# Statistical Programming with R

## Main Functions and Graphics for Exploratory analysis

Emmanuel Kemel (HEC Paris, CNRS)

## Notions seen in these slides

- Applying functions
- Making Graphics
- Analyzing variables and relations between variables

# Outline

# Function over a vector

Most functions apply to vectors, and return

- a single value
  min(), max(), mean(),median(), sum(), prod(), sd(), var()
- a vector of values
  quantile(), table()
- a vector of the same size as the argument
  exp(), log(), sqrt(), abs(), cumsum()
- one index
  which.min(), which.max()
- one or several indexes
  which(condition)
- a vector of indexes that has the same size as the argument
  order()
- a lists that contains the results of the computation
  t.test(), lm()

## Functions over two vectors

Functions can take several two or more vectors as argument and return

- a single value
  cor(x,y), min(x,y)
- a vector
  pmin(x,y)
- a matrix
  out(x,y) returns a matrix $m$ of size(length(x),length(y)) such that
  $m[i,j] = x[i] \times y[j]$
  cov(x,y)
- a list that contains the results of the computation
  t.test(x,y)

# Functions over a matrix

- Function **apply()** makes computations on the margins of a matrix.
- Its arguments are
  - the matrix
  - the dimension of the computation: 1 for lines, 2 for columns
  - the function to be applied

  Note that applying a function to a matrix without function apply will, if possible return a calculation over all the values of the matrix.

# Examples

```
setwd("~/Dropbox/PhD_Econometrics/data")
data=read.table("profsalary.txt", header=T)
apply(data,2,mean)
```

```
      Case     Salary Experience
  72.00000   65.16783   18.86014
```

```
apply(data,2,median)
```

```
      Case     Salary Experience
        72         69         18
```

```
apply(data,2, quantile)
```

```
      Case Salary Experience
0%     1.0   37.0        1.0
25%   36.5   60.5       12.5
50%   72.0   69.0       18.0
75%  107.5   72.0       26.0
100% 143.0   78.0       36.0
```

# Conditional applications of functions

- Function **tapply(x,y,FUN)** applies function FUN to vector x, conditionnaly on the values of y
  i.e. the function is applied for each value of y

```
setwd("~/Dropbox/Rprogramming/data")
data=read.table("cps08.csv",header=T, sep=";",dec=",")
tapply(data$ahe, data$female, mean)
```

```
       0        1
20.11387 17.48396
```

```
tapply(data$ahe, list(data$female,data$bachelor), mean)
```

```
         0        1
0 16.58962 24.97840
1 13.15342 20.87478
```

```
#with(data, tapply(ahe, list(female,bachelor), mean))
```

# Collapsing data with aggregate

Aggregate is a wrapper for tapply.

- it can be used like tapply (limited interest)
- it can be used with formula as an argument, and use the attributes of the data.frame.
- returns a data.frame

```
setwd("~/Dropbox/Rprogramming/data")
data=read.table("cps08.csv",header=T, sep=";",dec=",")
aggregate(ahe~female+bachelor, data, mean)
```

```
  female bachelor      ahe
1      0        0 16.58962
2      1        0 13.15342
3      0        1 24.97840
4      1        1 20.87478
```

# Another approach to data analysis

- Rstudio has developped a series of packages for data analysis
  - dplyr - Essential shortcuts for subsetting, summarizing, rearranging, and joining together data sets. dplyr is our go to package for fast data manipulation.
  - tidyr - Tools for changing the layout of your data sets. Use the gather and spread functions to convert your data into the tidy format, the layout R likes best.
- the packages have their own “synthax”
- See R for Data Science for a presentation: https://r4ds.had.co.nz/tidy-data.html

## Another approach to coding

- R commands can quickly become long and hard to read
- Example: mean salary for males and females for workers under 30

```
setwd("~/Dropbox/Rprogramming/data")
data=read.table("cps08.csv",header=T, sep=";",dec=",")
#aggregate(ahe~female, data[,data$age<30], mean)
#aggregate(ahe~female, data, mean,subset=c(age<30))
```

# Another approach to coding: the pipe operator

- The pipe operator %>% facilitates chains of operations (from the magrittr package)
  f(g(h(x))) would write x %>% h() %>% g() %>% f()
- Example 1

```
library(magrittr)
x=rnorm(10)
#sqrt(mean(exp(x)))
x %>% exp() %>% mean() %>% sqrt()
```

```
[1] 1.268682
```

- Example 2

```
setwd("~/Dropbox/Rprogramming/data")
data=read.table("cps08.csv",header=T, sep=";",dec=",")
data %>% subset(age<30) %>% aggregate(ahe~female,.,mean)
```

```
  female      ahe
1      0 18.00067
2      1 16.50339
```

# Outline

## Outline

## About graphics

During a statistical analysis, you generally need two types of graphics

- graphics for you
  that help you to understand your data
  and do not need to be particularly pretty
- graphics for others
  that illustrate key patterns in your data
  they must be optimized to convey a clear message

With R, getting a "raw" graphics of data is very easy/efficient
Very sophisticated graphics can also be produced, but their conception is (much) more time consumming. The use of templates is welcome.

## Graphics with R

- Unlike functions seen so far, it is not possible to assign a graphic to an object
- Graphics are assigned to graphical devices: a window, or a file.
- There are two kinds of functions:
  - primary (High level), that create a new graphical device
  - secondary (Low level), that add to an existing graphical device
- parameters of graphics can be set locally (when calling a function) or globally with function par()

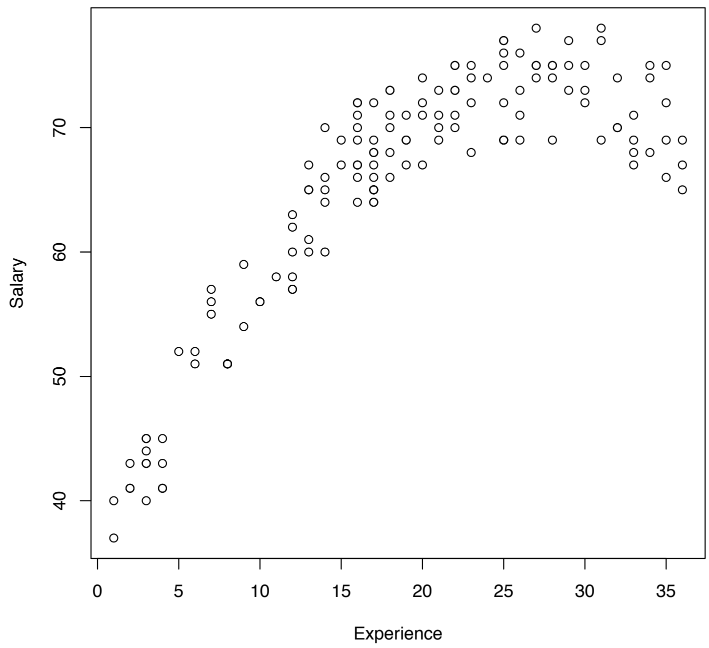
## The generic function plot()

plot() is a generic function that

- creates a graphic enviroment
- produces a graphic that is adapted to
  - the type of argument: vector, matrix, formula, list
  - the mode of the argument: vector or factor

# Example

```
data=read.table("profsalary.txt", header=T)
plot(Salary~Experience, data=data)
```



# Other functions are specific

The is a long list of specific plot functions, the most current are:

- boxplot()
- barplot()
- pie()
- interaction.plot()
- mosaicplot()

## Secondary (low level) plots

- These function do not open a graphical device, but
- Add something to an existing device
  - points()
  - lines() or abline()
  - text()
  - legend()
  - arrows()

...

# Outline

# Main local options

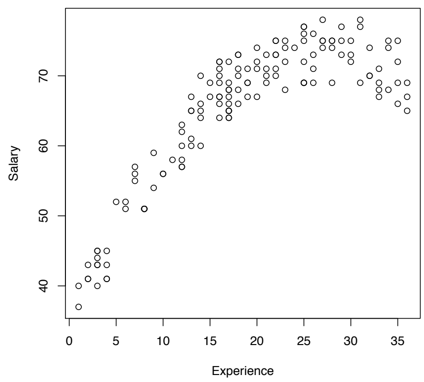
Main options are:

- main
- xlim, ylim
- xlab, ylab
- point types: pch
- lines: lty, lwd
- cex
- col or bg

# Example: customizing the plot of Prof's Salary

- We start from the simple graph

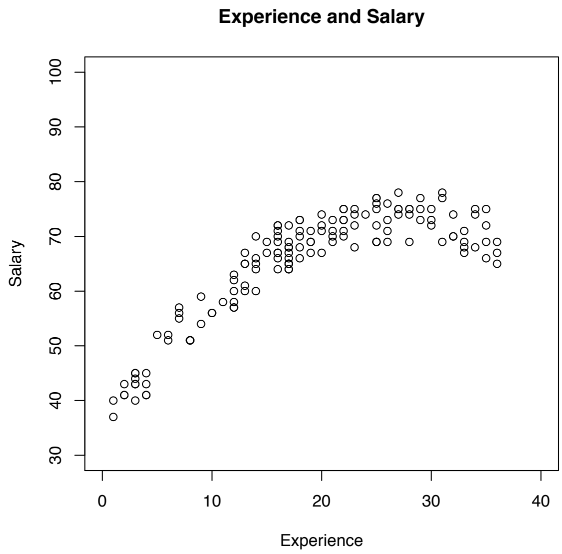
```
data=read.table("profsalary.txt", header=T)
plot(Salary~Experience, data=data)
```



# customizing the plot of Prof's Salary

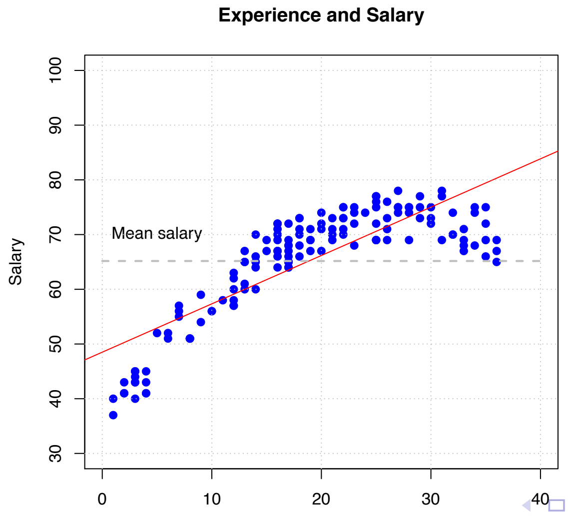
- adding title, labels, changing the scale

```
data=read.table("profsalary.txt", header=T)
plot(Salary~Experience, data=data, xlim=c(0,40), ylim=c(30,100),mai
```



# customizing the plot of Prof's Salary

```
data=read.table("profsalary.txt", header=T)
plot(Salary~Experience, data=data, xlim=c(0,40),
ylim=c(30,100),main="Experience and Salary",
pch=19, col="blue")
abline(lm(Salary~Experience, data=data),col="red")
lines(c(0,40),rep(mean(data$Salary),2),lty=2,col="grey", lwd=2)
text(5,70,"Mean salary")
grid(col="grey")
```



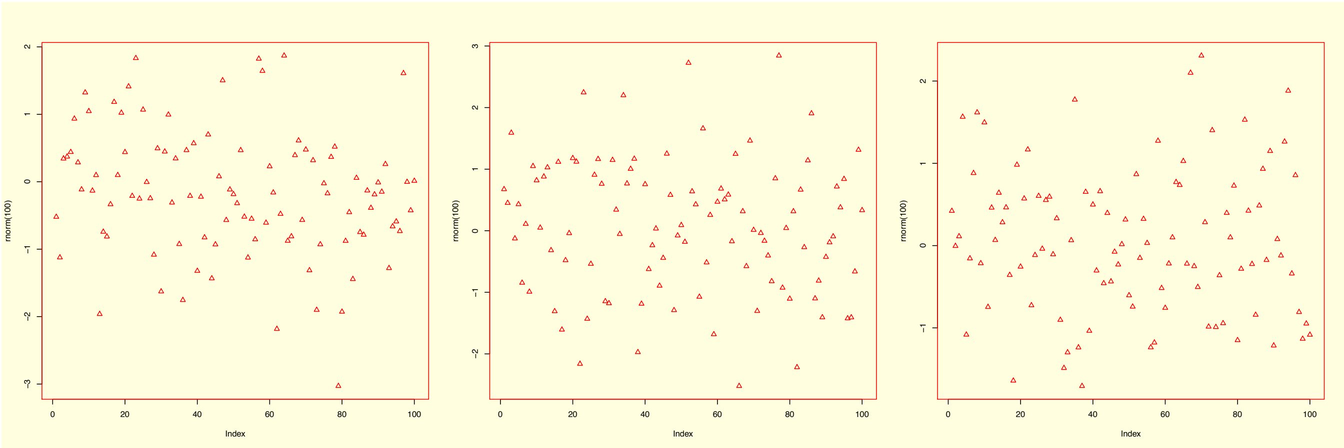
# Main global options

Options can be set globally with function **par()**
Example:

- par(bg = "yellow")
- par(mfrow=c(1,2))

# Example

```
par(mfrow=c(1,3),bg="lightyellow", col="red", pch=2) plot(rnorm(100
plot(rnorm(100)) plot(rnorm(100))
```



# Graphical device(s)

- For including the graph in a file, functions such as **postscript()**, **pdf()** or **png()** open a graphic file
- Several devices can be open at the same time.
  function **dev.list()** lists the currently open devices
- a device is closed with function dev.off()

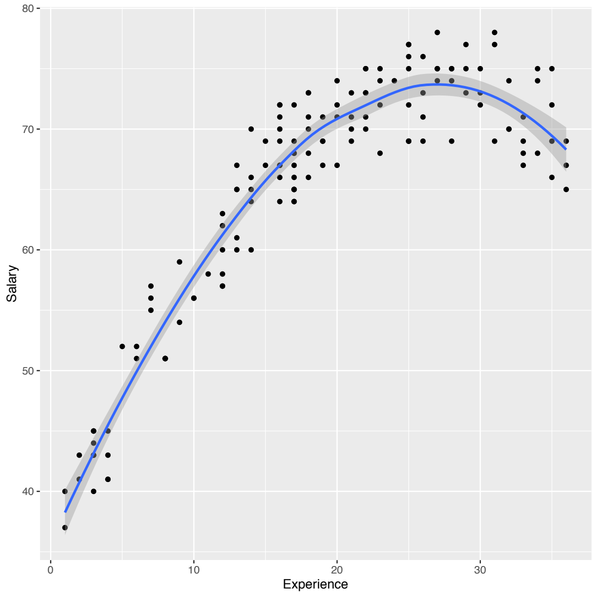
## Another approach to graphics: ggplot2

- Rstudio has developped a package for graphics, with its own grammar...
- See https://ggplot2-book.org/
- Principle:
  - the functions builts on aesthetics that define the frame, variables, colors and shapes
  - then functions plot content following the structure of the aesthetics

# ggplot2: example

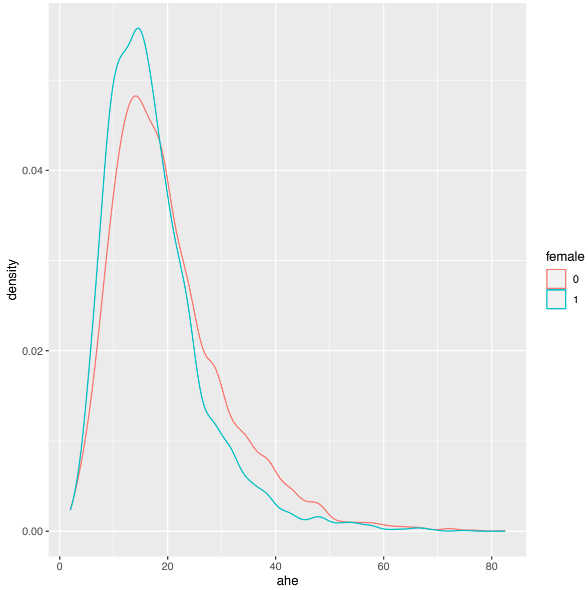
```
setwd("~/Dropbox/Rprogramming/data")
data=read.table("profsalary.txt", header=T)
library(ggplot2)
ggplot(data,aes(x=Experience,y=Salary))+
geom_point()+
geom_smooth()
```

*‘geom_smooth()‘ using method = ’loess’ and formula ’y ~ x’*



# ggplot2: example 2

```
setwd("~/Dropbox/Rprogramming/data")
data=read.table("CPS08.csv", header=T,sep=";",dec=",")
data$female=factor(data$female)
ggplot(data,aes(x=ahe,color=female))+
geom_density()
```



## Outline

## Outline

# Frequencies: computation

- Absolute frequencies (ie. counts) are given by function **table()**
- Relative frequencies (i.e. proportions) are given by **prop.table()** that takes a table as argument

# Frequencies

```
data=read.table("cps08.csv",header=T, sep=";",dec=",")
table(data$female)
```

```

   0    1
4375 3336
```

```
prop.table(table(data$female))
```

```

        0         1
0.5673713 0.4326287
```

```
data$female=factor(data$female, labels=c("M","F"))
table(data$female)
```

```

   M    F
4375 3336
```

```
prop.table(table(data$female))
```

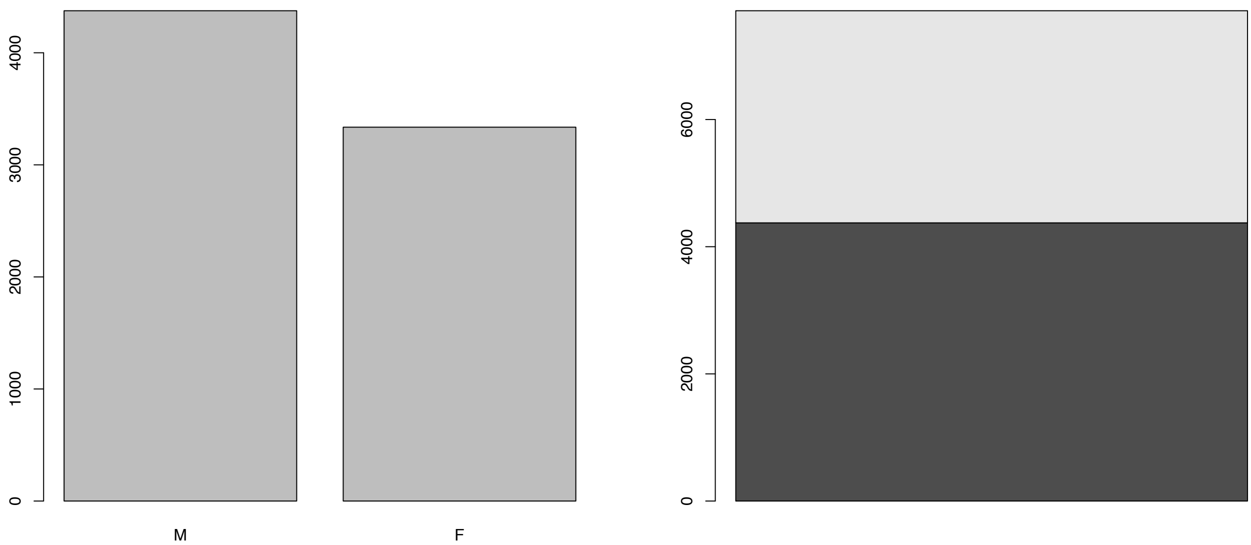
```

        M         F
0.5673713 0.4326287
```

# Frequencies: representation

- discrete variables should be represented by a barplot, that can be produced with function... **barplot()**
- **barplot()** takes as argument a table or a matrix, whose row elements will be stacked.

# Example of representation

```
barplot(table(data$female))
barplot(matrix(table(data$female),ncol=1))
```



## Usual test(s)

- When dealing with a variable possibly taking 2 values, a standard test is the binomial test
  Example:

```
binom.test(table(data$female))


Exact binomial test

data:  table(data$female)
number of successes = 4375, number of trials = 7711,
p-value < 2.2e-16
alternative hypothesis: true probability of success is not equal
95 percent confidence interval:
 0.5562245 0.5784671
sample estimates:
probability of success
             0.5673713
```

## Usual test(s)

- When the variable takes more than two values, a $\chi^2$ test can be run

```
chisq.test(table(data$female))


Chi-squared test for given probabilities

data:  table(data$female)
X-squared = 140, df = 1, p-value < 2.2e-16
```

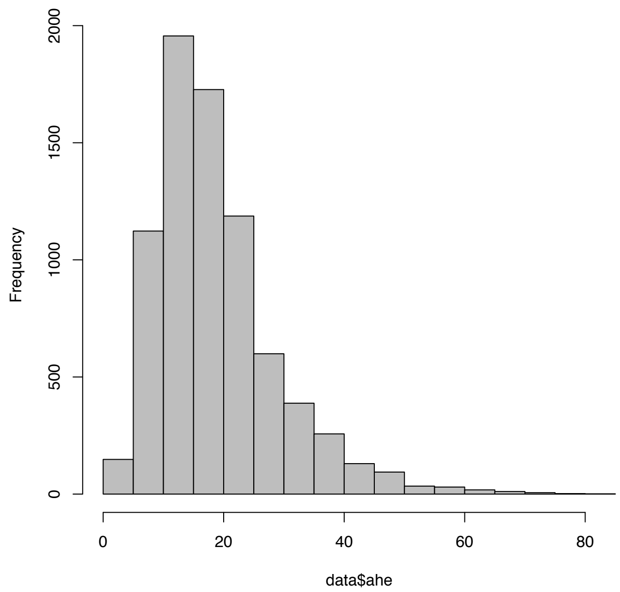
# Outline

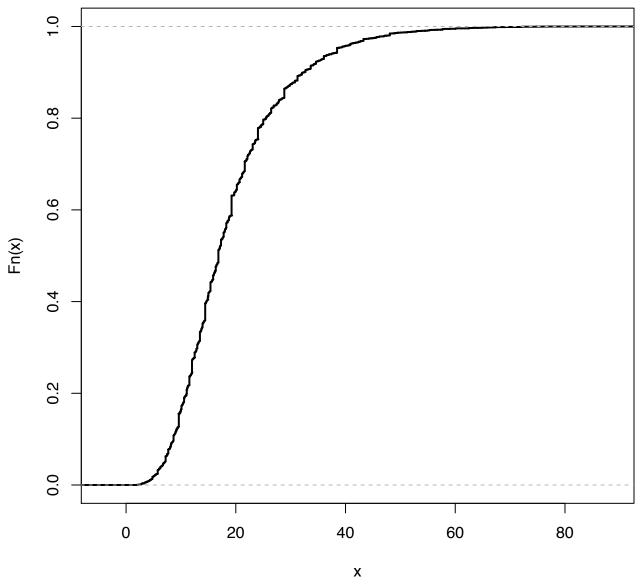
## Frequencies and histogram

- **table()** and **prob.table()** can be used to observe the empirical frequencies of a continuous variable
- Frequencies can be represented by an histogram **hist()** or by the cumulative empirical function, given by **ecdf()**
- Other usual statistics include **mean()**, **median()**, **var()**, **sd()**, **min()**, **max()**, **quantile()**, **kurtosis()**

# Illustration

```
data=read.table("cps08.csv", header=T, sep=";", dec=",")
hist(data$ahe, main="", col="grey")
plot(ecdf(data$ahe),verticals=T, lwd=2, main="")
```

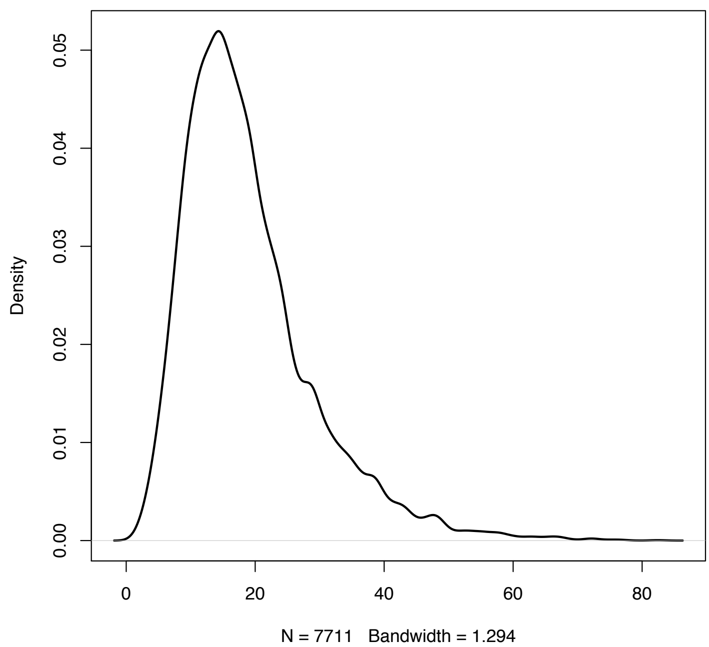


# Illustration: density

- the density computes a smoother for the histogram

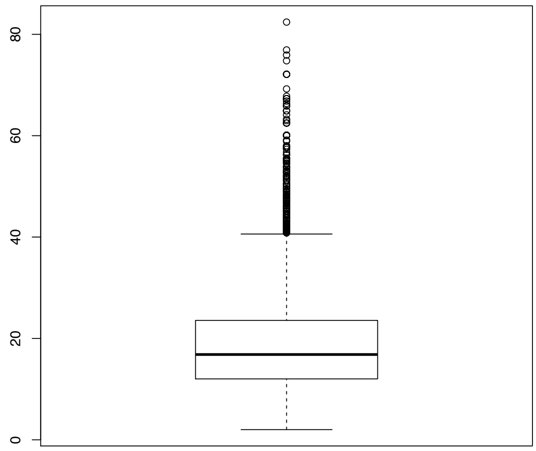
```
plot(density(data$ahe), lwd=2, main="")
```



## Boxplot

- Another usual way to represent univariate variables is the boxplot, with function .... **boxplot()**

```
boxplot(data$ahe, main="")
```

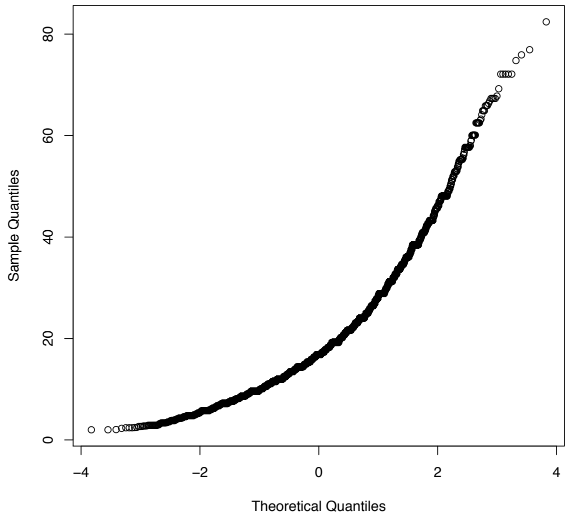



- The boxplot represents quartiles and outliers that deviate 1.5 IQR from Q1 or Q3.

# QQ plots

- **qqnorm()** plots the quantiles of the distributions against the quantiles of a normal distribution.

```
qqnorm(data$ahe, main="")
```



# Usual test(s)

- t.test() compares the mean of the variable to a given variable

```
t.test(data$ahe, mu=20)

One Sample t-test

data:  data$ahe
t = -8.8675, df = 7710, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 20
95 percent confidence interval:
 18.74975 19.20244
sample estimates:
mean of x
 18.97609
#t.test(data$ahe, mu=20, alternative="greater")
```

# Usual test(s) (continued)

- Wilcoxon rank test (a non parametric test, less powerful but robust to outliers)

```
wilcox.test(data$ahe, mu=20)


Wilcoxon signed rank test with continuity correction


data:  data$ahe
V = 11035832, p-value < 2.2e-16
alternative hypothesis: true location is not equal to 20
#wilcox.test(data$ahe, mu=20, alternative="greater")
```

# Outline

# Outline

## Joint and conditional frequencies

- Consider $X$ and $Y$ two random variables that each take a finite number of values,
  we generally want to study their joint or conditional probability distributions
- With R, each variable will be a factor.
- The joint absolute frequencies will be given by function **table()**, that creates a contingency table
- The relative frequencies will be given by function **prop.table()**, that applies to a table
- Conditional frequencies are also given by **prop.table()** when indicating the dimension to fix
  - 1 for fixing the column variable
  - 2 for fixing the row variable

  Note that these functions can be applied to more than two variables.

# Example of contingency table and joint frequencies

```
data(UCBAdmissions)
data=UCBAdmissions[,,1]
print(data)
```

```
          Gender
Admit      Male Female
  Admitted  512     89
  Rejected  313     19
```

```
prop.table(data)
```

```
          Gender
Admit            Male     Female
  Admitted 0.54876742 0.09539121
  Rejected 0.33547696 0.02036442
```

# Examples of conditional frequencies

- Conditional frequencies of variable admission, given gender

```
prop.table(data,2)
          Gender
Admit           Male    Female
  Admitted 0.6206061 0.8240741
  Rejected 0.3793939 0.1759259
```

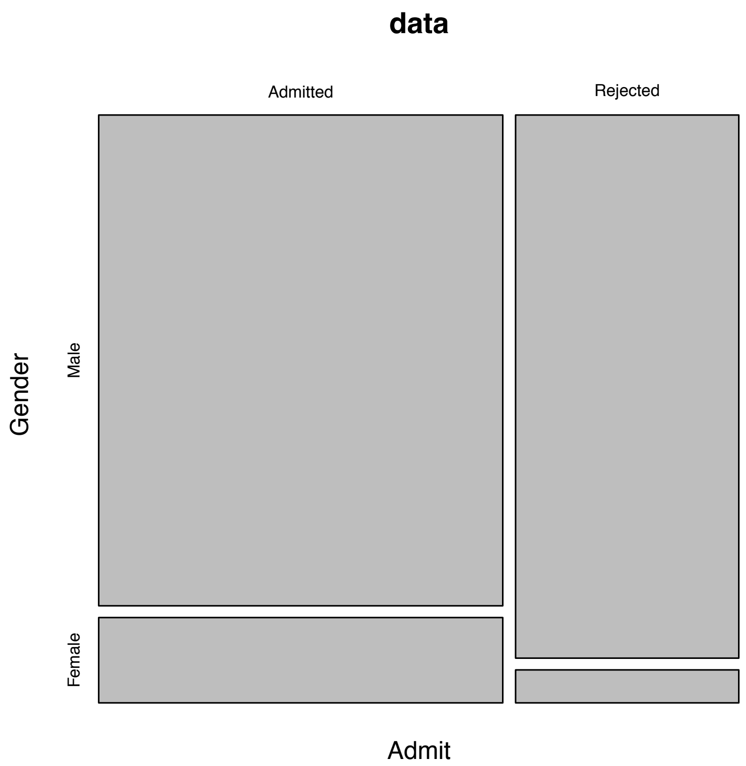
- Conditional frequencies of variable gender given admission

```
prop.table(data,1)
          Gender
Admit            Male      Female
  Admitted 0.85191348 0.14808652
  Rejected 0.94277108 0.05722892
```

# Graphical representation

- Joint frequencies can be represented with a mosaic plot

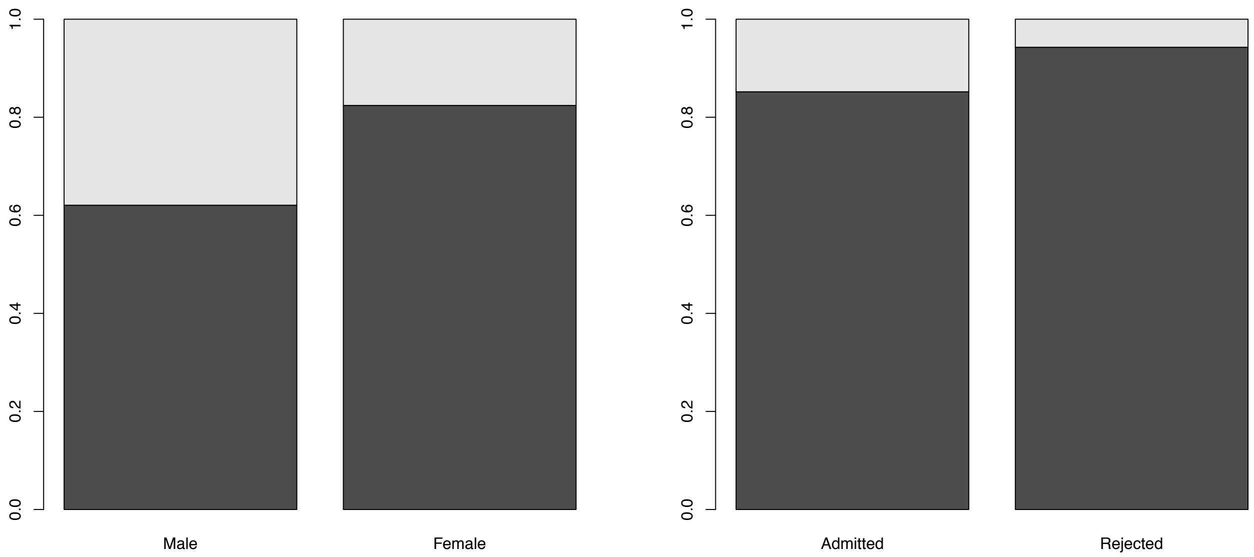
```r
plot(data)
```



# Graphical representation

- Conditional frequencies

```
barplot(prop.table(data,2))
barplot(t(prop.table(data,1)))
```



# Classical tests

Usual tests for contingencies tables are

- the Fisher exact test

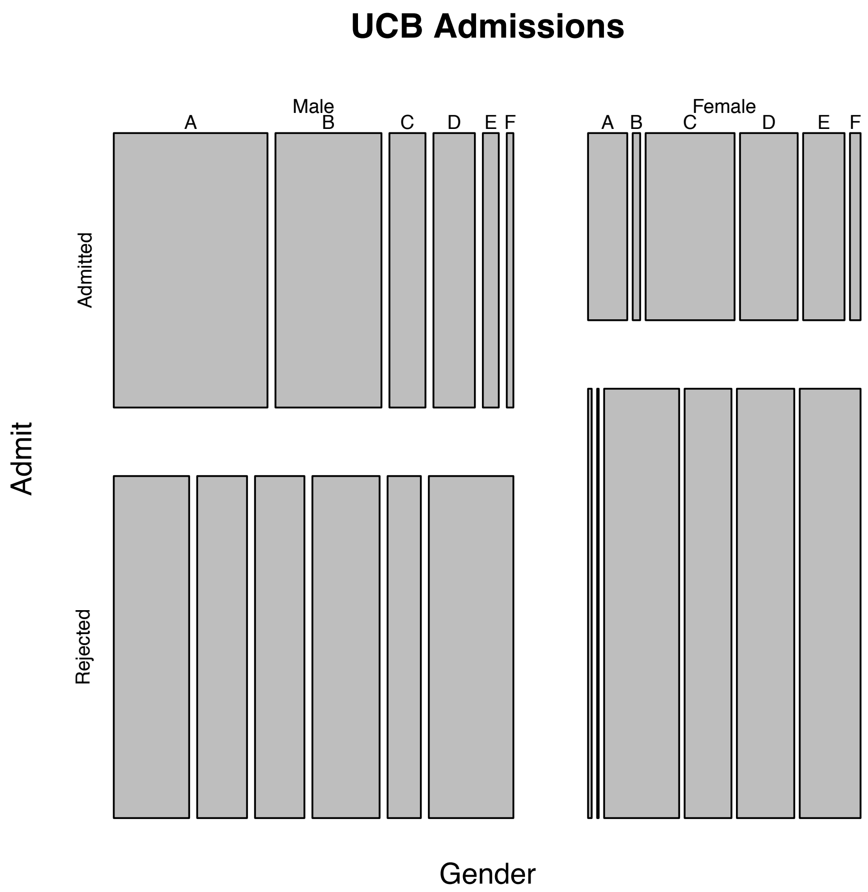
```
fisher.test(data)


Fisher's Exact Test for Count Data


data:  data
p-value = 1.669e-05
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.1970420 0.5920417
sample estimates:
odds ratio
 0.3495628
```

# Classical tests (continued)

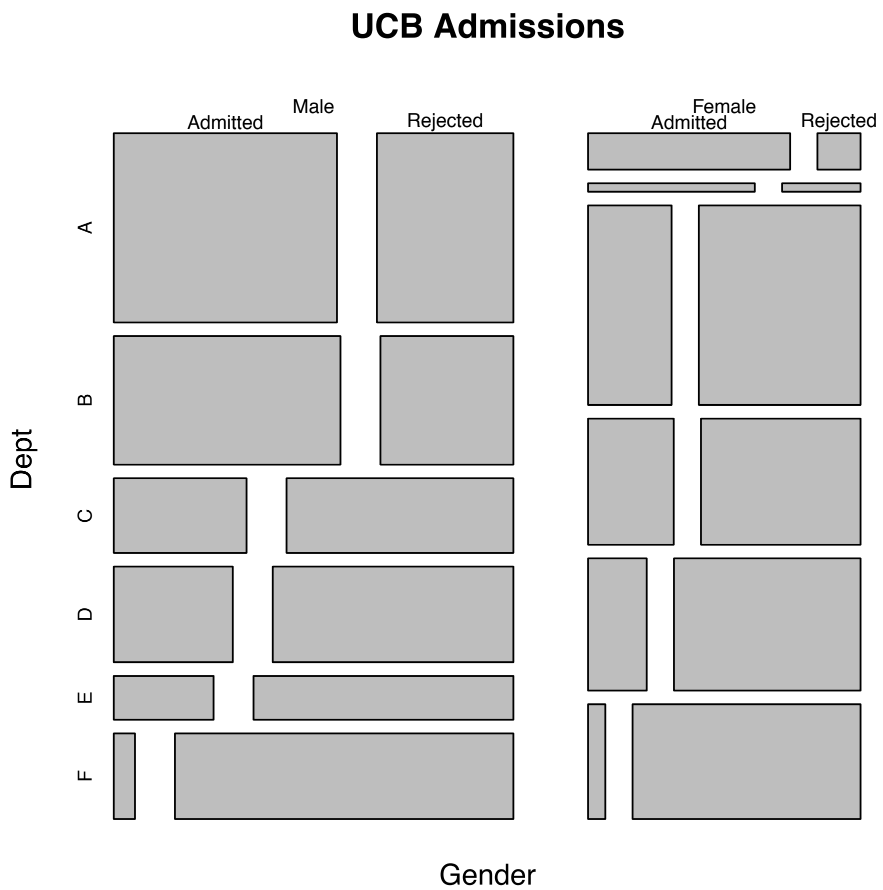
- the $\chi^2$ test

```
chisq.test(data)


Pearson's Chi-squared test with Yates' continuity
correction

data:  data
X-squared = 16.372, df = 1, p-value = 5.205e-05
```

# Illustration of Simpson's paradox: the "illusion"



# Illustration of Simpson's paradox: the explanation



# Outline

## The statistics

Consider $Y$ and $X$ two random variables, one continuous and the second discrete

In order to assess the dependence of the two variables we can compare:

- the distributions of $Y$ conditionally on the (finite number of) values of $X$
- more specifically, the conditional mean of $Y$ conditionally on the (finite number of) values of $X$

# Conditional distributions

The most flexible function for computing any statistic conditionally on any other variable(s) is **aggregate()**

- one or several variables can be treated
- conditions can rely on one of several variables
- one or several statistics can be computed

## Simple example

```
data=read.table("cps08.csv", header=T, sep=";", dec=",")
aggregate(ahe~female, data=data, mean)
  female      ahe
1      0 20.11387
2      1 17.48396
aggregate(ahe~female+bachelor, data=data, mean)
  female bachelor      ahe
1      0        0 16.58962
2      1        0 13.15342
3      0        1 24.97840
4      1        1 20.87478
```

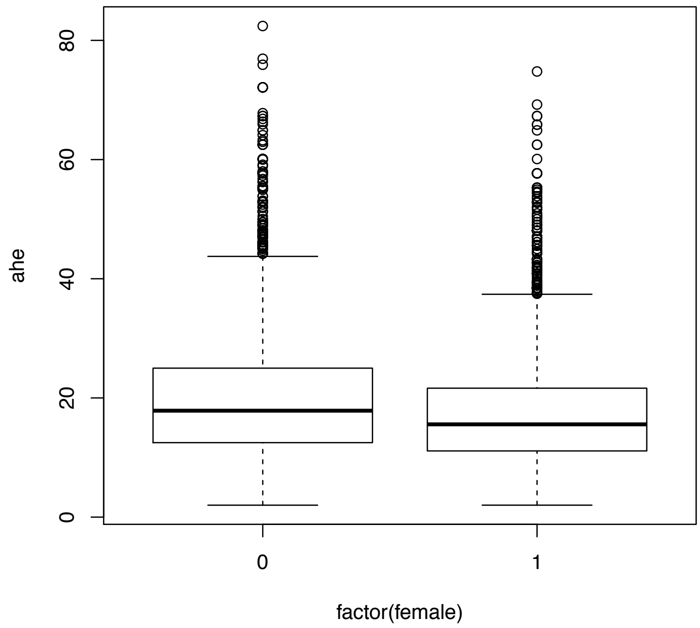
## Simple example continued

```
aggregate(ahe~female+bachelor, data=data,
function(x) c(mean=mean(x),med=median(x),sd=sd(x)))
```

```
  female bachelor  ahe.mean   ahe.med    ahe.sd
1      0        0 16.589619 14.903846  8.157563
2      1        0 13.153424 12.019231  6.270027
3      0        1 24.978404 23.076923 11.778632
4      1        1 20.874779 19.230770  9.657140
```

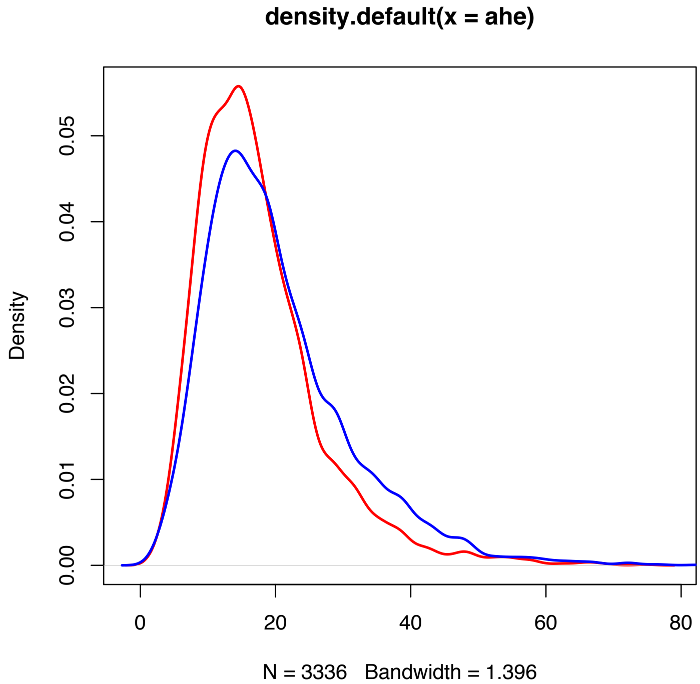
# Graphing conditional distributions

```
plot(ahe~factor(female), data=data)
```



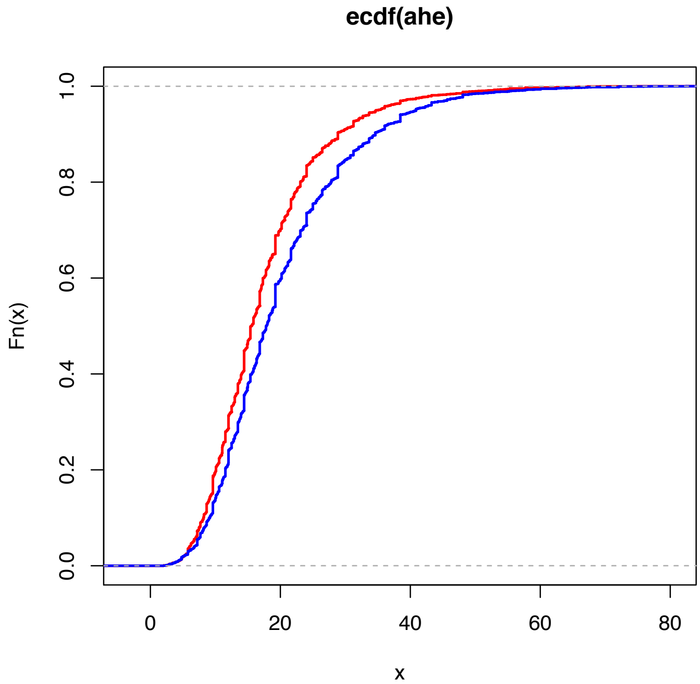
# Graphing conditional distributions: densities

```
with(subset(data,female==1),
plot(density(ahe),col="red",lwd=2))
with(subset(data,female==0),
lines(density(ahe),col="blue",lwd=2))
```



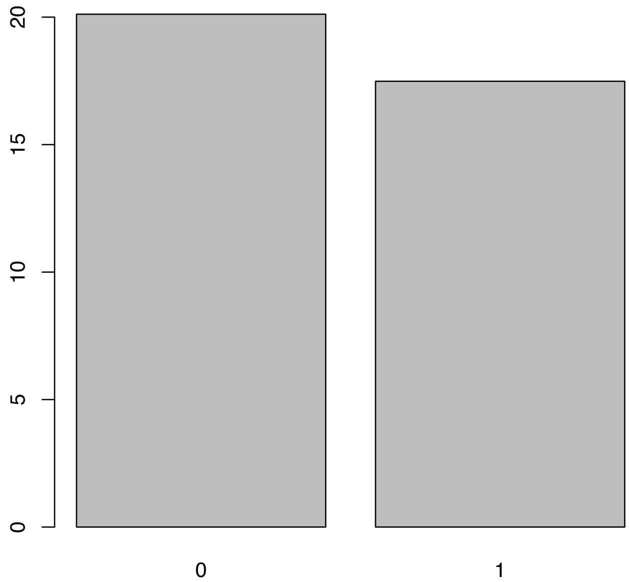
## Graphing conditional distributions

```
with(subset(data,female==1),
plot(ecdf(ahe),do.points=F, verticals=T,col="red",lwd=2))
with(subset(data,female==0),
lines(ecdf(ahe),do.points=F, verticals=T,col="blue",lwd=2))
```



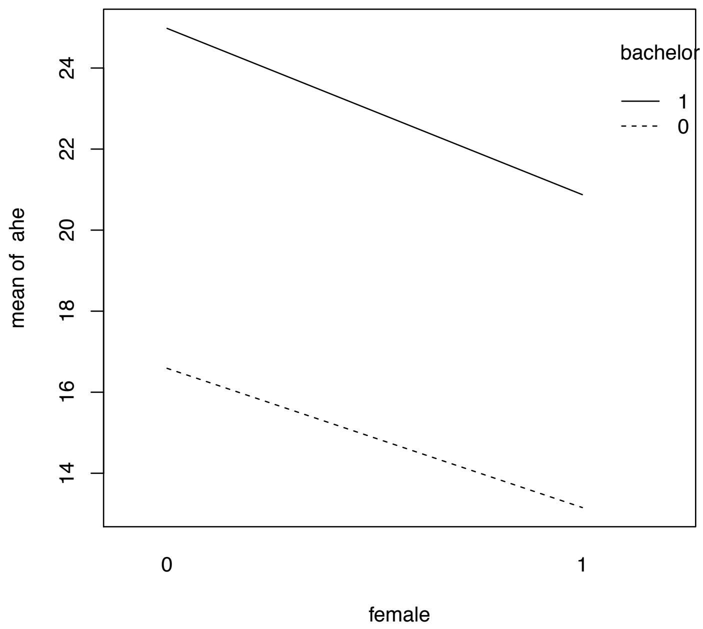
# Graphics conditional means

```
tab=aggregate(ahe~female,data=data, mean)
with(tab,barplot(ahe, names.arg=female))
```



# Graphics conditional means

```
tab=aggregate(ahe~female,data=data, mean)
with(data,interaction.plot(female, bachelor, ahe, mean))
```



# Outline

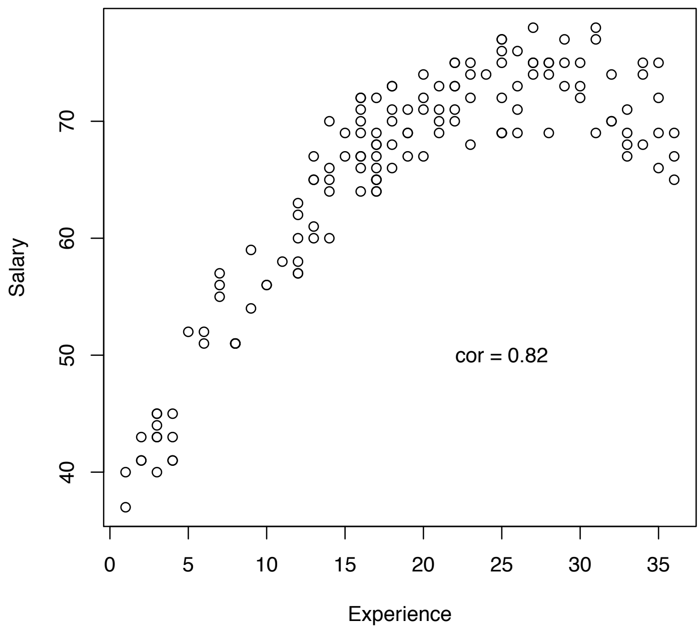
## Conditional distributions

Consider $Y$ and $X$ two continuous random variables,
In order to assess the dependence of the two variables we can:

- compute their correlation, but that will only assess the linear dependence
- compare the conditional of one variable on specific values or ranges of the other variable. (cf previous subsection)

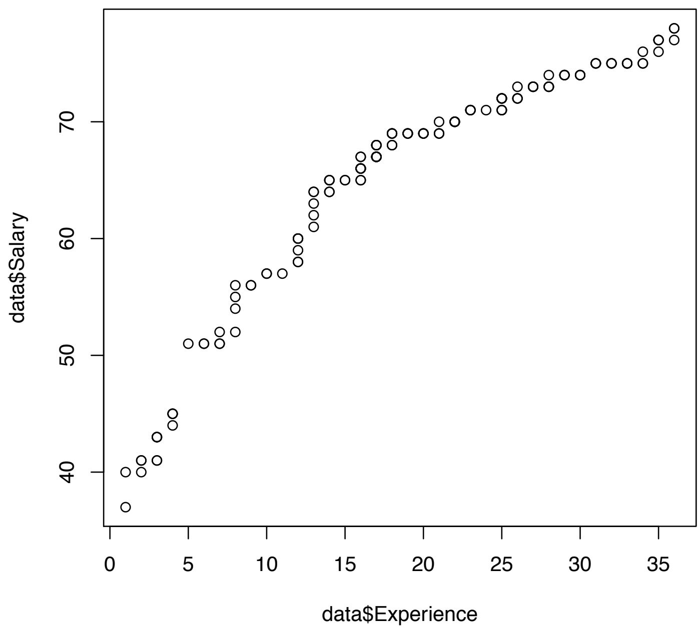
# Examples

```
data=read.table("profsalary.txt", header=T)
rho=cor(data$Salary, data$Experience)
plot(Salary~Experience, data=data)
text(25,50,paste("cor =",round(rho,2)))
```



## Examples

```
data=read.table("profsalary.txt", header=T)
qqplot( data$Experience, data$Salary)
```



# Outline

## Variance-covariance matrix

- Considering $k$ continuous random variables,
- The variance covariance matrix is given by **cov()**
- The matrix of correlations is given by **cor()**

# Multiple plots

```
library(foreign)
data=read.dta("journals.dta")
plot(data[,4:8])
```