Seminar_3

Stefan Lyocsa

2024-10-09

Content

- Data cleaning and manipulation.
- Separating data into training and testing.
- Summary statistics:
  - mean,
  - sd,
  - quantiles,
  - skewness,
  - kurtosis,
  - normality test,
  - correlation table.
- Visualize data: Overlapping histograms, Ordered box-plots, x-y plots.
- Estimate OLS models:
  - Given model specifications.
  - Your own model specifications.
- Model prediction
- Forecast evaluation:
  - Statistical loss function.
  - Model confidence set

Data cleaning and manipulation

Load the original dataset into the environment

```
oc <- read.csv(file='C://Users//ckt//OneDrive - MUNI//Institutions//MU//ML Finance//2024//Week 3//octav
```

Let's inspect the small data.

```
head(oc,n=10)
```

```
##     X price year     km  kw    kraj      fuel trans
## 1   1 15290 2017 157017  85 BB kraj    Diesel     0
## 2   2  9300 2013 231000 125 PO kraj    Diesel     0
## 3   3 17990 2019 185500 110 BA kraj Benz\xedn     0
## 4   4 22500 2018 138477 110 ZA kraj    Diesel     1
## 5   5 25490 2018  87400 180 BA kraj Benz\xedn     1
## 6   6 16990 2018 180711 110 BB kraj    Diesel     1
## 7   7 24990 2018 120000 135 BB kraj    Diesel     1
## 8   8  5990 2010 266000  77 ZA kraj    Diesel     0
## 9   9  9200 2014 172000  77 ZA kraj    Diesel     0
## 10 10 13990 2016 168500 110 TT kraj Benz\xedn     1
```

```
tail(oc,n=10)
```

```
##       X price year    km  kw    kraj      fuel trans
## 891 891 30599 2020 35958 110 BA kraj    Diesel     1
## 892 892 30990 2021 40093 110 BA kraj Benz\xedn     1
## 893 893 30999 2020 48200 110 BA kraj    Diesel     1
## 894 894 31990 2022    50 110 NR kraj Benz\xedn     0
## 895 895 32000 2020 54000 110 BA kraj    Diesel     1
## 896 896 33490 2021  9873 110 ZA kraj Benz\xedn     1
## 897 897 33750 2020 21000 110 BA kraj    Diesel     1
## 898 898 34900 2022  9990 110 BA kraj    Diesel     1
## 899 899 34990 2022    15 110 NR kraj Benz\xedn     0
## 900 900 35990 2022  6385 110 NR kraj    Diesel     1
```

It seems that in the 'fuel' column we have special characters. Let's get rid of that. First we transform the values in the column to strings (text).

```
oc$fuel <- as.character(oc$fuel)
table(oc$fuel)
```

```
##
## Benz\xedn+Plyn      Benz\xedn         Diesel
##              6            115            779
```

Now we create three dummy variables. One for each type of fuel:

```
oc$diesel <- (oc$fuel == 'Diesel')*1
oc$petrol <- (oc$fuel == 'Benz\xedn')*1
oc$petgas <- (oc$fuel == 'Benz\xedn+Plyn')*1
head(oc,n=10)
```

```
##     X price year     km  kw    kraj      fuel trans diesel petrol petgas
## 1   1 15290 2017 157017  85 BB kraj    Diesel     0      1      0      0
## 2   2  9300 2013 231000 125 PO kraj    Diesel     0      1      0      0
## 3   3 17990 2019 185500 110 BA kraj Benz\xedn     0      0      1      0
## 4   4 22500 2018 138477 110 ZA kraj    Diesel     1      1      0      0
## 5   5 25490 2018  87400 180 BA kraj Benz\xedn     1      0      1      0
## 6   6 16990 2018 180711 110 BB kraj    Diesel     1      1      0      0
## 7   7 24990 2018 120000 135 BB kraj    Diesel     1      1      0      0
## 8   8  5990 2010 266000  77 ZA kraj    Diesel     0      1      0      0
## 9   9  9200 2014 172000  77 ZA kraj    Diesel     0      1      0      0
## 10 10 13990 2016 168500 110 TT kraj Benz\xedn     1      0      1      0
```

It seems good now. We remove the fuel variable, but also the first column seems uninformative. So why have it?

```
oc[,c('X','fuel')] = NULL
head(oc,n=10)
```

```
##    price year     km  kw    kraj trans diesel petrol petgas
## 1  15290 2017 157017  85 BB kraj     0      1      0      0
## 2   9300 2013 231000 125 PO kraj     0      1      0      0
## 3  17990 2019 185500 110 BA kraj     0      0      1      0
## 4  22500 2018 138477 110 ZA kraj     1      1      0      0
## 5  25490 2018  87400 180 BA kraj     1      0      1      0
## 6  16990 2018 180711 110 BB kraj     1      1      0      0
## 7  24990 2018 120000 135 BB kraj     1      1      0      0
## 8   5990 2010 266000  77 ZA kraj     0      1      0      0
```

```
## 9   9200 2014 172000  77 ZA kraj     0      1      0      0
## 10 13990 2016 168500 110 TT kraj     1      0      1      0
```

I would also rename variables:

```
names(oc)[c(4,5,6)] = c('power','region','transmission')
names(oc)
```

```
## [1] "price"        "year"         "km"           "power"        "region"
## [6] "transmission" "diesel"       "petrol"       "petgas"
```

We could transform year into age. The data were retrieved in 2022 so the age is 2022 - year

```
oc$age = 2022-oc$year
summary(oc$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   4.000   5.000   6.732   9.000  23.000
```

But we notice that there are not many older cars. Perhaps we should winsorize:

```
table(oc$age)
```

```
##
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
##   7  11  23  83 172 177  74  57  60  50  34  28  32  31  18  15   8   9   4   1
##  20  23
##   5   1
```

```
oc$age[oc$age >= 18] = 18
table(oc$age)
```

```
##
##   0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
##   7  11  23  83 172 177  74  57  60  50  34  28  32  31  18  15   8   9  11
```

The mileage is a little bit suspicious - why?

```
summary(oc$km)
```

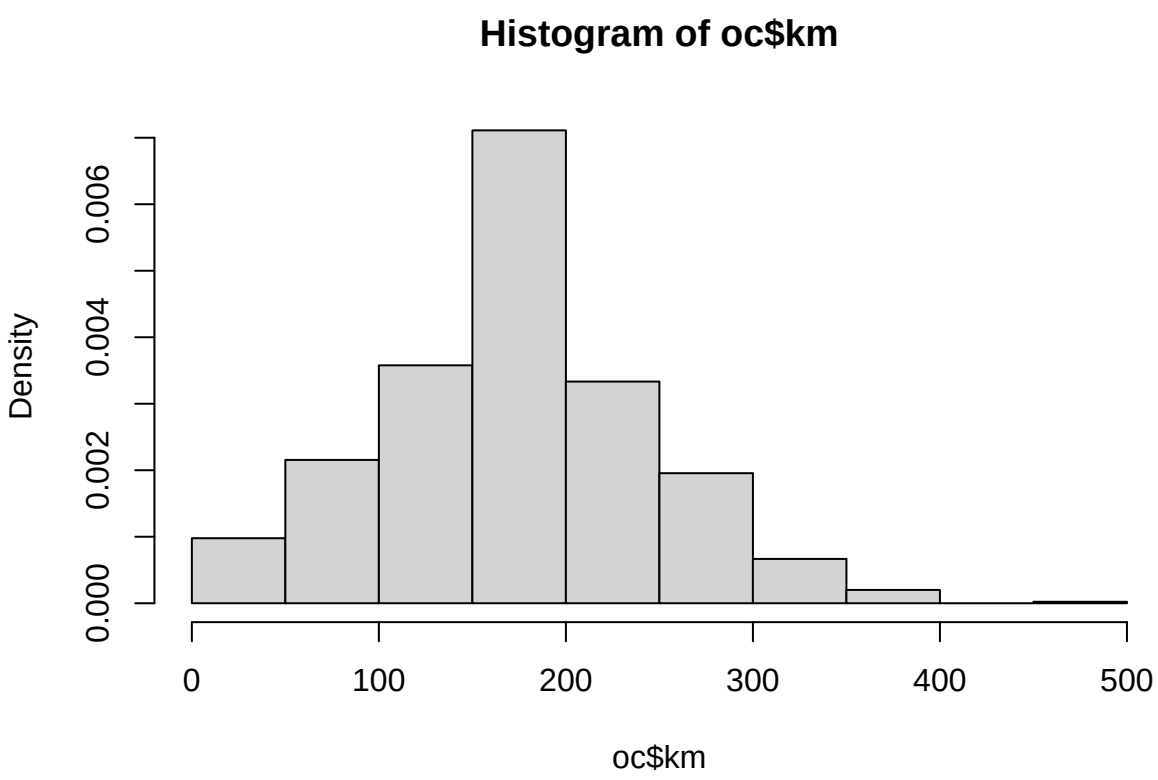
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       6  132610  174955  174823  216734  500000
```

I also do not like the unnecessary large numbers

```
oc$km = oc$km/1000
```

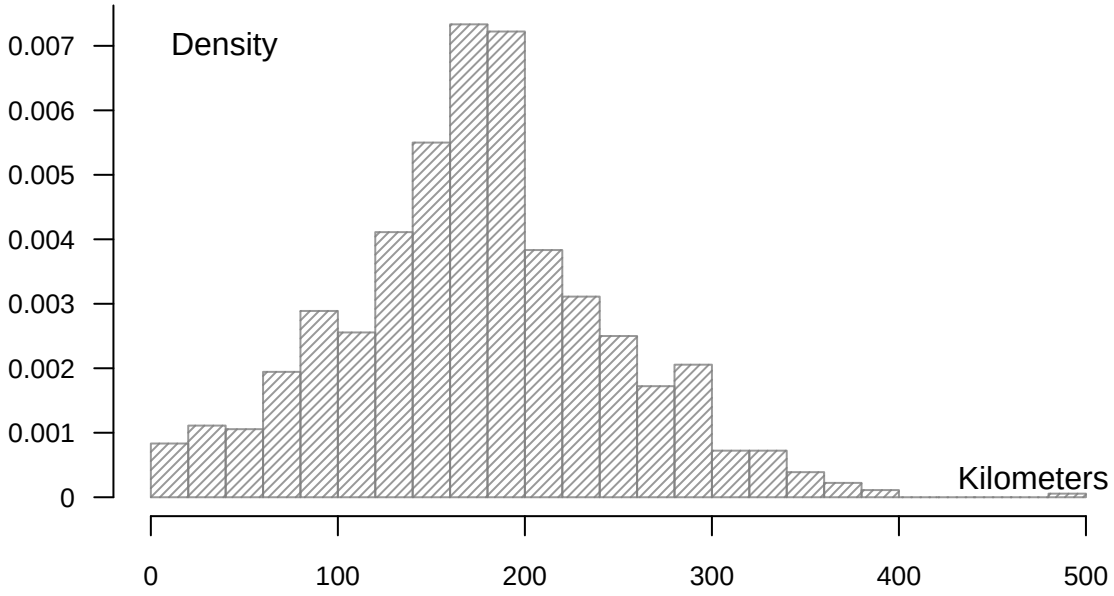
Let's visualize the data using a simple histogram

```
hist(oc$km,prob=T)
```



We can do better:

```
hist(oc$km,breaks=20,prob=T,yaxt='n',xaxt='n',density=30,col=rgb(0.5,0.5,0.5,alpha=0.8),xlab=c(),ylab=c
axis(1,at=seq(from=0,to=max(oc$km),by=100),label=seq(from=0,to=max(oc$km),by=100),cex.axis=0.85)
axis(2,at=seq(from=0,to=0.01,by=0.001),label=seq(from=0,to=0.01,by=0.001),cex.axis=0.85,las=2)
legend('bottomright',bty='n',legend='Kilometers')
legend('topleft',bty='n',legend='Density')
```

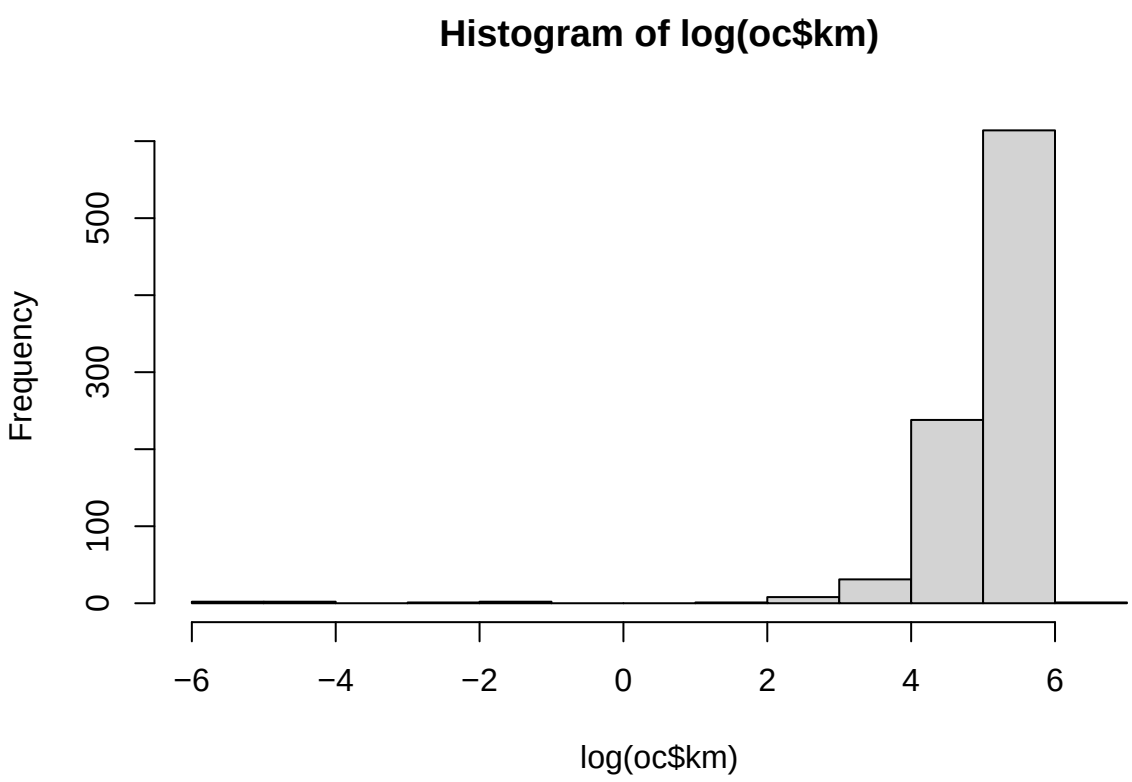


Relatively unused cars are perhaps unique and deserve a separate variable

```
oc$km10 = (oc$km<10)*1
```

Also notice that mileage is right-skewed so perhaps we should use a log-transform

```
hist(log(oc$km))
```



But we will not, the right-skewness just transformed into left-skewness. Let's take a look at the power of the cars.

```
summary(oc$power)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    63.0    85.0   103.0   100.8   110.0   180.0
```

```
table(oc$power)
```

```
##
##  63  66  74  75  77  81  85  90  96 100 103 110 118 125 132 135 140 147 162 169
##   2  11   5   7 121  62 196   3   4   1  63 274   9  16  12  89   3   8   8   2
## 180
##   4
```

It seems like there might be errors but also power levels. Let's create dummies for different ranges of car power. Here the domain knowledge might guide us to make more accurate power categories.

```
oc$power_lowest  = (oc$power<77)*1
oc$power_low     = (oc$power>=77 & oc$power<85)*1
oc$power_mid     = (oc$power>=85 & oc$power<103)*1
oc$power_high    = (oc$power>=103 & oc$power<118)*1
oc$power_highest = (oc$power>118)*1
```

We could also create interaction terms:

```
oc$eng1 = c(oc$power == 77 & oc$diesel == 1)*1
oc$eng2 = c(oc$power == 81 & oc$diesel == 1)*1
```

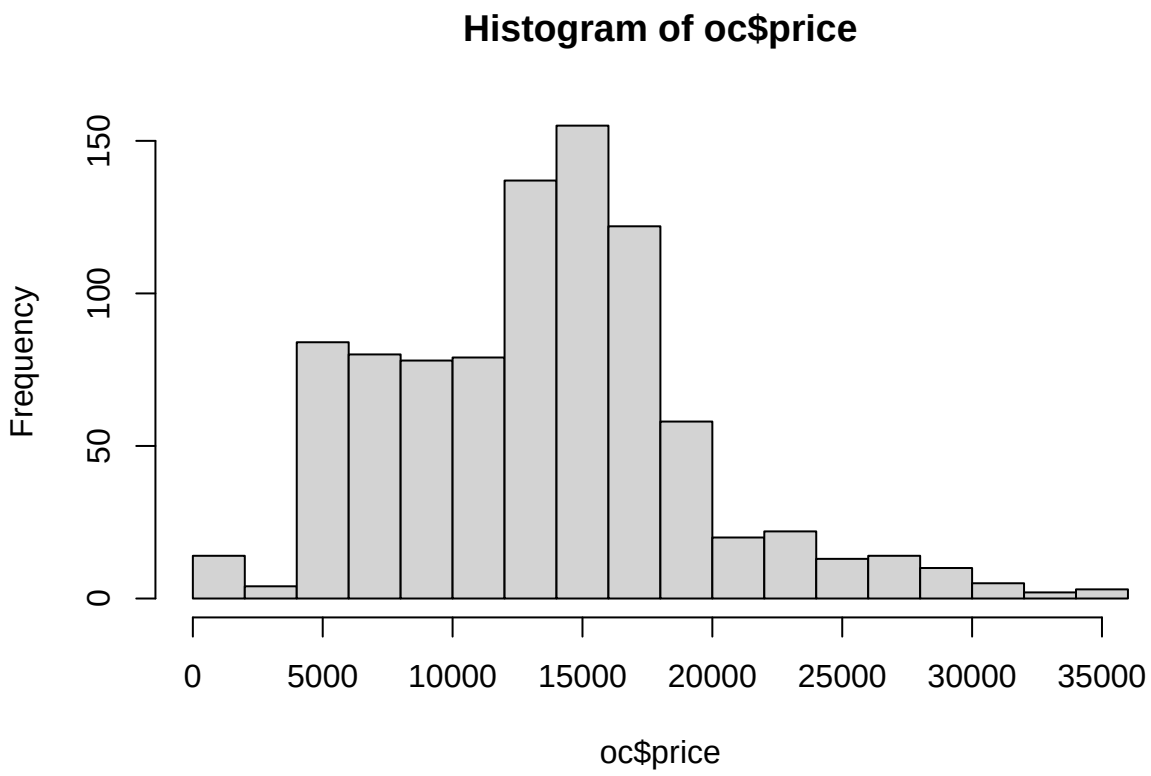
```
oc$eng3 = c(oc$power == 85 & oc$diesel == 1)*1
oc$eng4 = c(oc$power == 103 & oc$diesel == 1)*1
oc$eng6 = c(oc$power == 110 & oc$diesel == 1)*1
```

Now let's focus on the price - key variable of interest.

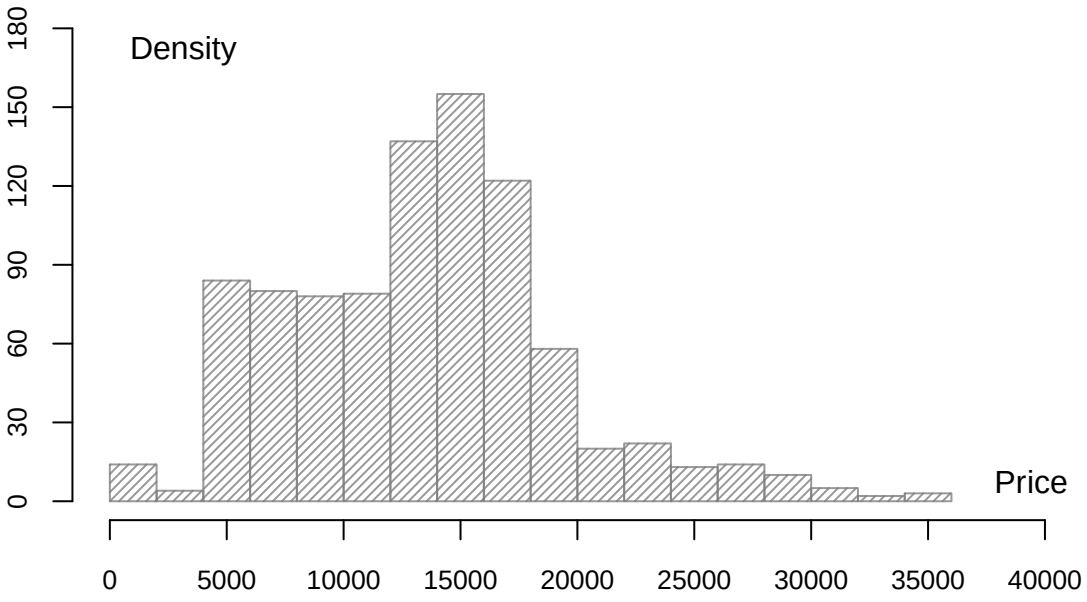
```
summary(oc$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     700    9300   13868   13606   16923   35990
```

```
hist(oc$price,prob=F,breaks=15)
```




```
# dev.off() # this is used if you want to reset any plot attributes (layout of the plot, size of text,
hist(oc$price,breaks=15,prob=F,yaxt='n',xaxt='n',ylim=c(0,180),xlim=c(0,40000),density=30,col=rgb(0.5,0
axis(1,at=seq(from=0,to=40000,by=5000),label=seq(from=0,to=40000,by=5000),cex.axis=0.85)
axis(2,at=seq(from=0,to=180,by=30),label=seq(from=0,to=180,by=30),cex.axis=0.85)
legend('bottomright',bty='n',legend='Price')
legend('topleft',bty='n',legend='Density')
```



For now, we will let the price be what it is. Finalize the dataset and remove unnecessary variables.

```
names(oc)
```

```
##  [1] "price"         "year"          "km"            "power"
##  [5] "region"        "transmission"  "diesel"        "petrol"
##  [9] "petgas"        "age"           "km10"          "power_lowest"
## [13] "power_low"     "power_mid"     "power_high"    "power_highest"
## [17] "eng1"          "eng2"          "eng3"          "eng4"
## [21] "eng6"
```

```
oc[,c('year','power','region','fuel')] = NULL
names(oc)
```

```
##  [1] "price"         "km"            "transmission"  "diesel"
##  [5] "petrol"        "petgas"        "age"           "km10"
##  [9] "power_lowest"  "power_low"     "power_mid"     "power_high"
## [13] "power_highest" "eng1"          "eng2"          "eng3"
## [17] "eng4"          "eng6"
```

Let's import new data (prepared dataset) from 2024 (October)

```
oc     <- read.csv(file='C://Users//ckt//OneDrive - MUNI//Institutions//MU//ML Finance//2024//octavia20
oc     <- oc[,-1]
oc$km  <- oc$km/1000
oc$age <- 2024 - oc$year
oc$year<- NULL
```

Separating data into training and testing

This should be random. In order to make our analysis reproducible we set initial conditions of the pseudo-random generator to be the same.

```
set.seed(50)
```

Now we randomly select 80% of observations into the training dataset and 20% into the testing dataset.

```
N = dim(oc)[1]
idx = sample(1:N,size=floor(0.8*N),replace=F)
ocr = oc[ idx,]
NR = dim(ocr)[1]
oct = oc[-idx,]
NT = dim(oct)[1]
```

I usually save the dataset at this stage:

Summary statistics

TASK Write a function() that will return 2 tables: * Descriptive statistics for given variables * Correlation table with statistical significance Find out if there are any two features that are excessively correlated (say above 0.95?).

```
NV = dim(ocr)[2]
table1 <- matrix(NA,nrow=NV,ncol=11) #Mean, SD, min, 5%, 25%, Median, 75%, 95%, max, skew, kurt
rownames(table1) <- names(ocr)
colnames(table1) <- c("Mean", "SD", "min", "5%", "25%", "Median", "75%", "95%", "max", "skew", "kurt")
library(moments)
for (i in 1:NV) {
  x          <- na.omit(ocr[,i],na.rm=T)
  table1[i,] <- c(mean(x),sd(x),quantile(x,p=c(0,0.05,0.25,0.50,0.75,0.95,1)),skewness(x),kurtosis(x))
}
table1 <- round(table1,2)
```

Now the correlation table:

```
# Correlation table
table2 <- matrix(NA,nrow=NV,ncol=NV)
rownames(table2) <- names(ocr)
colnames(table2) <- names(ocr)
for (i in 1:(NV-1)) {
  for (j in (i+1):NV) {
    m = cor.test(ocr[,i],ocr[,j],method='spearman')
    # The warnings -> we have dummy variables that might cause the issues (use 'kendall')
    table2[i,j] <- m$estimate
    table2[j,i] <- m$p.value
  }
}
```

```
## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
```

```
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
```

```
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
```

```
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
```

```
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
```

```
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
```

```
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
```

```
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
```

```
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties

## Warning in cor.test.default(ocr[, i], ocr[, j], method = "spearman"): Cannot
## compute exact p-value with ties
```

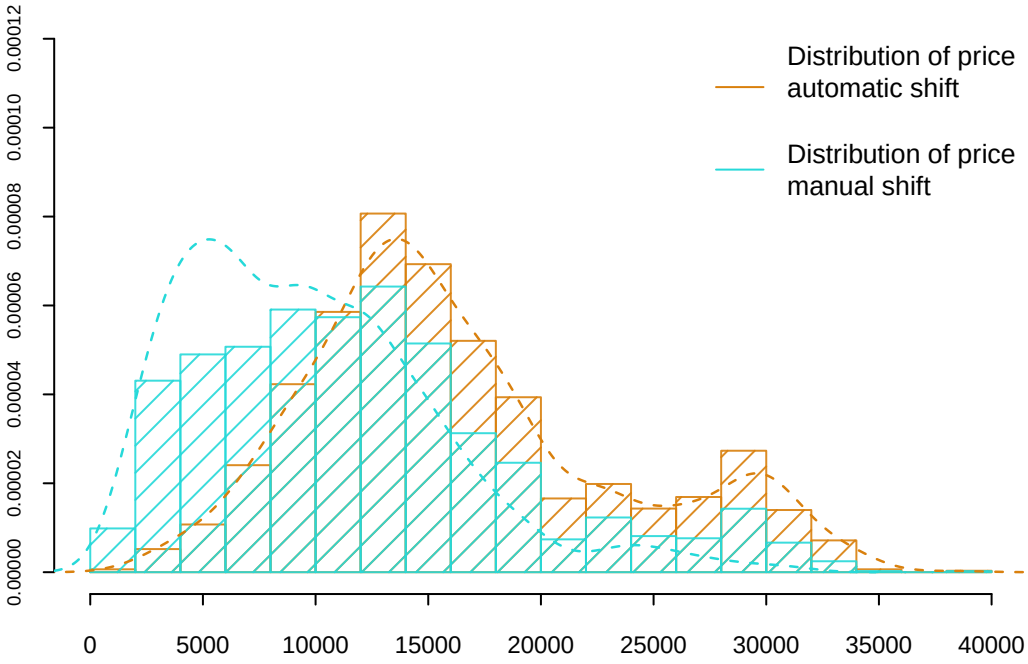
```
table2 <- round(table2,2)
```

TASK

Visualize data: Overlapping histograms, Ordered box-plots, x-y plots.

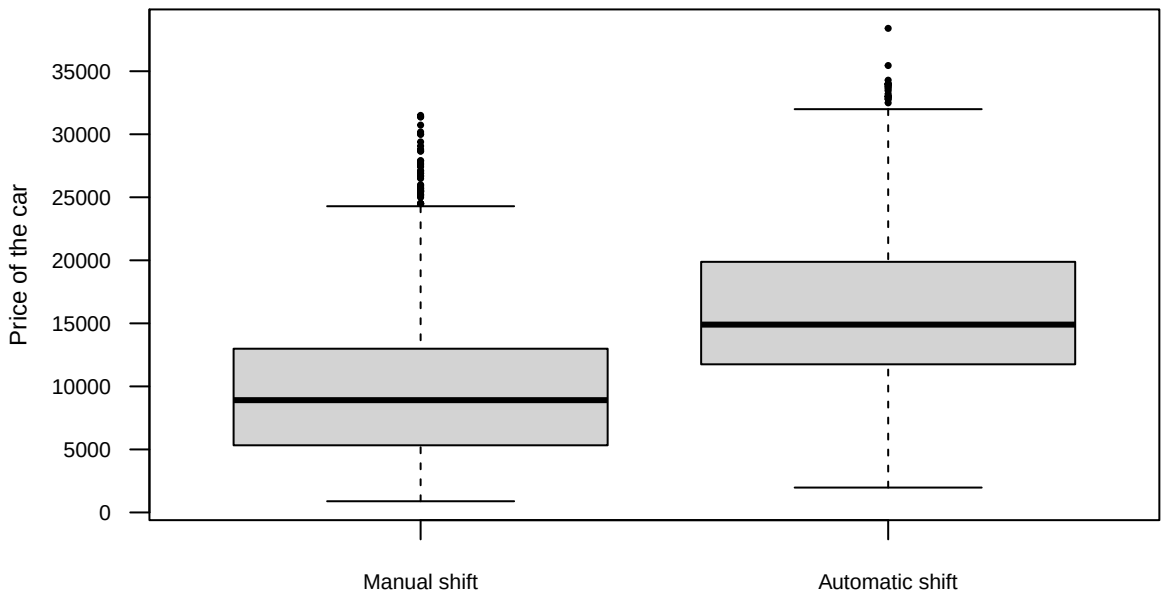
Let's take a look at some examples of plots. Overlapping histograms?

```
par(mfrow=c(1, 1)) # layout of figures - (rows,columns)
par(cex = 1.1)
par(oma = c(2, 2.0, 1.0, 1.0))
par(tcl = -0.25)
par(mgp = c(2, 0.6, 0))
par(mar = c(2.0, 3.0, 1.5, 0.5))
hist(oc$price[oc$trans==1],breaks=15,prob=T,xaxt='n',
     xlim=c(0,40000),density=10,
     col=rgb(0.85,0.5,0.05,alpha=0.9),
     xlab=c(),ylab=c(),main='',cex.axis=0.55,
     ylim=c(0,9.5^(-4)))
hist(ocr$price[oc$trans==0],breaks=15,prob=T,add=T,
     col=rgb(0.15,0.85,0.85,alpha=0.9),density=10)
axis(1,at=seq(from=0,to=40000,by=5000),label=seq(from=0,to=40000,by=5000),cex.axis=0.65)
lines(density(ocr$price[ocr$trans==1]),col=rgb(0.85,0.5,0.05),lwd=1.25,lty=2)
lines(density(ocr$price[ocr$trans==0]),col=rgb(0.15,0.85,0.85),lwd=1.25,lty=2)
legend('topright',bty='n',legend=c('Distribution of price\nautomatic shift\n',
                                   'Distribution of price\nmanual shift'),
       col=c(rgb(0.85,0.5,0.05),rgb(0.15,0.85,0.85)),cex=0.75,lty=1)
```



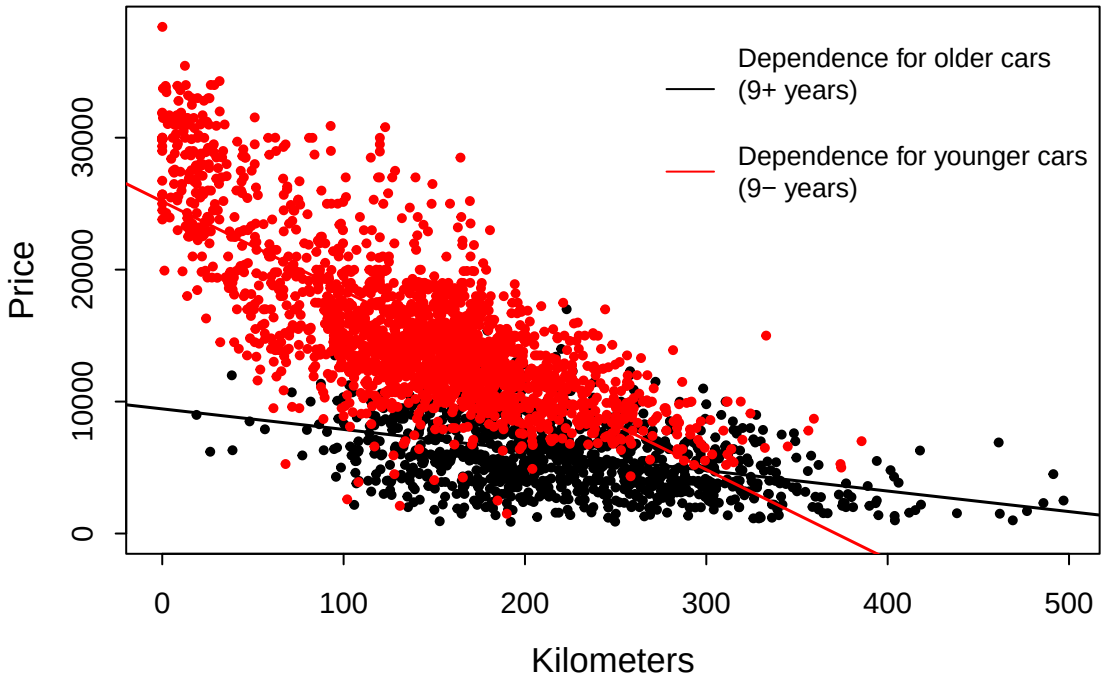
Separated box-plots?

```
boxplot(price~trans,data=ocr,pch=19,cex=0.35,yaxt='n',xlab='',
        ylab = 'Price of the car',xaxt='n',cex.lab=0.75)
axis(2,at=seq(from=0,to=40000,by=5000),label=seq(from=0,to=40000,by=5000),cex.axis=0.65,las=2)
axis(1,at=c(1,2),label=c('Manual shift', 'Automatic shift'),
     cex.axis=0.65)
```



Separated x-y plots?

```
par(mfrow=c(1, 1)) # layout of figures - (rows,columns)
par(cex = 1.1)
par(oma = c(2, 2.0, 1.0, 1.0))
par(tcl = -0.25)
par(mgp = c(2, 0.6, 0))
par(mar = c(3.0, 3.0, 1.5, 0.5))
plot(x=ocr$km[ocr$age > 9],y=ocr$price[ocr$age > 9],
     pch=19, cex=0.5, col='black',ylab='Price',xlab='Kilometers',
     ylim=c(0,max(ocr$price)),xlim=c(0,max(ocr$km)),
     cex.axis=0.85,cex.lab=0.95)
abline(lm(price~km,data=ocr[ocr$age > 9,]),lwd=1.5)
points(x=ocr$km[ocr$age <= 9],y=ocr$price[ocr$age <= 9],
      col='red',pch=19,cex=0.5)
abline(lm(price~km,data=ocr[ocr$age <= 9,]),lwd=1.5,
       col='red')
legend('topright',bty='n',legend=c('Dependence for older cars\n(9+ years)\n',
                                   'Dependence for younger cars\n(9- years)'),
       col=c('black','red'),cex=0.75,lty=1)
```



Estimate OLS models

Start with a simplest of models. Note that the model is estimated using the training dataset!

```
m1 = lm(price ~ km, data=ocr)
summary(m1)
```

```
##
## Call:
## lm(formula = price ~ km, data = ocr)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -15421  -2775     38   2808  15922
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 23357.464    194.356  120.18   <2e-16 ***
## km            -65.605      1.048  -62.62   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4632 on 3022 degrees of freedom
## Multiple R-squared:  0.5648, Adjusted R-squared:  0.5646
## F-statistic:  3922 on 1 and 3022 DF,  p-value: < 2.2e-16
```

Let's try age and combine it.

```
m2 = lm(price ~ age, data=ocr)
summary(m2)
```

```
##
## Call:
## lm(formula = price ~ age, data = ocr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14583.8  -2572.8   -644.8   1819.4  18892.3
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 21841.72     133.91  163.11   <2e-16 ***
## age         -1166.99      14.06  -83.03   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3876 on 3022 degrees of freedom
## Multiple R-squared:  0.6952, Adjusted R-squared:  0.6951
## F-statistic:  6893 on 1 and 3022 DF,  p-value: < 2.2e-16
```

```
m3 = lm(price ~ km + age, data=ocr)
summary(m3)
```

```
##
## Call:
## lm(formula = price ~ km + age, data = ocr)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15399.8  -2145.1   -437.5   1677.4  15342.8
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 24732.8737   138.8398  178.14   <2e-16 ***
## km            -33.3102     0.9377  -35.52   <2e-16 ***
## age          -836.3693    15.0341  -55.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3256 on 3021 degrees of freedom
## Multiple R-squared:  0.785,  Adjusted R-squared:  0.7849
## F-statistic:  5516 on 2 and 3021 DF,  p-value: < 2.2e-16
```

Now let's look at an interaction terms.

```
m4 = lm(price ~ km + age + I(km*age), data=ocr)
summary(m4)
```

```
##
## Call:
## lm(formula = price ~ km + age + I(km * age), data = ocr)
##
## Residuals:
```

```
##    Min     1Q Median     3Q    Max
## -15623  -1697   -223   1422  12538
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 29326.721    172.783  169.73   <2e-16 ***
## km            -64.073      1.162  -55.16   <2e-16 ***
## age         -1544.275     23.378  -66.06   <2e-16 ***
## I(km * age)     3.918      0.109   35.93   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2725 on 3020 degrees of freedom
## Multiple R-squared:  0.8494, Adjusted R-squared:  0.8493
## F-statistic:  5678 on 3 and 3020 DF,  p-value: < 2.2e-16
```

What if we add almost everything?

```
m5 = lm(price ~ km + + age + trans + combi + allw + ambi + styl +
          rs + dsg + scr + diesel + lpg + hybrid + cng + I(km*age), data=ocr)
summary(m5)
```

```
##
## Call:
## lm(formula = price ~ km + +age + trans + combi + allw + ambi +
##     styl + rs + dsg + scr + diesel + lpg + hybrid + cng + I(km *
##     age), data = ocr)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -15271  -1208    -47   1111   9588
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.739e+04  1.737e+02 157.688  < 2e-16 ***
## km          -6.523e+01  9.788e-01 -66.643  < 2e-16 ***
## age         -1.447e+03  1.957e+01 -73.951  < 2e-16 ***
## trans        1.346e+03  1.148e+02  11.728  < 2e-16 ***
## combi        2.656e+02  8.874e+01   2.993  0.00279 **
## allw         1.993e+03  1.583e+02  12.588  < 2e-16 ***
## ambi        -1.011e+03  1.214e+02  -8.327  < 2e-16 ***
## styl         2.160e+02  1.142e+02   1.891  0.05868 .
## rs           3.333e+03  1.429e+02  23.331  < 2e-16 ***
## dsg          6.002e+02  1.332e+02   4.506 6.87e-06 ***
## scr          1.136e+03  1.841e+02   6.173 7.61e-10 ***
## diesel       2.024e+02  1.017e+02   1.990  0.04670 *
## lpg         -6.245e+02  5.586e+02  -1.118  0.26369
## hybrid       2.549e+03  5.470e+02   4.659 3.31e-06 ***
## cng         -2.823e+03  9.635e+02  -2.930  0.00341 **
## I(km * age)  3.866e+00  8.683e-02  44.532  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2136 on 3008 degrees of freedom
## Multiple R-squared:  0.9079, Adjusted R-squared:  0.9074
```

```
## F-statistic:  1976 on 15 and 3008 DF,  p-value: < 2.2e-16
```

Model prediction

Now we will use models m1 to m5 and make predictions on the training data. I ll first create a matrix where I ll store all the predictions and the predicted price:

```
predictions = matrix(NA,nrow=NT,ncol=5+1)
colnames(predictions) = c('True',paste('p',1:5,sep=''))
predictions[,1] = oct$price
```

Predictions itself - from the model 1 and 2.

```
p1 = predict(m1,new=oct); summary(p1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -9793    9402   12292   12462   15529   23357
```

Notice that model to predicted negative price... is that possible? What to do?

```
p1[p1 < 0] = min(p1[p1>0]); summary(p1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   395.6  9402.1 12292.0 12510.0 15529.4 23357.1
```

```
predictions[,2] = p1
p2 = predict(m2,new=oct); summary(p2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -8500    9005   13673   12543   17174   21842
```

```
p2[p2 < 0] = min(p2[p2>0]); summary(p2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   835.9  9004.8 13672.8 12663.2 17173.8 21841.7
```

```
predictions[,3] = p2
p3 = predict(m3,new=oct); summary(p3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -8503    8938   12968   12536   16498   24733
```

```
p3[p3 < 0] = min(p3[p3>0]); summary(p3)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     591    8938   12968   12644   16498   24733
```

```
predictions[,4] = p3
p4 = predict(m4,new=oct); summary(p4)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -6864    7866   11925   12475   15908   29326
```

```
p4[p4 < 0] = min(p4[p4>0]); summary(p4)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   924.8  7866.2 11925.0 12507.0 15907.6 29326.4
```

```
predictions[,5] = p4
p5 = predict(m5,new=oct); summary(p5) # Again
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -6536    7628   11974   12457   15892   30861
```

```
p5[p5 < 0] = min(p5[p5>0]); summary(p5)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1103    7628   11974   12487   15892   30861
```

```
predictions[,6] = p5
```

Forecast evaluation:

Statistical loss function

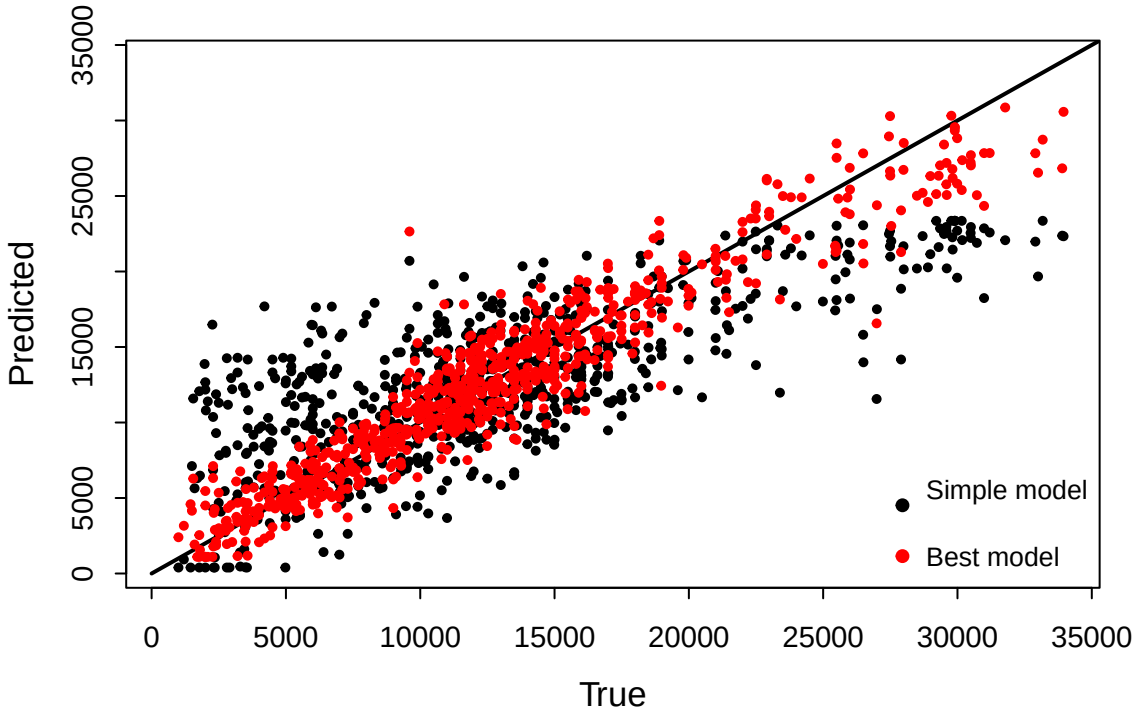
Now we evaluate forecasts using mean square error loss function

```
mse = matrix(NA,nrow=NT,ncol=5)
for (i in 1:5) mse[,i] = (predictions[,1] - predictions[,i+1])^2
apply(mse,2,mean)
```

```
## [1] 20777468 13595984  9496892  6970451  4247139
```

We can also check a figure - predicted vs. true.

```
par(mfrow=c(1, 1))
par(cex = 1.1)
par(oma = c(2, 2.0, 1.0, 1.0))
par(tcl = -0.25)
par(mgp = c(2, 0.6, 0))
par(mar = c(3.0, 3.0, 1.5, 0.5))
plot(x=predictions[,1],y=predictions[,2],
     pch=19, cex=0.5, col='black',ylab='Predicted',xlab='True',
     cex.axis=0.85,cex.lab=0.95,ylim=range(predictions),xlim=range(predictions))
lines(x=c(0,45000),y=c(0,45000),lwd=2)
points(x=predictions[,1],y=predictions[,6],
       col='red',pch=19,cex=0.5)
legend('bottomright',bty='n',legend=c('Simple model\n',
                                      'Best model'),
       col=c('black','red'),cex=0.75,pch=19)
```



Statistical significance test

Finally, we can run a hypothesis test which of the model - under the MSE loss function - led to be most accurate forecasts.

```
library(MCS)
MCSprocedure(mse)
```

```
##
## Model model_1 eliminated 2024-10-09 19:31:16.07331
## Model model_2 eliminated 2024-10-09 19:31:17.65469
## Model model_3 eliminated 2024-10-09 19:31:18.997339
## Model model_4 eliminated 2024-10-09 19:31:20.091636
## ##############################################################################################################
## Superior Set Model created   :
##         Rank_M       v_M MCS_M Rank_R       v_R MCS_R    Loss
## model_5      1 -6.289524     1      1 -6.289524     1 4247139
## p-value  :
## [1] 0
##
## ##############################################################################################################
##
## ------------------------------------------
## -          Superior Set of Models        -
## ------------------------------------------
##         Rank_M       v_M MCS_M Rank_R       v_R MCS_R    Loss
```

```
## model_5      1 -6.289524     1      1 -6.289524     1 4247139
##
## Details
## ---------------------------------------
##
## Number of eliminated models  :   4
## Statistic    :   Tmax
## Elapsed Time :   Time difference of 6.241868 secs
```

TASK

- Home assignment: Evaluate forecasts using mean absolute error
- Who provides an OLS model with highest accuracy?
- Does you model have a tendency to under- or over-perform?