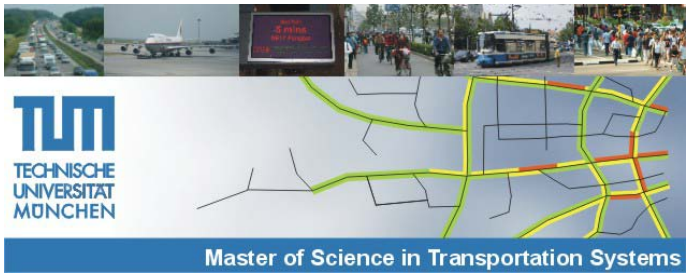
M.Sc. in ‚Transportation Systems‘



Applied Statistics in Transport
Descriptive Statistics, Data Preparation in R

Prof. Regine Gerike
Technische Universität München, mobil.TUM
regine.gerike@tum.de

Munich, 08/11/2010

Timetable Updated

1	18.10.2011	9:45-11:15 L	Welcome and Introduction
2	25.10.2011	8:00-9:30 LE	Introduction R, Tinn-R, Math
3	25.10.2011	9:45-11:15 L	Theory of probability
	01.11.2011	All Saints Day	
4	08.11.2011	8:00-9:30 LE	Descriptive analysis with R
5	08.11.2011	9:45-11:15 L	Descriptive statistics
6	15.11.2011	8:00-9:30 LE	Real world data input and preparation
	15.11.2011	9:45-11:15 L	Studentische Vollversammlung
7	22.11.2011	8:00-9:30 LE	Descriptive analysis with R, real world data
8	22.11.2011	9:45-11:15 L	Distributions
9	29.11.2011	8:00-9:30 LE	Reserve
10	29.11.2011	9:45-11:15 L	Distributions
11	06.12.2011	8:00-9:30 L	Inferential statistics
12	13.12.2011	8:00-9:30 L	Hypotheses testing
13	20.12.2011	8:00-9:30 L	Tests, statistical modelling
14	10.01.2012	8:00-9:30 L	ANOVA
15	17.01.2012	8:00-9:30 L	Regression
16	17.01.2012	9:45-11:15 L	Reserve
17	24.01.2012	8:00-9:30 L	Repetition
	02.02.2012	Tina Gehlert	Hypothesis-driven data analysis in transport
	03.02.2012	Tina Gehlert	Hypothesis-driven data analysis in transport
	07.02.2012	10:00-11:00	Exam

Mutual funds – data for illustrating descriptive statistics

- Description:
- The data contain 40 large general equity mutual funds, with information on return rates over 1 and 5 years as well as asset values, investment strategies and sales expenses.

Variable description

load

- 1 sales charge > 4:5 %
- 2 sales charge 4:5 %
- 3 no sales charge

type

- 1 capital appreciation
- 2 growth oriented
- 3 small company growth oriented
- 4 growth and income oriented
- 5 equity income oriented

exprat

- expense ratio= expenses/average net assets (expenses include management, operating and marketing fees)

assets market value of assets in millions of $
return5 5 year return in percentage
return1 1 year return in percentage
name name of the mutual fund

Reading data into R

```
#The data is contained in file mfund.data with space delimited values.
#The first row gives the variable names.
setwd("c:\\R\\")
mfund.data <- read.table(file = "mfund.data", header = T)
attach(mfund.data)
mfund.data[1:10, ]
#    load exprat type assets return5 return1                      name
#1     2   1.06    2  15252     117      45         Fidelity_Magellan
#2     3   0.37    4   7709      63      37              Windsor_Funds
#3     1   0.55    4   7643      92      30 Investment_Co_of_America
#4     1   0.77    4   6792      81      33     Washington_Mutual_Inv
#5     2   0.65    5   4790      62      28          Fidelity_Puritan
#6     2   0.70    5   4089      56      31       Fidelity_Equity-Inc
#7     1   0.75    4   3999      61      25                Pioneer_II
#8     1   0.60    4   3899      72      24           American_Mutual
#9     3   1.00    2   3761      89      27     Twentieth_Cent:Select
#10    1   0.50    4   3379      67      27           Affiliated_Fund
```

You find all the commands in AS_DescriptiveStatistics_mfundData.r.

Variable classification

Type	Variables
categorical	load, type (ordinal)
quantitative	exprat, assets, return5, return1

```
> str(mfund.data)
'data.frame':   40 obs. of  7 variables:
 $ load   : num  2 3 1 1 2 2 1 1 3 1 ...
 $ exprat : num  1.06 0.37 0.55 0.77 0.65 0.7 0.75 0.6 1 0.5 ...
 $ type   : num  2 4 4 4 5 5 4 4 2 4 ...
 $ assets : num  15252 7709 7643 6792 4790 ...
 $ return5: num  117.4 62.8 92 81.5 62.4 ...
 $ return1: num  44.9 37.1 30.2 32.8 27.7 ...
 $ name   : Factor w/ 40 levels "Affiliated Fund",..: 14 39 22 38 15
```

Absolute, relative frequencies

```
> table(load)
load
 1  2  3
21  5 14
> table(type)
type
 1  2  4  5
 4 12 18  6
```

```
> n<-length(load)
> n
[1] 40
> table(load)/n
load
    1     2     3
0.525 0.125 0.350
> table(type)/n
type
   1    2    4    5
0.10 0.30 0.45 0.15
> sum(table(type)/n)
[1] 1
```

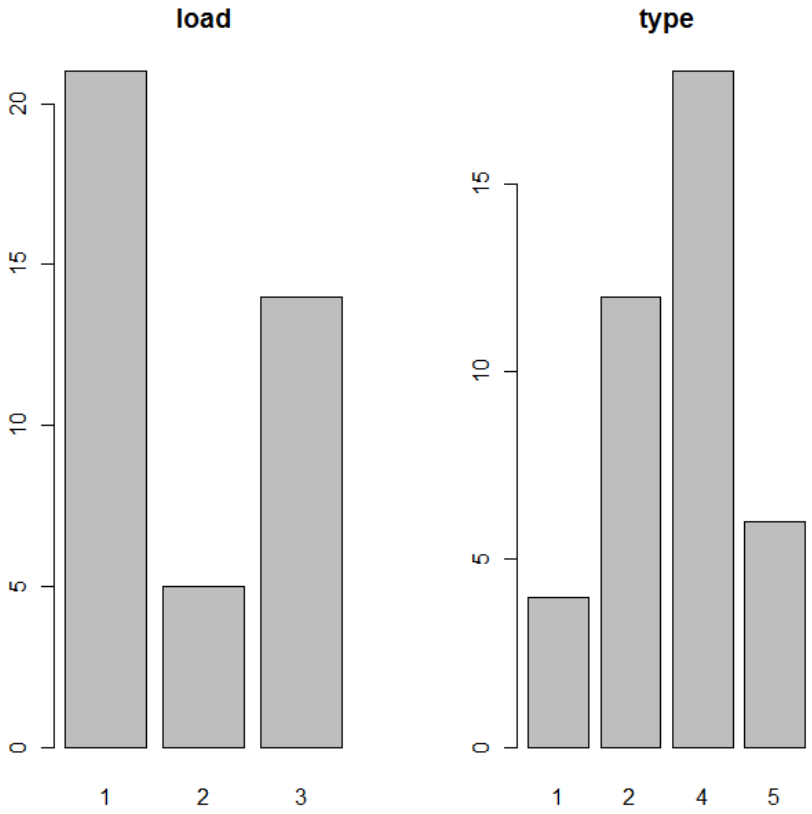
Grouping of expense ratio

- Quantitative variables can be grouped by using cut, where the bin limits have to be provided.

```
> exprat.group <- cut(exprat, breaks = c(0, 0.5, 1, 1.5, 2))
> exprat.group
 [1] (1,1.5] (0,0.5] (0.5,1] (0.5,1] (0.5,1] (0.5,1] (0.5,1] (0.5,1] (0.5,1] (0,0.5] (1.5,2] (0
[26] (0.5,1] (1,1.5] (0.5,1] (0.5,1] (0.5,1] (0.5,1] (0.5,1] (0.5,1] (0.5,1] (0.5,1] (0,0.5] (1
Levels: (0,0.5] (0.5,1] (1,1.5] (1.5,2]
> table(exprat.group)/n
exprat.group
(0,0.5] (0.5,1] (1,1.5] (1.5,2]
  0.125   0.750   0.100   0.025
```

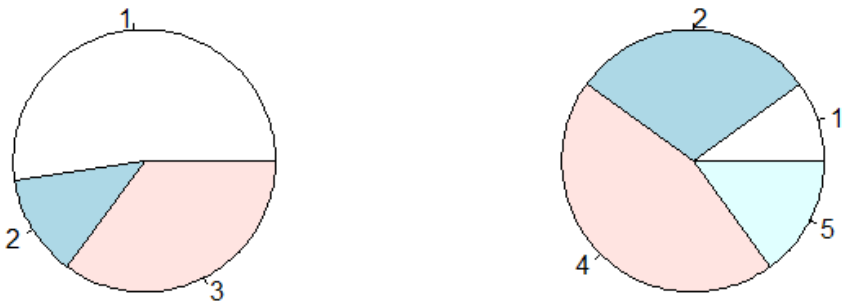
Bar plots of load and type

```
> par(mfrow = c(1, 2))
> barplot(table(load), main = "load")
> barplot(table(type), main = "type")
```



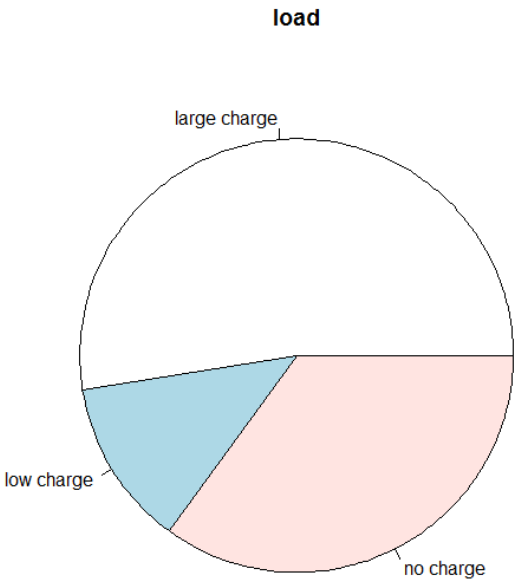
Pie plots of load and type

```
> par(mfrow = c(1, 2))
> pie(table(load), main = "load")
> pie(table(type), main = "type")
```



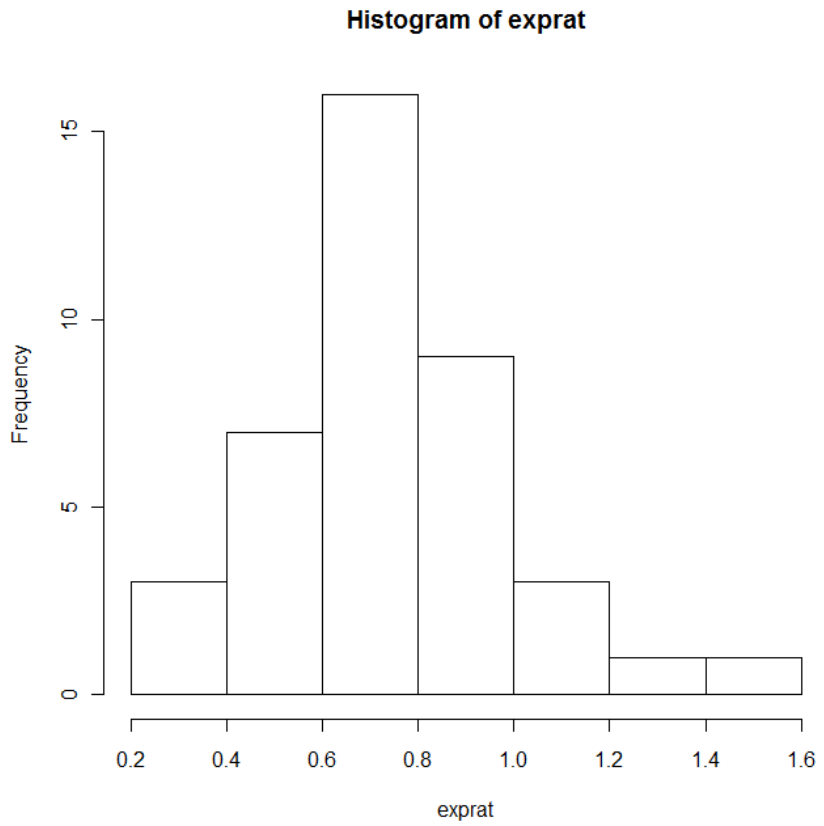
Load with labels

```
> par(mfrow = c(1, 1))
> freq.load <- table(load)
> names(freq.load) <- c("large charge", "low charge", "no charge")
> pie(freq.load, labels = names(freq.load), main = "load")
```



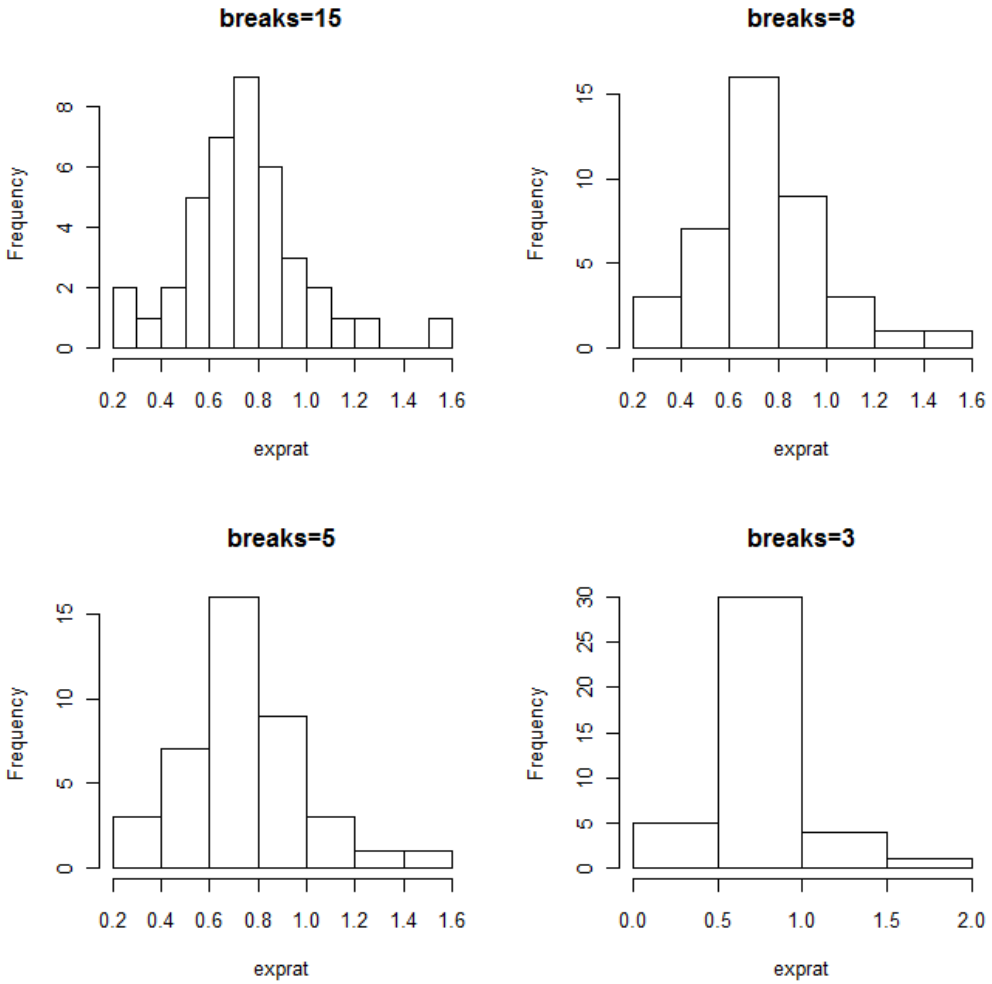
Histogram of expense ratio

```
> hist(exprat)
```



Histogram with different number of bins

```
> par(mfrow = c(2, 2))
> hist(exprat, breaks = 15, main = "breaks=15")
> hist(exprat, breaks = 8, main = "breaks=8")
> hist(exprat, breaks = 5, main = "breaks=5")
> hist(exprat, breaks = 3, main = "breaks=3")
```



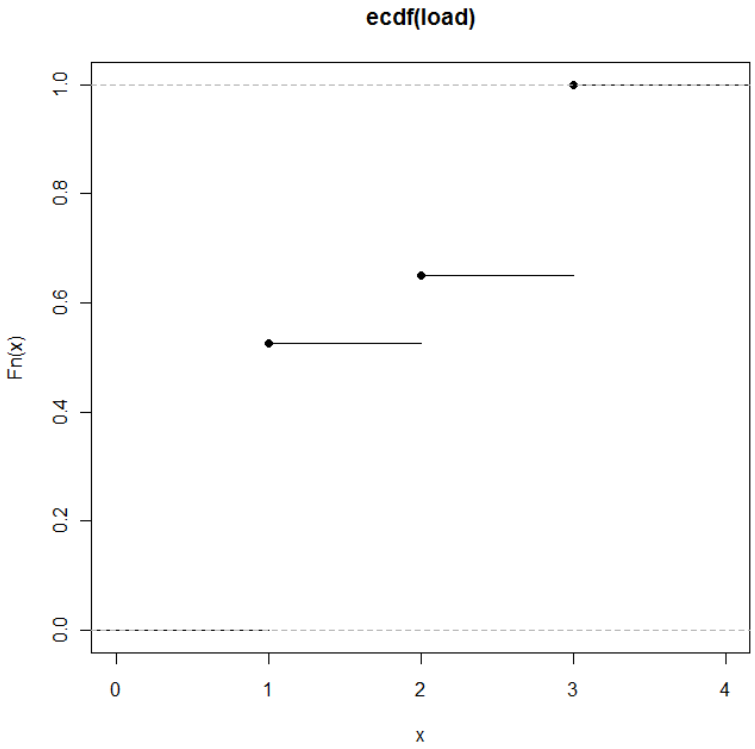
Cumulative frequencies

- The function *table* can be used to obtain frequencies, of which we can obtain cumulative frequencies through the function *cumsum*.

```
> table(load)/n
load
    1     2     3
0.525 0.125 0.350
> cumsum(table(load)/n)
    1     2     3
0.525 0.650 1.000
```

Empirical distribution function

```
> par(mfrow = c(1, 1))
> plot(ecdf(load))
```



Empirical mean and median, range

```
> mean(load)
[1] 1.8
> median(load)
[1] 1
> mean(exprat)
[1] 0.76
> median(exprat)
[1] 0.77
> range(load)
[1] 1 3
```

Empirical mode

- For qualitative variables the empirical mode is the value which occurs most, while for quantitative variables we need to group the variables and use the midpoint of the bin, which contains the most observations.

```
> table(load)
load
 1  2  3
21  5 14
> table(exprat.group)
exprat.group
(0,0.5] (0.5,1] (1,1.5] (1.5,2]
      5      30       4       1
> table(cut(exprat, breaks = seq(0, 2, length = 11)))

  (0,0.2] (0.2,0.4] (0.4,0.6] (0.6,0.8]   (0.8,1]   (1,1.2] (1.2,1.4] (1.4,1.6] (1.6,1.8]   (1.8,2]
        0         3         7        16         9         3         1         1         0         0
```

- This shows that .75 is an estimate of the mode for exprat, if 4 bins are used and .7 , if 10 bins are used.

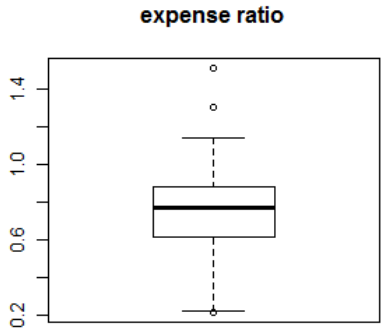
Empirical quantiles

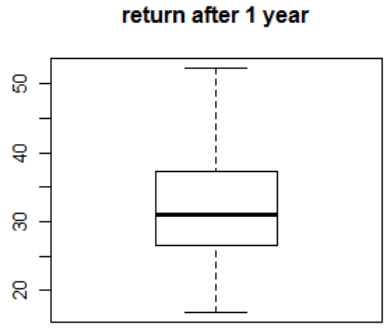
```
> quantile(exprat, probs = seq(0, 1, 0.2))
   0%   20%   40%   60%   80%  100%
0.210 0.574 0.700 0.790 0.908 1.510
> quantile(load, probs = seq(0, 1, 0.2))
  0%  20%  40%  60%  80% 100%
   1    1    1    2    3    3
```

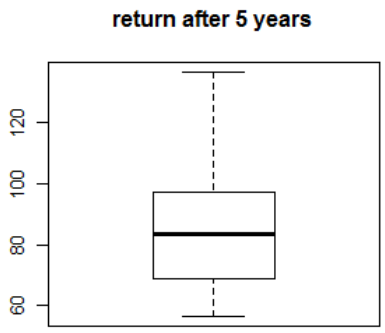
- Note that for categorical variables the quantiles do not need to be unique.

Individual box plots

```
> par(mfrow = c(2, 2))
> boxplot(exprat, main = "expense ratio")
> boxplot(return1, main = "return after 1 year")
> boxplot(return5, main = "return after 5 years")
```

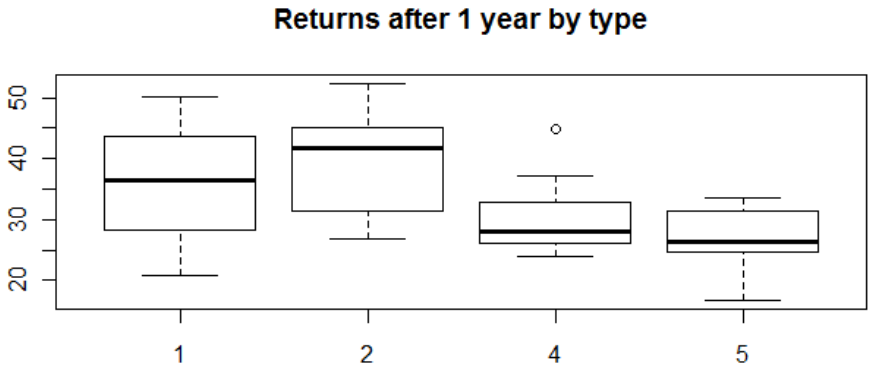


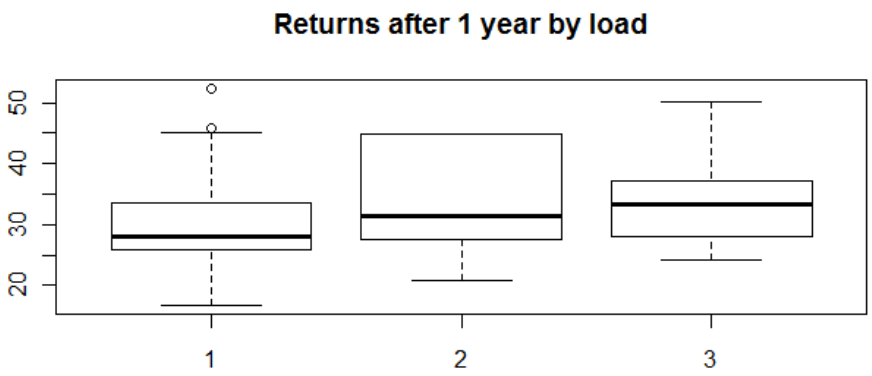




Boxplots by groups

```
> par(mfrow = c(2, 1))
> boxplot(return1 ~ type, main = "Returns after 1 year by type")
> boxplot(return1 ~ load, main = "Returns after 1 year by load")
```





Variance estimates for several groups

- Value for company 37 of return5 is missing, therefore we remove this company for this calculation since apply cannot handle missing values.

```
> which(is.na(return5))
[1] 37
> (1:n)[is.na(return5)]
[1] 37
> apply(cbind(return1, return5, exprat)[-37, ], 2, sd)
   return1    return5     exprat
 8.3289071 22.4833304  0.2521883
> apply(cbind(return1, return5, exprat)[-37, ], 2, IQR)
return1 return5  exprat
 10.525  28.260   0.260
> apply(cbind(return1, return5, exprat)[-37, ], 2, mad) #Median Absolute Deviation
  return1   return5    exprat
 7.101654 22.164870  0.177912
```

Skewness and kurtosis

```
library(fBasics)
skewness(return1)
#[1] 0.6
attr(,"method")
#[1] "moment"
kurtosis(return1)
#[1] -0.52
attr(,"method")
#[1] "excess"
```

Summary statistics using *summary*

The function summary gives minimum and maximum, the 25% , 50% and 75% quantile as well as the mean. It applies to single variables or dataframes.

```
> summary(mfund.data)
      load           exprat            type           assets
 Min.   :1.000   Min.   :0.2100   Min.   :1.00   Min.   : 1193
 1st Qu.:1.000   1st Qu.:0.6225   1st Qu.:2.00   1st Qu.: 1522
 Median :1.000   Median :0.7700   Median :4.00   Median : 2372
 Mean   :1.825   Mean   :0.7608   Mean   :3.25   Mean   : 3012
 3rd Qu.:3.000   3rd Qu.:0.8800   3rd Qu.:4.00   3rd Qu.: 3236
 Max.   :3.000   Max.   :1.5100   Max.   :5.00   Max.   :15252

    return5          return1                                name
 Min.   : 56.44   Min.   :16.82   Affiliated_Fund              : 1
 1st Qu.: 68.88   1st Qu.:26.59   AIM_Equity:Weingarten_Eq     : 1
 Median : 83.64   Median :31.11   Amcap_Fund                   : 1
 Mean   : 87.58   Mean   :32.64   Amer_Cap_Pace                : 1
 3rd Qu.: 97.14   3rd Qu.:37.17   American_Mutual              : 1
 Max.   :136.68   Max.   :52.30   Dean_Witter_Divid_Gro        : 1
 NA's   :  1.00                   (Other)                     :34
```

Contingency tables for qualitative and grouped variables

First we classify according to two variables

```
> table(load, type)
     type
load  1  2  4  5
   1  0  7 10  4
   2  1  1  1  2
   3  3  4  7  0
```

Higher order contingency tables

Now with three variable it is better to use *ftable*. It produces a flat table.

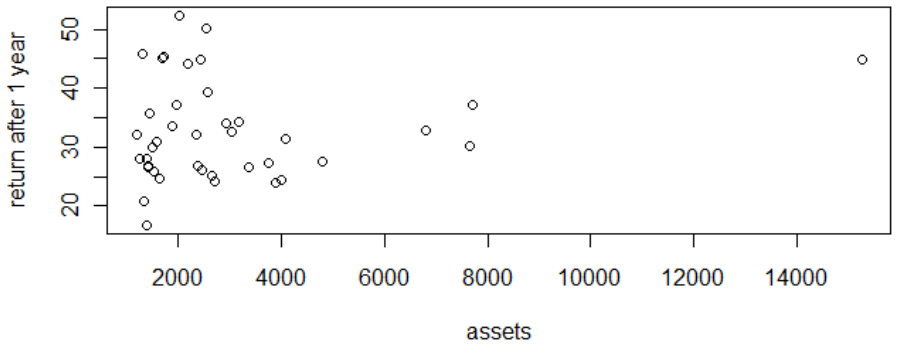
```
> table(load, type)
    type
load  1  2  4  5
   1  0  7 10  4
   2  1  1  1  2
   3  3  4  7  0
> ftable(load, type)
     type  1  2  4  5
load
1          0  7 10  4
2          1  1  1  2
3          3  4  7  0
```

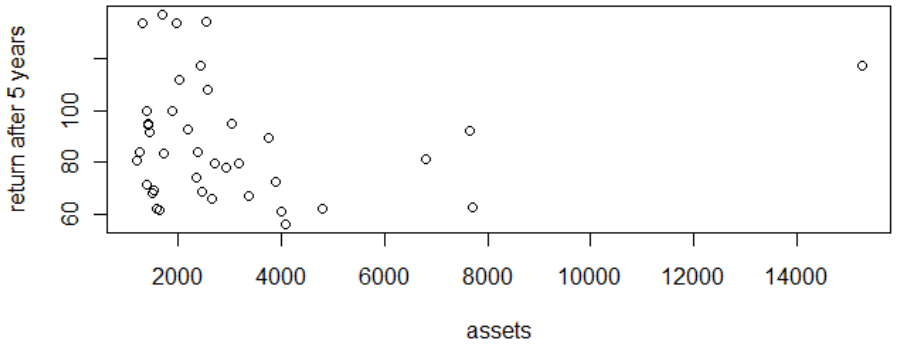
```
> table(load, type, exprat.group)
, , exprat.group = (0,0.5]

    type
load 1 2 4 5
   1 0 0 2 0
   2 0 0 0 0
   3 0 1 2 0

, , exprat.group = (0.5,1]

    type
load 1 2 4 5
   1 0 6 8 4
   2 0 0 1 2
   3 2 3 4 0

, , exprat.group = (1,1.5]
```

```
> ftable(load, type, exprat.group)
                exprat.group (0,0.5] (0.5,1] (1,1.5] (1.5,2]
load type
1    1                             0        0        0        0
     2                             0        6        1        0
     4                             2        8        0        0
     5                             0        4        0        0
2    1                             0        0        1        0
     2                             0        0        1        0
     4                             0        1        0        0
     5                             0        2        0        0
3    1                             0        2        1        0
     2                             1        3        0        0
     4                             2        4        0        1
     5                             0        0        0        0
```

Graphical display of quantitative variables

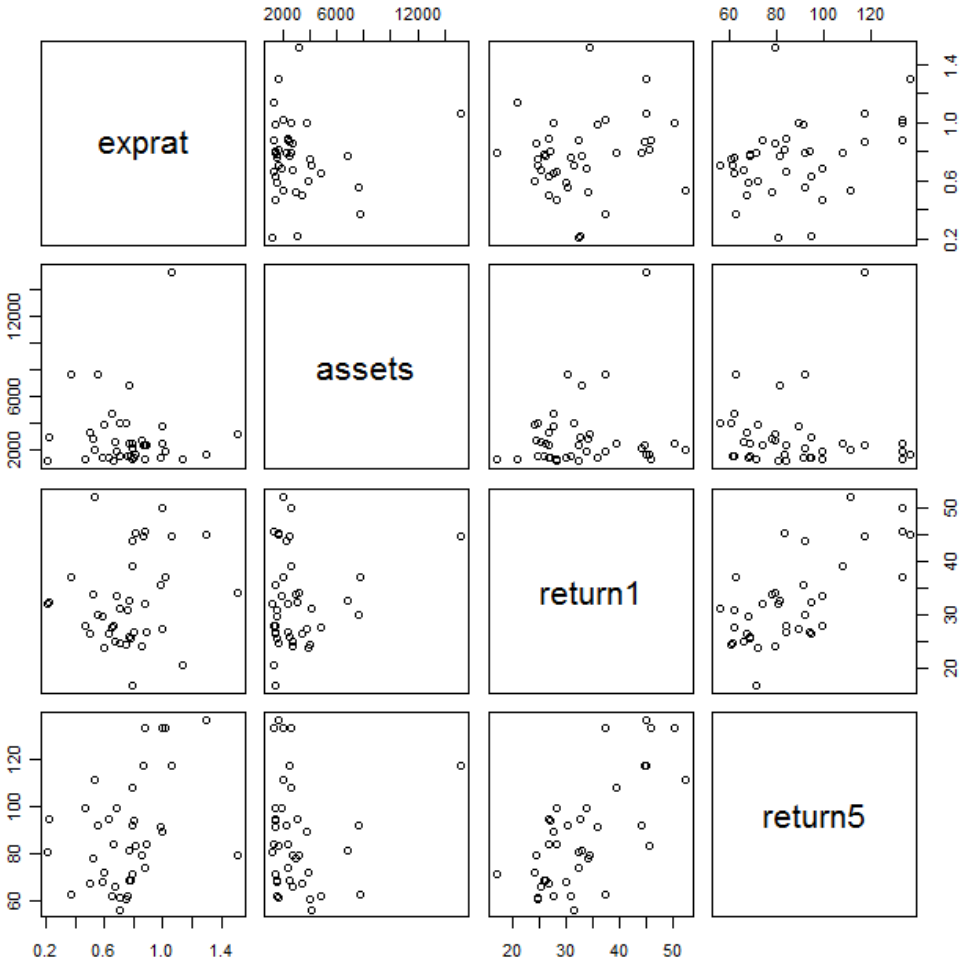
```
> par(mfrow = c(2, 1))
> plot(assets, return1, xlab = "assets", ylab = "return after 1 year")
                                                 'return after 5 years")
```





Pairs plots for several quantitative variables

```
> pairs(cbind(exprat, assets, return1, return5))
```

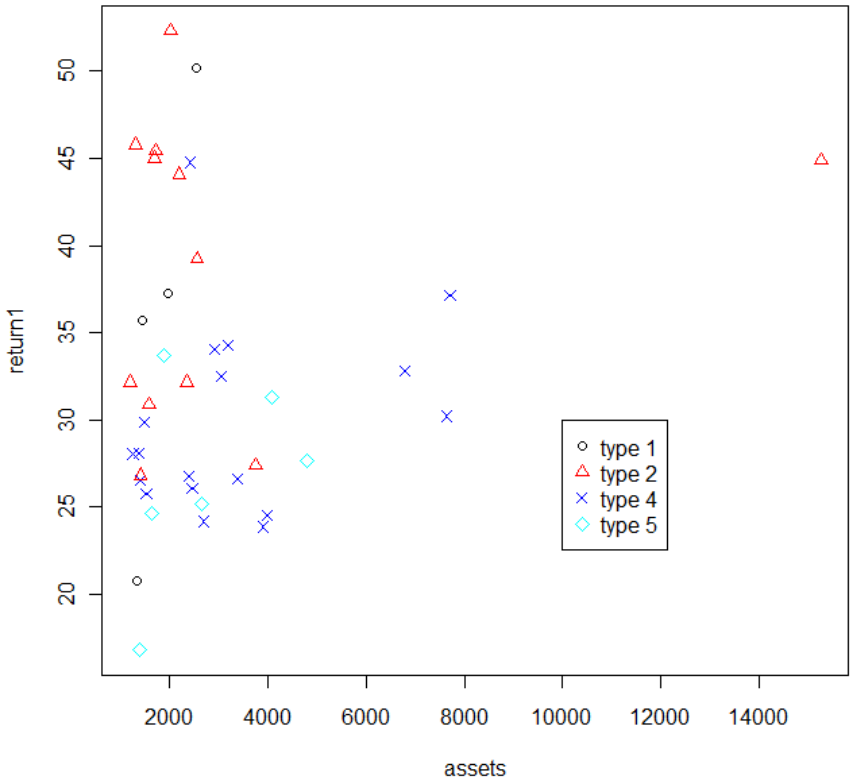


Empirical correlation between return variables

```
> cor(return1, return5, use = "complete.obs")
[1] 0.7049173
```

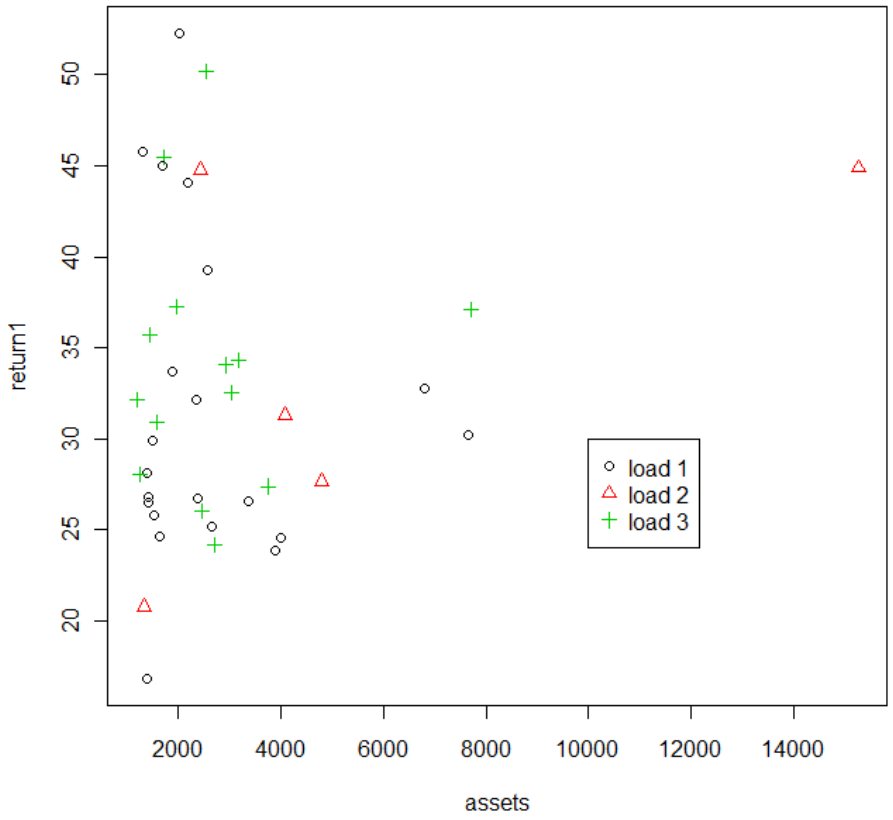
Scatter plots for subgroups formed by type

```
> par(mfrow = c(1, 1))
> plot(return1 ~ assets, data = mfund.data, pch = type, col = type)
> legend(10000, 30, legend = paste("type", c(1, 2, 4, 5)),
+ col = c(1, 2, 4, 5), pch = c(1, 2, 4, 5))
```



Scatter plots for subgroups formed by load

```
> par(mfrow = c(1, 1))
> plot(return1 ~ assets, data = mfund.data, pch = load, col = load)
> legend(10000, 30, legend = paste("load", 1:3), col = 1:3, pch = 1:3)
```



The commands “plot”, “lines”, “points”

- plot(x,y) gives a scatterplot if x is continuous,
  and a box-and-whisker plot if x is a factor.
- alternative syntax: plot(y~x) using 'tilde' as in a model fomula.
- types of plot: options include lines type="l" or null (axes only) type="n".

- lines(x,y) plots a smooth function of y against x using the x and y values provided. You might prefer lines(y~x).

- points(x,y) adds another set of data points to a plot.
  You might prefer points(y~x).

Data preparation, management

- Change the variable “type” to a factor variable:

```
> str(mfund.data)
'data.frame':   40 obs. of  7 variables:
 $ load   : int  2 3 1 1 2 2 1 1 3 1 ...
 $ exprat : num  1.06 0.37 0.55 0.77 0.65 0.7 0.75 0.6 1 0.5 ...
 $ type   : int  2 4 4 4 5 5 4 4 2 4 ...
 $ assets : num  15252 7709 7643 6792 4790 ...
 $ return5: num  117.4 62.8 92 81.5 62.4 ...
 $ return1: num  44.9 37.1 30.2 32.8 27.7 ...
 $ name   : Factor w/ 40 levels "Affiliated_Fund",..: 14 39 22 38 15 11 30 5 35 1 ...
```

```
str(mfund.data)
type_factor <- as.factor(type)
levels(type_factor) <-c("ca","go","gio","eio")
with(mfund.data,table(type))
mfund.data<-cbind(mfund.data,type_factor)
head(mfund.data)
```

```
> head(mfund.data)
  load exprat type assets return5 return1                      name
1    2   1.06    2  15252     117      45        Fidelity_Magellan
2    3   0.37    4   7709      63      37            Windsor_Funds
3    1   0.55    4   7643      92      30 Investment_Co_of_America
4    1   0.77    4   6792      81      33    Washington_Mutual_Inv
5    2   0.65
6    2   0.70
  type_factor
1          go
2         gio
3         gio
4         gio
5         eio
6         eio
```

```
> str(mfund.data)
'data.frame':   40 obs. of  8 variables:
 $ load       : int  2 3 1 1 2 2 1 1 3 1 ...
 $ exprat     : num  1.06 0.37 0.55 0.77 0.65 0.7 0.75 0.6 1 0.5 ...
 $ type       : int  2 4 4 4 5 5 4 4 2 4 ...
 $ assets     : num  15252 7709 7643 6792 4790 ...
 $ return5    : num  117.4 62.8 92 81.5 62.4 ...
 $ return1    : num  44.9 37.1 30.2 32.8 27.7 ...
 $ name       : Factor w/ 40 levels "Affiliated_Fund",..: 14 39 22 38 15 11 30 5 35 1 ...
 $ type_factor: Factor w/ 4 levels "ca","go","gio",..: 2 3 3 3 4 4 3 3 2 3 ...
```

Rownames and colnames

```
> colnames(mfund.data)
 [1] "unload"      "exprat"      "type"        "assets"
 [5] "return5"     "return1"     "name"        "type_factor"
 [9] "type_factor" "type_factor"
> colnames(mfund.data)[1]<-"unload"
> colnames(mfund.data)
 [1] "unload"      "exprat"      "type"        "assets"
 [5] "return5"     "return1"     "name"        "type_factor"
 [9] "type_factor" "type_factor"
```

```
> rownames(mfund.data)
 [1] "Trial.1" "2"       "3"       "4"       "5"       "6"
 [7] "7"       "8"       "9"       "10"      "11"      "12"
[13] "13"      "14"      "15"      "16"      "17"      "18"
[19] "19"      "20"      "21"      "22"      "23"      "24"
[25] "25"      "26"      "27"      "28"      "29"      "30"
[31] "31"      "32"      "33"      "34"      "35"      "36"
[37] "37"      "38"      "39"      "40"
> rownames(mfund.data)[1]<-"Trial.1"
> rownames(mfund.data)
 [1] "Trial.1" "2"       "3"       "4"       "5"       "6"
 [7] "7"       "8"       "9"       "10"      "11"      "12"
[13] "13"      "14"      "15"      "16"      "17"      "18"
[19] "19"      "20"      "21"      "22"      "23"      "24"
[25] "25"      "26"      "27"      "28"      "29"      "30"
[31] "31"      "32"      "33"      "34"      "35"      "36"
[37] "37"      "38"      "39"      "40"
```

Sorting, selecting, dropping data

```
#sorting your data
mfund.data[order(type),c(1:5)]
mfund.data[rev(order(type)),c(1:5)]
mfund.data[order(type,load),]
#rule: if in doubt, sort using more variables than you think you need
#put only some variables in the sorted dataframe:
#specify the column numbers in the sequence we want them to appear
mfund.data[order(type,load),c(3,1,2,4,5)]
#or more cumbersome:
mfund.data[order(type,load),c("type","load","exprat","assets")]
#using logical conditions to select rows
mfund.data[type==1,]
mfund.data[assets>median(assets)&type<4,]  #median(assets): 2372
mfund.data[,sapply(mfund.data,is.numeric)]
mfund.data[,sapply(mfund.data,is.factor)]
mfund.data[,sapply(mfund.data,is.character)]
#drop rows with negative subcripts
mfund.data[-(6:15),]
```

For more commands …

… see the R introduction, the literature, just google, …

Thank you for your attention.

Regine Gerike
Technische Universität München
mobil.TUM
Office: 1753
Tel +49.89.289.28575
regine.gerike@tum.de