ldots, T$.
- $k(t) \ldots$ komise od klienta v čase t, $t = 0, \ldots, T$.
- $A \ldots$ výše anuity (absolutně). $A = u(t) + j(t), t = 1, \ldots, T$.
- $p(t) \ldots$ pravděpodobnost 90 denního defaultu úvěru na splátce $t, t = 1, \ldots, T$
- $\mathrm{E}\, Z \ldots$ očekávaná ztráta z úvěru
- $\mathrm{E}\, P \ldots$ očekávaný úrokový příjem z úvěru
- $RC \ldots$ absolutní výše z dlužné částky klienta 90 dní po splatnosti, která je klientem splacena v budoucnu, přepočtena přes NPV k okamžiku devadesátidenního defaultu klienta
- $r(t, f) \ldots$ procento výtěžnosti z dlužné částky klienta, který je poprvé 90 dní po splatnosti na splátce t a klient má hodnotu podvodnického skóre (nesplacení první splátky) f. Procento zohledňuje NPV všech budoucích splátek klienta po okamžiku defaultu.

RAROA

- GM ... hrubý očekávaný zisk z klienta
- s je sazba úvěru p.a.
- i... cena zdrojů vyjádřená v procentu p.a.
- c... komise z obchodu poskytnutá obchodnímu partnerovi vyjádřená jako procento z jistiny
- NM_I ... čistý očekávaný zisk typu I z klienta po odečtení ceny zdrojů
- NM_{II} ... čistý očekávaný zisk z klienta typu II po odečtení ceny zdrojů a komisí z obchodu.¨¨
- ROA... ukazatel Return on Asset počítaného z hrubého zisku
- ROA_I ... ukazatel Return on Asset typu I počítaný z čistého zisku typu I
- ROA_{II} ... ukazatel Return on Asset typu II počítaný z čistého zisku typu II
- KRN je úroková míra p.a. vyjadřující rizikovost úvěru.

RAROA

$$\mathrm{E}\,Z = \sum_{t=1}^{T} p(t) \cdot x(t-1).$$

$$\mathrm{E}\,P = k(0) + \sum_{t=1}^{T} \left(1 - \sum_{s=1}^{t} p(s)\right) (u(t) + k(t)).$$

$$GM = \mathrm{E}\,P - \mathrm{E}\,Z + RC.$$

$$RC = \sum_{t=1}^{T} p(t) \cdot r(t,f) \cdot x(t-1)$$

$$NM_I = GM - \sum_{t=1}^{T} \left(1 - \sum_{s=1}^{t} p(s)\right) \frac{i}{12} \cdot x(t-1).$$

$$NM_{II} = NM_I - c \cdot x(0).$$

$$ROA = \frac{GM}{x(0)}.$$

$$ROA_I = \frac{NM_I}{x(0)}.$$

$$ROA_{II} = \frac{NM_{II}}{x(0)}.$$

$$\frac{\mathrm{KRN}}{12} \cdot \sum_{t=1}^{T} \left(1 - \sum_{s=1}^{t} p(s)\right) x(t-1) = EZ - RC.$$

$$\sum_{t=1}^{T} \left(1 - \sum_{s=1}^{t} p(s)\right) x(t-1) = \frac{\sum_{t=1}^{T} \left(1 - \sum_{s=1}^{t} p(s)\right) u(t)}{s/12},$$

$$\mathrm{KRN} = \frac{EZ - RC}{EP} \cdot s.$$

Výhody RAROA

	Case A		Case B	
	Ideal flow	Expected flow	Ideal flow	Expected flow
	-1000	-1000	-1000	-1000
1	400	200	150	110
2	400	180	150	100
3	400	170	150	90
4	400	160	150	80
5			150	70
6			150	60
7			150	50
8			150	40
9			150	30
10			150	16
11			150	10
12			150	0

- A – krátkodobý úvěr s vysokým rizikem fraudu
- B – dlouhodobý úvěr s vysokým rizikem defaultu

Úroková míra (A) = 22%
Úroková míra (B) = 10%

KRN(A) = 44%
KRN(B) = 20%
cutoff na škále KRN preferuje B

Marže (A) = -22%
Marže (B) = -10%
cutoff na škále marže preferuje B

RAROA (A) = -0.29
RAROA (B) = -0.36
cutoff na škále RAROA preferuje A

Úvěr A je lepší, protože z něj plyne vyšší zisk (710>656), navíc je ho dosaženo mnohem dříve.

Cutoff segmentace

- ***Možná segmentace podle:***
  - ***Prodejní síť (skupina obchodních míst)***
  - ***Profitabilita produktu***
  - ***Kvalita prodejního místa***
  - ***Typ zboží (pro spotřebitelské úvěry)***
  - ***Výše úvěru***
  - ***…***

Cutoff scénáře

all

scenario	All credits		Approved credits		
	Reject rate		Avg. margin		Avg. KRN
	Credits	Volume	Credits	Volume	
0reject	0.0%	0.00%	-7.83%	-21.34%	30.70%
tk	22.4%	24.74%	-3.97%	-15.72%	26.33%
current	36.8%	46.79%	2.32%	-4.58%	19.11%
all30	29.5%	37.07%	3.18%	-3.36%	19.78%
total30	30.3%	38.55%	4.11%	-1.29%	19.26%
total32	31.5%	39.99%	4.63%	-0.77%	18.96%
total35	35.8%	46.01%	7.32%	3.22%	16.07%
all40	38.9%	48.97%	8.17%	3.41%	15.19%
tk_ekonom	59.3%	70.03%	19.39%	17.14%	13.47%
ekonom	50.9%	63.64%	19.24%	17.03%	14.23%

new

scenario	All credits		Approved credits		
	Reject rate		Avg. margin		Avg. KRN
	Credits	Volume	Credits	Volume	
0reject	0.0%	0.00%	-4.19%	-16.40%	26.19%
tk	9.0%	10.25%	-2.39%	-13.71%	24.50%
current	24.5%	34.89%	3.26%	-3.42%	17.99%
all30	16.5%	23.87%	4.07%	-2.34%	18.60%
total30	17.3%	25.42%	4.85%	-0.59%	18.19%
total32	18.6%	27.16%	5.36%	-0.03%	17.87%
total35	23.2%	33.80%	7.74%	3.52%	15.34%
all40	26.7%	37.35%	8.61%	3.88%	14.45%
tk_ekonom	50.7%	62.90%	19.53%	17.26%	13.05%
ekonom	47.5%	60.46%	19.48%	17.23%	13.26%

Cutoff impact evaluation

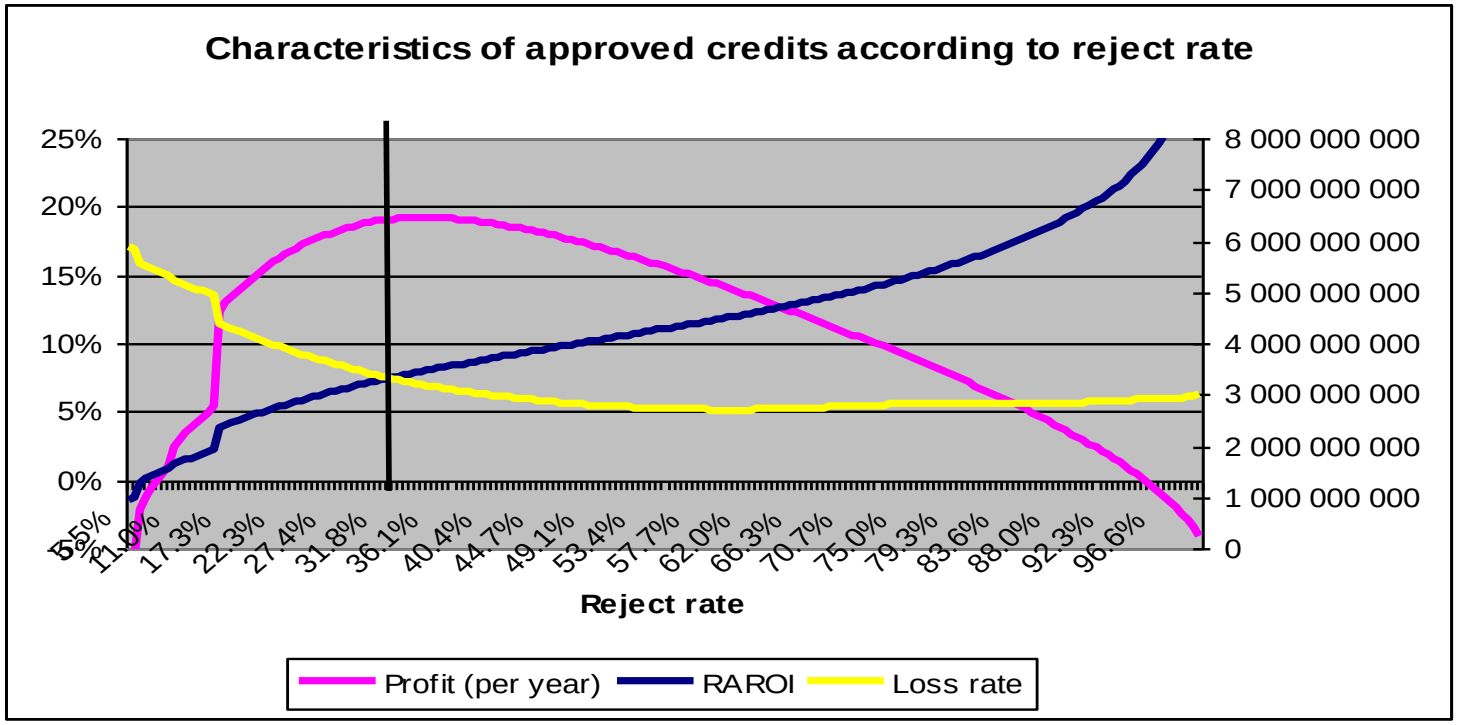
Evaluation of Reject rate, Profitability, Default and Loss rates before and after cutoff change according to Distribution channel or Segment of scorecard.

Cutoff impact evaluation table

	Before Christmas (approved credits)				After Christmas (approved credits)			
	Reject rate	RAROA	Loss rate	Profit (per year)	Reject rate	RAROA	Loss rate	Profit (per year)
Segment 1	24.7%	3.65%	11.33%	414 363 110	24.3%	3.75%	11.19%	428 757 430
Segment 2	12.1%	4.01%	8.22%	160 364 072	12.9%	3.95%	8.29%	159 917 943
Segment 3	45.1%	9.64%	9.69%	747 636 468	45.1%	9.8%	9.5%	758 966 512
Segment 4	22.2%	5.80%	4.89%	52 213 720	20.1%	5.62%	5.05%	51 715 263
Segment 5	20.9%	6.77%	5.41%	54 312 614	19.7%	6.61%	5.48%	53 975 903
Segment 6	33.4%	7.04%	7.22%	212 090 365	32.6%	7.04%	7.16%	211 684 371
Segment 7	49.3%	9.30%	8.93%	36 840 287	49.2%	9.4%	8.8%	37 140 165
Segment 8	19.3%	4.68%	2.96%	15 668 962	14.9%	4.54%	3.16%	15 636 910
Segment 9	32.0%	8.41%	5.06%	3 679 430	27.2%	7.97%	5.26%	3 535 809
Segment 10	33.4%	7.14%	6.69%	1 823 050 341	33.4%	7.2%	6.6%	1 832 986 599
Segment 11	28.5%	6.34%	7.36%	2 633 609 071	28.6%	6.47%	7.24%	2 651 352 740
ALL	**32.6%**	**6.64%**	**8.37%**	**6 153 828 440**	**32.6%**	**6.96%**	**8.17%**	**6 205 669 645**

Cutoff sensitivity analysis

Profitability, Default and Loss rates according to reject rate into one graph

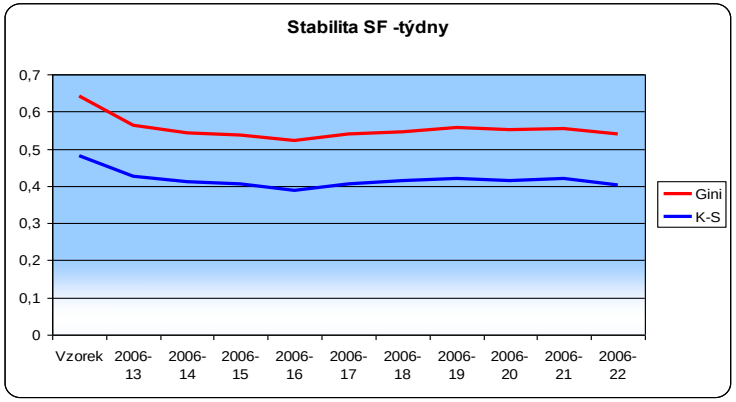


<u>Decision</u>

Reasoning, why the final cutoffs were chosen

Monitoring

	výv. vzorek [1]	týden1 [2]	[3]=[2] -[1]	[4]=[2]/[1]	[5]=ln[4]	[6]=[3]*[5]
skóre_1	10,00%	5,63%	-0,044	0,563	-0,574	0,025
skóre_2	10,00%	11,21%	0,012	1,121	0,114	0,001
skóre_3	10,00%	11,00%	0,010	1,100	0,095	0,001
skóre_4	10,00%	10,97%	0,010	1,097	0,092	0,001
skóre_5	10,00%	10,31%	0,003	1,031	0,031	0,000
skóre_6	10,00%	10,12%	0,001	1,012	0,012	0,000
skóre_7	10,01%	9,62%	-0,004	0,961	-0,039	0,000
skóre_8	10,00%	9,89%	-0,001	0,989	-0,011	0,000
skóre_9	10,00%	10,31%	0,003	1,031	0,030	0,000
skóre_10	10,00%	10,94%	0,009	1,095	0,091	0,001
					PSI	0,030



Monitoring scoringových modelů

- Není překvapivé, že prediktivní modely se ve statistickém slova smyslu chovají nejlépe na vývojovém vzorku dat. Výstupy těchto modelů, např. skóre nebo rating klienta, jsou počítány pomocí jistých vzorců, jejichž koeficienty příslušející nezávislým proměnným (prediktorům) jsou odvozeny na datech vývojového vzorku. Posun distribuce výstupu daného modelu je pak zapříčiněn právě změnou vstupních hodnot modelu, tj. prediktorů, v průběhu času. V podstatě ihned (alespoň většinou) po nasazení prediktivního modelu do praxe dochází k jistému poklesu jeho prediktivní síly, který je způsoben určitou změnou vstupních hodnot modelu. Zásadní je v praxi nastavení takových procesů, které odhalí, že se tak děje, proč se tak děje a jak vážný problém to ve svých důsledcích znamená.

Monitoring scoringových modelů

- Faktorů způsobujících posun v distribuci prediktorů, a následně posun v distribuci výstupu prediktivního modelu, je několik:

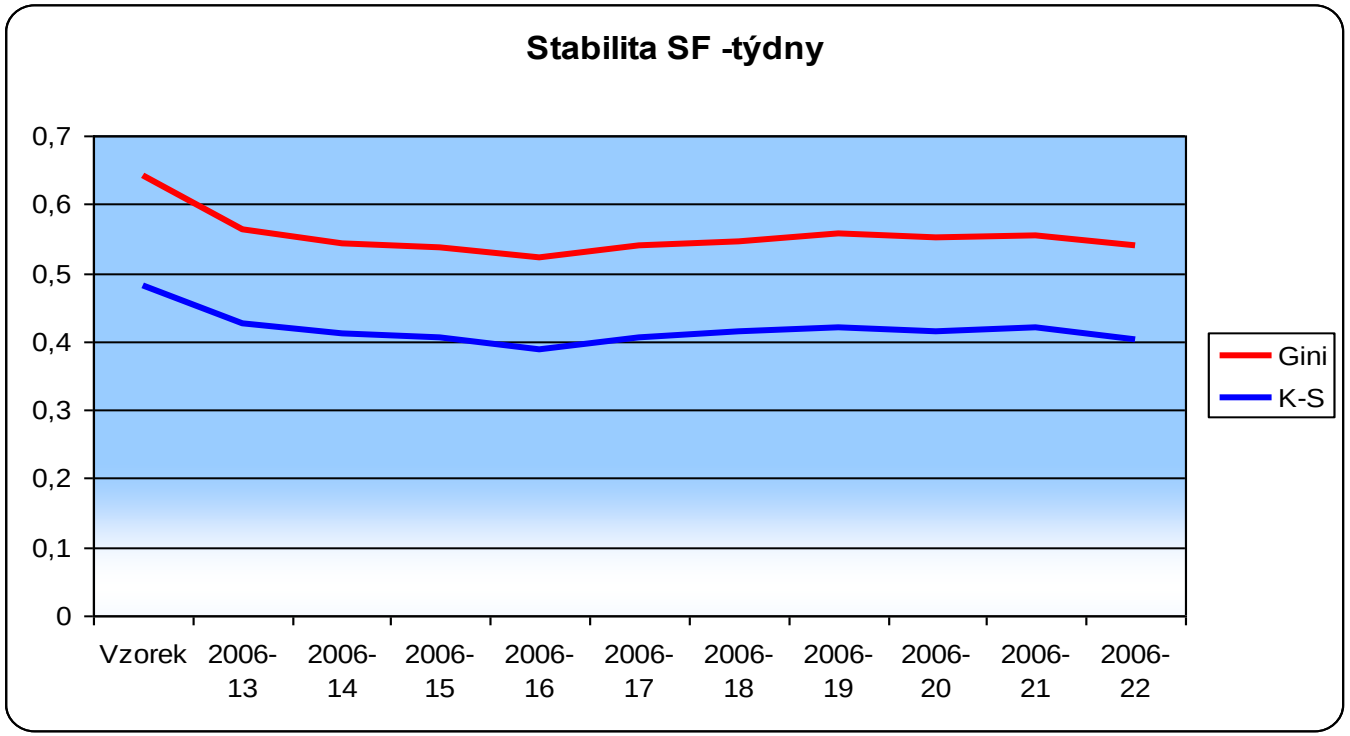
  - Přirozený posun v datech/změna demografické struktury dat
  - Databázové chyby
  - Změna datového zdroje
  - Změna definice/formátu vstupních dat
  - Změna datového univerza
  - Ostatní

Monitoring scoringových modelů

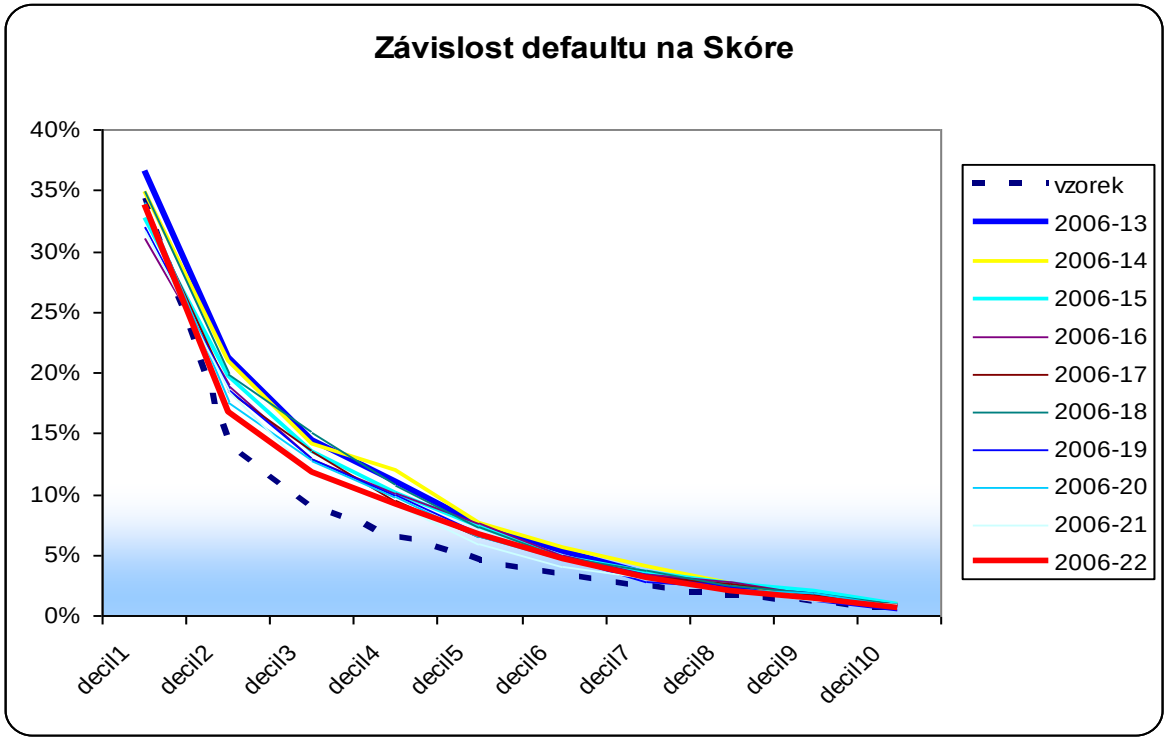
- Typickým příkladem prvního uvedeného důvodu je příjem klienta (všeobecným trendem je růst příjmu populace). Změnou definice/formátu vstupních dat je myšlena například situace, kdy je rozšířen číselník hodnot, kterých může vstupní proměnná nabývat. Změnou datového univerza je myšlen případ kdy je vyvinutý prediktivní model použit např. pro odlišný/nový segment portfolia nebo odlišný/nový produkt.

Monitoring scoringových modelů

- K-S, Gini:



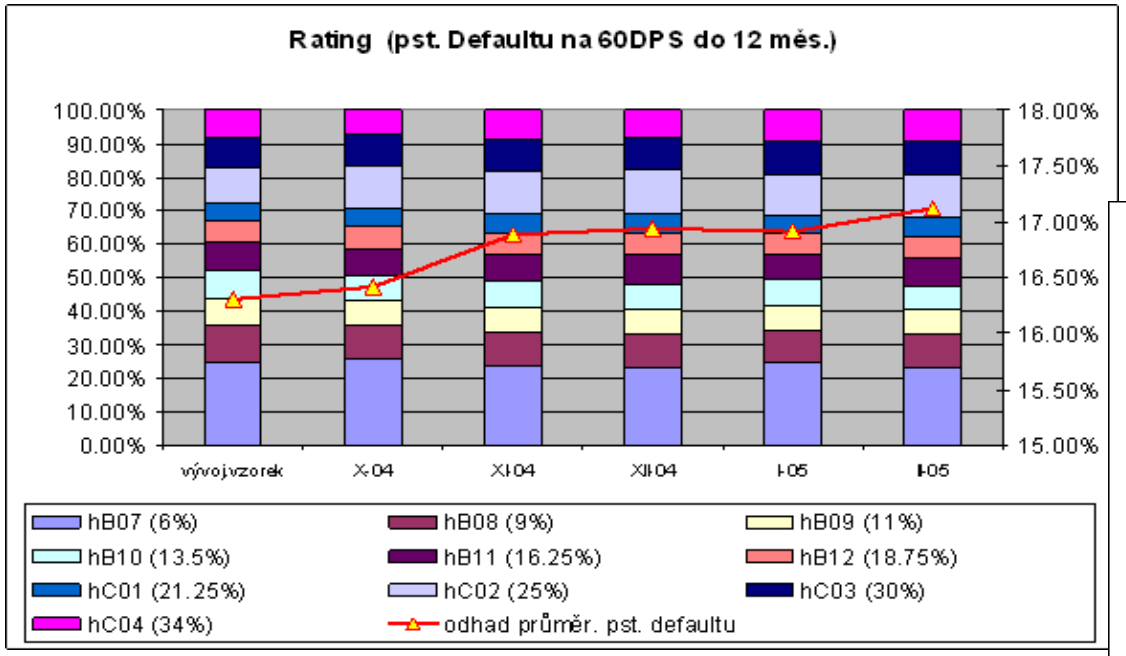
Monitoring scoringových modelů

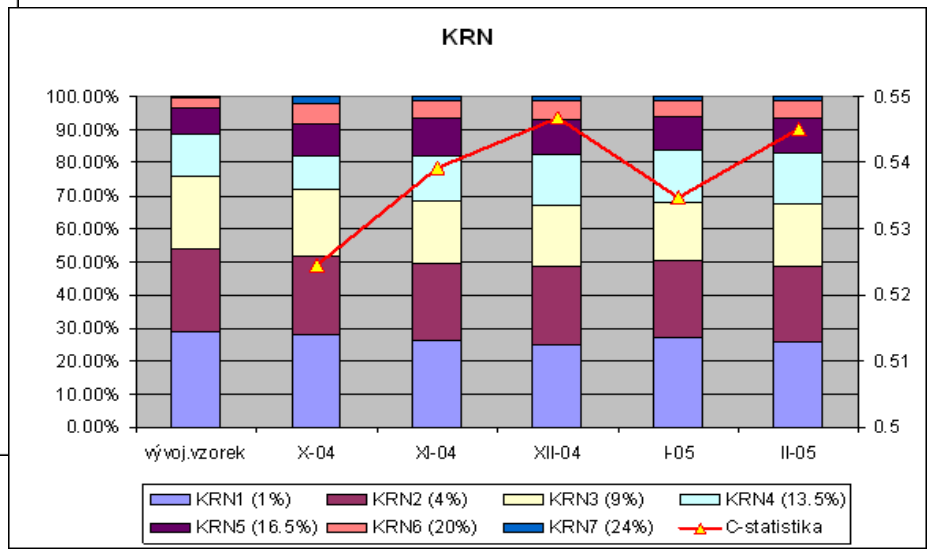


- Čím strmější křivka tím lépe.
- V průběhu času se zplošťuje – jde o to, jak moc.

Monitoring scoringových modelů

- c-statistika:





Monitoring scoringových modelů

- Chceme posoudit zda se distribuce skóre na vývojovém vzorku liší od distribuce skóre v daném časovém intervalu:

$$\chi^2 = \sum_{i=1}^{r} \frac{(O_i - E_i)^2}{E_i}$$

$$PSI = \sum_{i=1}^{r} (O_i - E_i) \ln\left(\frac{O_i}{E_i}\right)$$

Monitoring scoringových modelů

	výv. vzorek [1]	týden1 [2]	[3]=[2] -[1]	[4]=[2]/[1]	[5]=ln[4]	[6]=[3]*[5]
skóre_1	10,00%	5,63%	-0,044	0,563	-0,574	0,025
skóre_2	10,00%	11,21%	0,012	1,121	0,114	0,001
skóre_3	10,00%	11,00%	0,010	1,100	0,095	0,001
skóre_4	10,00%	10,97%	0,010	1,097	0,092	0,001
skóre_5	10,00%	10,31%	0,003	1,031	0,031	0,000
skóre_6	10,00%	10,12%	0,001	1,012	0,012	0,000
skóre_7	10,01%	9,62%	-0,004	0,961	-0,039	0,000
skóre_8	10,00%	9,89%	-0,001	0,989	-0,011	0,000
skóre_9	10,00%	10,31%	0,003	1,031	0,030	0,000
skóre_10	10,00%	10,94%	0,009	1,095	0,091	0,001
					PSI	0,030

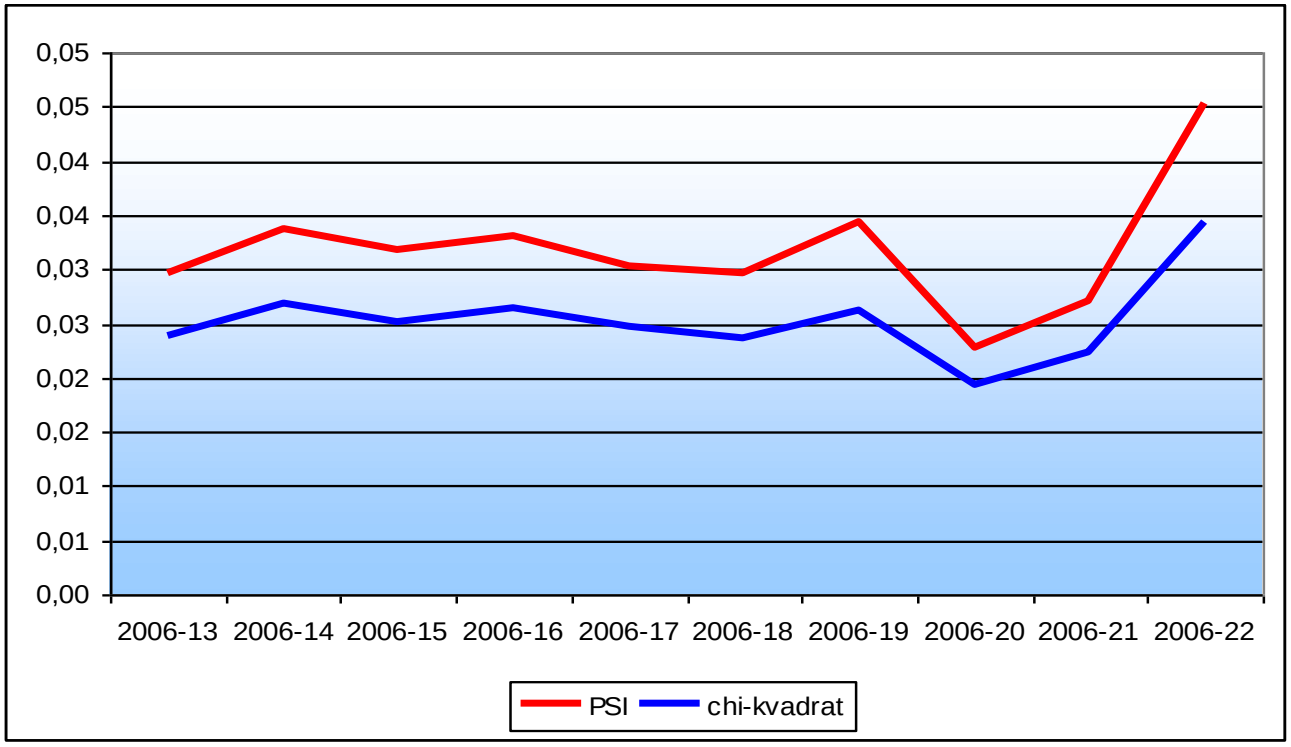
Monitoring scoringových modelů

$PSI \leq 0{,}1$ značí žádný nebo jen velmi malý rozdíl daných distribucí skóre.

$0{,}1 < PSI \leq 0{,}25$ znamená, že došlo k nějakému posunu distribuce, nicméně nikterak významnému.

$PSI > 0{,}25$ signalizuje významný posun v distribuci skóre, tj. zamítáme hypotézu o shodě daných distribucí.

Monitoring scoringových modelů

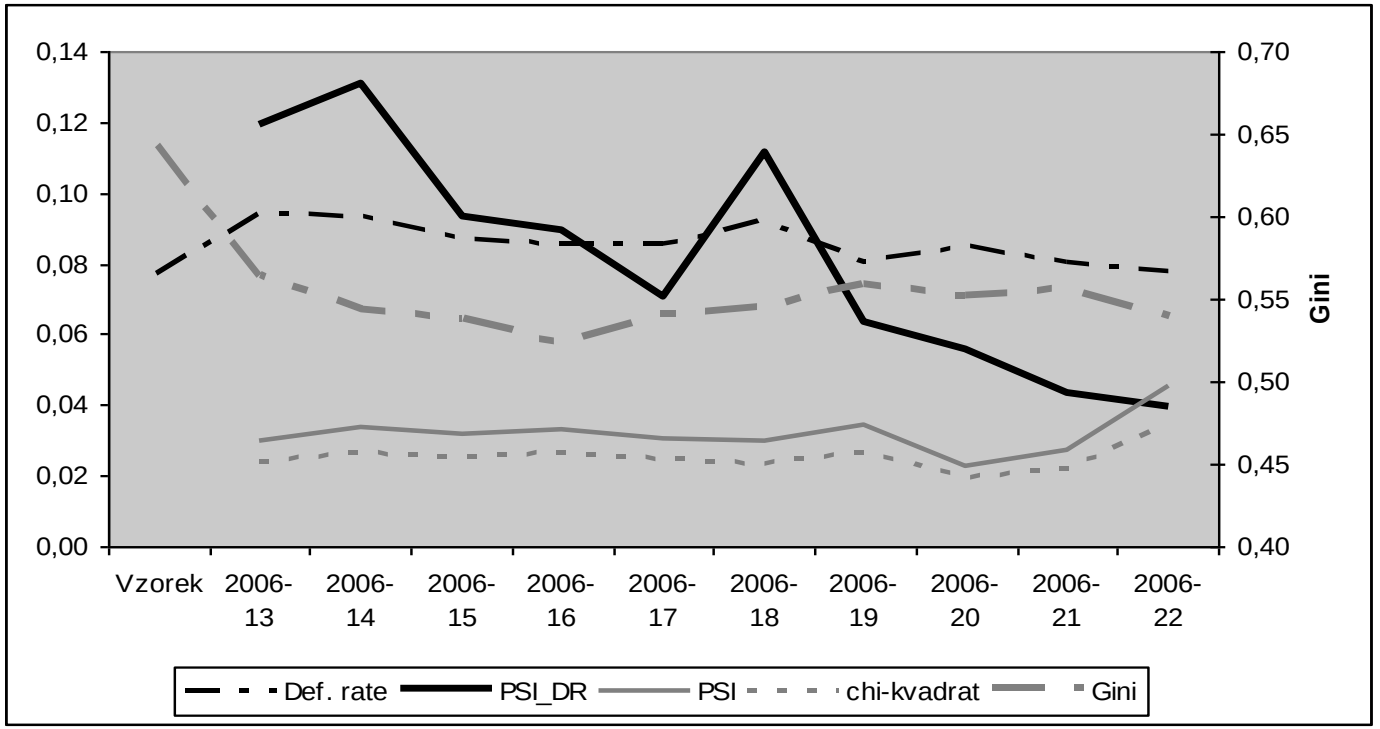


Monitoring scoringových modelů

$$PSI_{DR} = \sum_{i=1}^{r} (DR2_i - DR1_i) \ln\left(\frac{DR2_i}{DR1_i}\right)$$

	def_rate	Gini	PSI_DR	PSI	chi-kvardat
vzorek	7,69%	0,643			
200613	9,38%	0,564	0,120	0,030	0,024
200614	9,35%	0,542	0,131	0,034	0,027
200615	8,70%	0,537	0,093	0,032	0,025
200616	8,57%	0,523	0,089	0,033	0,026
200617	8,59%	0,540	0,071	0,030	0,025
200618	9,19%	0,544	0,111	0,030	0,024
200619	8,03%	0,558	0,063	0,034	0,026
200620	8,52%	0,552	0,055	0,023	0,019
200621	8,05%	0,555	0,043	0,027	0,022
200622	7,76%	0,539	0,039	0,045	0,034

Monitoring scoringových modelů



Champion-challenger (mistr – vyzyvatel)

- K rozšíření využití strategie champion-challenger došlo v devadesátých letech minulého století. Princip je velmi jednoduchý. Předpokládejme, že existuje nějaký způsob dělání něčeho (např. aktuálně používaný scoringový model pro schvalování/zamítání žádostí o úvěr). Tento způsob nazveme mistrem (champion). Nicméně existují další, jeden nebo více, alternativní způsoby jak dosáhnout téhož (nebo velmi podobného) cíle. Tyto nazveme vyzyvateli (challengers). Na náhodném vzorku otestujeme vyzyvatele a porovnáme s mistrem. To nám umožní nejen porovnat efektivnost vyzyvatelů a mistra, ale získáme možnost identifikovat existenci a rozsah vedlejších efektů. Výsledkem pak může být zjištění, že některý z vyzyvatelů je lepší než mistr a tento vyzyvatel se stane novým mistrem.

13. Reference

Literatura - knihy

- 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.

- 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.

Literatura - knihy

- 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.

- 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.

- 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
- www.kaggle.com