HOSA Workshop on R

Dr. Wolfgang Rolke

April 20, 2021

Why R?

- software of choice for professional Statisticians
- Just about every statistical method ever invented is available in R
- any method invented recently is likely available only in R
- R is used more and more by non-statisticians. In many fields it has replaced other software like SPSS, SAS and Minitab.
- The main strength of R likely is its users. They form a very large community of people who contribute actively to the development of R.
- What can you do with R? Actually, what can't you do? For example, this workshop was written entirely in R!

Setting things up

Installation of base R

You can get a free version of R:

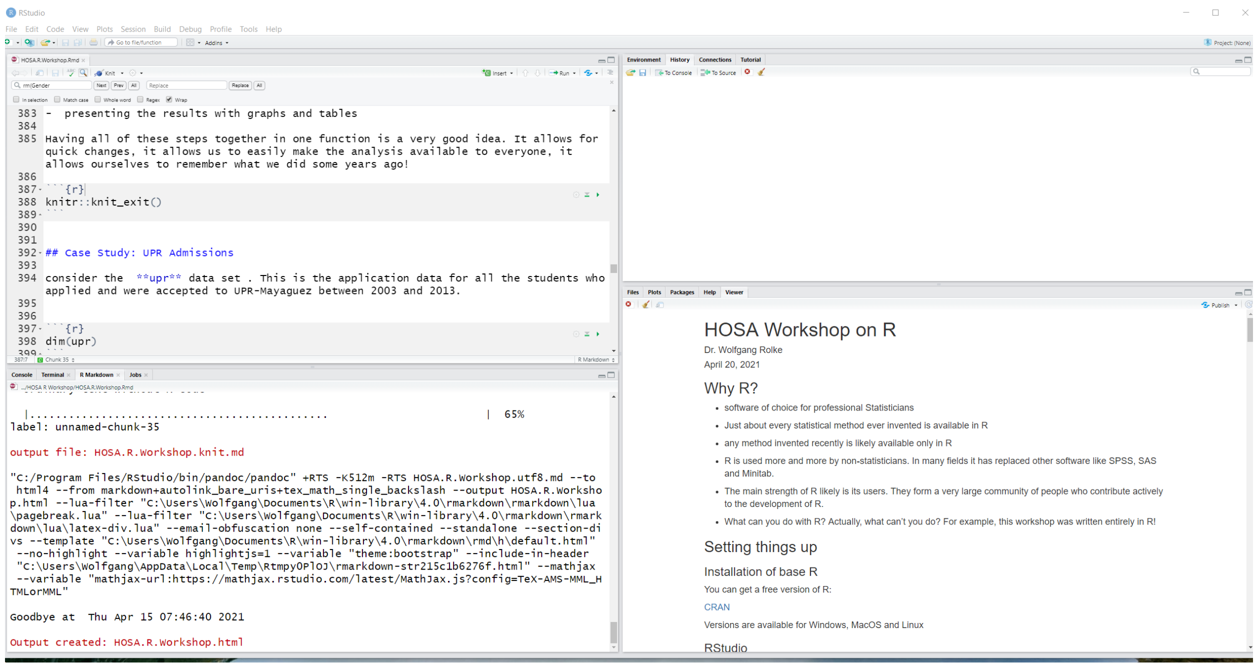
CRAN

Versions are available for Windows, MacOS and Linux

RStudio

Most people use R via the front-end **RStudio*. You can download it at RStudio. Again it is free.

Here is what it looks like:



~~~
-  presenting the results with graphs and tables

Having all of these steps together in one function is a very good idea. It allows for
quick changes, it allows us to easily make the analysis available to everyone, it
allows ourselves to remember what we did some years ago!

```{r}
knitr::knit_exit()
~~~

Case Study: UPR Admissions

consider the **upr** data set . This is the application data for all the students who
applied and were accepted to UPR-Mayaguez between 2003 and 2013.

```{r}
dim(upr)
```

```
…/HOSA R Workshop/HOSA.R.Workshop.Rmd

  |..............................................                       |  65%
label: unnamed-chunk-35

output file: HOSA.R.Workshop.knit.md

"C:/Program Files/RStudio/bin/pandoc/pandoc" +RTS -K512m -RTS HOSA.R.Workshop.utf8.md --to
 html4 --from markdown+autolink_bare_uris+tex_math_single_backslash --output HOSA.R.Worksho
p.html --lua-filter "C:\Users\Wolfgang\Documents\R\win-library\4.0\rmarkdown\rmarkdown\lua
\pagebreak.lua" --lua-filter "C:\Users\Wolfgang\Documents\R\win-library\4.0\rmarkdown\rmark
down\lua\latex-div.lua" --email-obfuscation none --self-contained --standalone --section-di
vs --template "C:\Users\Wolfgang\Documents\R\win-library\4.0\rmarkdown\rmd\h\default.html"
 --no-highlight --variable highlightjs=1 --variable "theme:bootstrap" --include-in-header
 "C:\Users\Wolfgang\AppData\Local\Temp\RtmpyOPlOJ\rmarkdown-str215c1b6276f.html" --mathjax
 --variable "mathjax-url:https://mathjax.rstudio.com/latest/MathJax.js?config=TeX-AMS-MML_H
TMLorMML"

Goodbye at  Thu Apr 15 07:46:40 2021

Output created: HOSA.R.Workshop.html
```

HOSA Workshop on R

Dr. Wolfgang Rolke

April 20, 2021

Why R?

- software of choice for professional Statisticians
- Just about every statistical method ever invented is available in R
- any method invented recently is likely available only in R
- R is used more and more by non-statisticians. In many fields it has replaced other software like SPSS, SAS and Minitab.
- The main strength of R likely is its users. They form a very large community of people who contribute actively to the development of R.
- What can you do with R? Actually, what can't you do? For example, this workshop was written entirely in R!

Setting things up

Installation of base R

You can get a free version of R:

CRAN

Versions are available for Windows, MacOS and Linux

RStudio

Packages / Libraries

- one of the great strength of R
- self contained sets of routines and data that somebody wrote to help with a specific task
- you have an analysis problem? google it and almost always you will find that someone has already done it for you

The main repository for R packages is again CRAN. Currently there are over 17500! (One of them was written by me).

To install a package (here a package called ggplot2) click on the Packages tab in the upper right portion of RStudio.

Once a package is installed you can load it into R with

```r
library("ggplot2")
```

Updating

- new versions of R are released about every three months
- Usually these are minor updates and it is not necessary to update your version every time
- When you do want to update simply download the latest version and install it on top of the old one

- After updating R also update your packages with

```
update_packages(dependencies = TRUE)
```

Further Reading

- For a more extensive introduction to R go here
- I teach a one semester graduate level course on R. For details go to Computing with R
- There are literally 100s of books dedicated to R. A short list of those that I have found useful is
  - Learning R
  - Introduction to Data Science with R
  - R Cookbook
  - Advanced R

R Studio Projects

Rstudio organizes everything in terms of *projects*, so if you should start by creating a new one each time you start a (well..), project. A nice feature is that a project can be linked to a GITHUB repository, which is useful for automatic backup and collaboration.

Once you have started a session the first thing you see is some text, and then the > sign in the Console pane. This is the **R prompt**, it means R is waiting for you to do something.

Let's start with

```
ls()
```

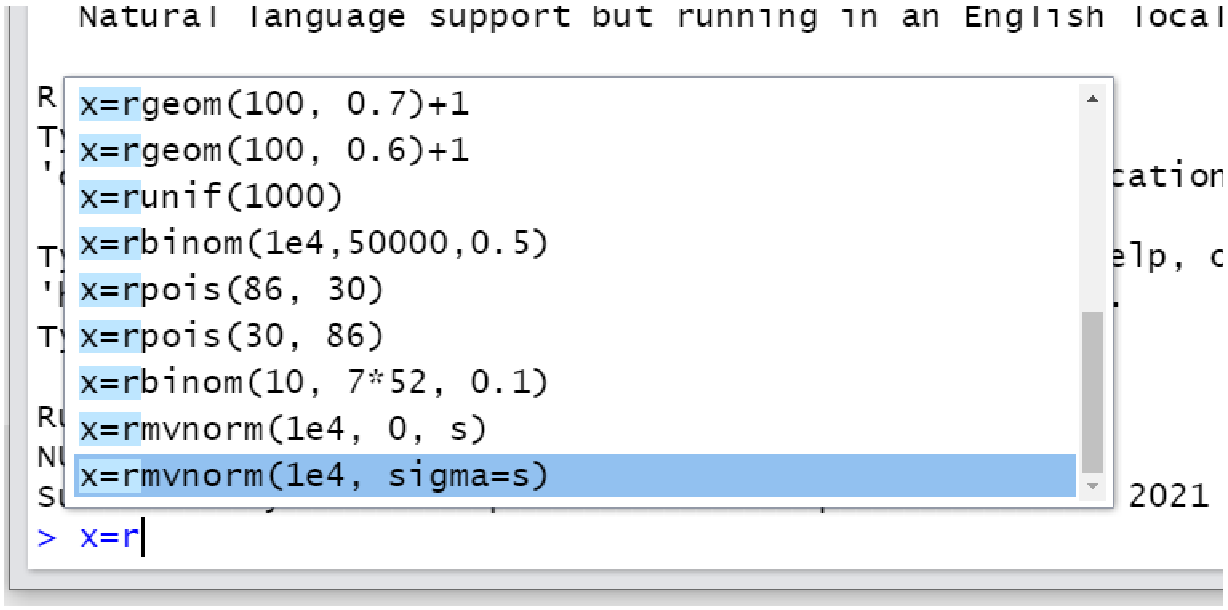
shows you a "listing"" of the files (data, routines etc.) Of course in the beginning there isn't anything there.

Everything in R is either a **data set** or a **function**. It is a function if it is supposed to do something (maybe calculate something, show you something like a graph or do something else). If it is a function it ALWAYS NEEDS (). Sometimes there is something (an *argument*) in between the parentheses. Sometimes there isn't like in the ls(). But the () has to be there anyway.

R has a nice recall feature, using the up and down arrow keys. Also, typing

```
history()
```

shows you the most recent things entered. Finally in RStudio you can type the first three letter of a command and then CTRL-↑ to see a list of when you used commands that start with these three letters



```
  Natural language support but running in an English local

x=rgeom(100, 0.7)+1
x=rgeom(100, 0.6)+1
x=runif(1000)
x=rbinom(1e4,50000,0.5)
x=rpois(86, 30)
x=rpois(30, 86)
x=rbinom(10, 7*52, 0.1)
x=rmvnorm(1e4, 0, s)
x=rmvnorm(1e4, sigma=s)
> x=r
```

RStudio has a lot of useful keyboard shortcuts, see which at Tools - Keyboard Shortcuts Help.

R is case-sensitive, so a and A are two different things.

Often during a session you create objects that you need only for a short time. When you no longer need them use **rm** to get rid of them:

```
x=10
x^2
```

```
## [1] 100
```

```
rm(x)
```

Sometimes instead of = people use = as the *assignment* character in R, the two do the same thing.

Data in R

Data Entry

- With the keyboard

For a few numbers the easiest thing is to just type them in:

```
x =  c(10, 2, 6, 9)
x
```

```
## [1] 10  2  6  9
```

c() is a function that takes the objects inside the () and **c**ombines them into one single object (a vector).

Useful commands for creating vectors are:

```
1:5
```

```
## [1] 1 2 3 4 5
```

```
rep(1, 5)
```

```
## [1] 1 1 1 1 1
```

```
rep(1:5, 3)
```

```
##  [1] 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5
```

```
rep(1:5, each=3)
```

```
##  [1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5
```

```
seq(0, 10, length=101)
```

```
##   [1]  0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0  1.1  1.2  1.3  1.4
##  [16]  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2  2.3  2.4  2.5  2.6  2.7  2.8  2.9
##  [31]  3.0  3.1  3.2  3.3  3.4  3.5  3.6  3.7  3.8  3.9  4.0  4.1  4.2  4.3  4.4
##  [46]  4.5  4.6  4.7  4.8  4.9  5.0  5.1  5.2  5.3  5.4  5.5  5.6  5.7  5.8  5.9
##  [61]  6.0  6.1  6.2  6.3  6.4  6.5  6.6  6.7  6.8  6.9  7.0  7.1  7.2  7.3  7.4
##  [76]  7.5  7.6  7.7  7.8  7.9  8.0  8.1  8.2  8.3  8.4  8.5  8.6  8.7  8.8  8.9
##  [91]  9.0  9.1  9.2  9.3  9.4  9.5  9.6  9.7  9.8  9.9 10.0
```

and these can be combined:

```
c(rep(1, 3), rep(2, 5), rep(3, 2))
```

```
##  [1] 1 1 1 2 2 2 2 2 3 3
```

If it is not numbers it needs quotes:

```r
c("Adam", "Karla", "Linda", "@")
```

```
## [1] "Adam"  "Karla" "Linda" "@"
```

- data from a file

Say you have a text file with numbers in it called *mydata1.txt* in a folder called *c:/myexamples*. You can read it into R with

```r
dta = scan("c:/myexamples/mydata1.txt")
```

if the data is not numbers but characters you need to use

```r
dta = scan("c:/myexamples/mydata1.txt", what="char")
```

R assumes the numbers and/or characters are separated by an empty space. If instead they are separated by (say) a ; use

```r
dta = scan("c:/myexamples/mydata1.txt", sep=";")
```

Often your data is in the form of a table, with different variables in different columns. Then you can use

```r
dta = read.table("c:/myexamples/mydata1.txt")
```

Very popular is data that comes as an Excel worksheet. In that case you can save it as a *comma delimited file* and read it into R with

```r
dta = read.csv("c:/myexamples/mydata1.csv")
```

There are a number of packages designed to help with data input/output. A very good one is called *rio*.

Data Types

R has the following basic data types:

- numeric
- character
- logical (TRUE/FALSE)

- factor

the last one is specific to Statistics. It is data with relatively few values that repeat many times. Examples are things like gender and treatment labels.

Data Formats

- Vectors

the most basic type of data in R is a **vector**, simply a list of values. However they all have to be of the same data type.

Say we want the numbers 1.5, 3.6, 5.1 and 4.0 in an R vector called x, then we can type

```
c(1.5, 3.6, 5.1, 4.0)
```

```
## [1] 1.5 3.6 5.1 4.0
```

```
c(1.5, 3.6, 5.1, 4.0, "A")
```

```
## [1] "1.5" "3.6" "5.1" "4"   "A"
```

Notice how this last one has all quotes, even so I didn't write them first. This is because if one element in a vector is not a number, none of them can be and R turns everything into a character.

- Data Frames

The most common data type in R is a data frame. This is a collection of vectors, arranged as columns:

```
df = data.frame(ID=1000:1005,
       Gender=c("Male", "Female", "Female",
                "Male", "Female", "Male"),
       Age=c(20, 23, 19, 21, 23, 18),
       T.Shirt.Size=c("Small", "Medium", "Small",
                      "Large","Medium", "Small"))
df
```

```
##     ID Gender Age T.Shirt.Size
## 1 1000   Male  20        Small
## 2 1001 Female  23       Medium
## 3 1002 Female  19        Small
## 4 1003   Male  21        Large
## 5 1004 Female  23       Medium
## 6 1005   Male  18        Small
```

The last variable is defined as a character vector. If we did a table R would sort it alphabetically:

```r
table(df$T.Shirt.Size)
```

```
##
##  Large Medium  Small
##      1      2      3
```

But of course we would expect Small to come before Medium! This can be fixed using an *ordered factor*:

```r
T.Shirt.Size = factor(c("Small", "Medium", "Small",
                      "Large","Medium", "Small"),
                       levels=c("Small", "Medium", "Large"),
                       ordered=TRUE)
df$T.Shirt.Size = T.Shirt.Size
```

```r
table(df$T.Shirt.Size)
```

```
##
##  Small Medium  Large
##      3      2      1
```

- other formats

other common formats are

a. matrices: data frames were all columns have the same data type

b. arrays: higher dimensional matrices

c. lists: collections of data of various types and lenghts. Often used as output from functions.

Basic Commands for Vectors and Data Frames

```r
x = c(1.4, 5.1, 2.0, 6.8, 3.5, 2.1, 5.6, 3.3, 6.9, 1.1)
length(x)
```

```
## [1] 10
```

```r
dim(df)
```

```
## [1] 6 4
```

```r
colnames(df)
```

```
## [1] "ID"           "Gender"       "Age"          "T.Shirt.Size"
```

Subsetting

The elements of a vector or a data frame are accessed with the bracket [ ] notation:

```r
x[3]
```

```
## [1] 2
```

```r
x[1:3]
```

```
## [1] 1.4 5.1 2.0
```

```r
x[c(1, 3, 8)]
```

```
## [1] 1.4 2.0 3.3
```

```r
x[-3]
```

```
## [1] 1.4 5.1 6.8 3.5 2.1 5.6 3.3 6.9 1.1
```

```r
x[-c(1, 2, 5)]
```

```
## [1] 2.0 6.8 2.1 5.6 3.3 6.9 1.1
```

In the case of a data frame we need to specify the row(s) and the column(s):

```r
df[1, 2]
```

```
## [1] "Male"
```

```r
df[1:3, 1]
```

```
## [1] 1000 1001 1002
```

```r
df[, 3]
```

```
## [1] 20 23 19 21 23 18
```

```
df[, -3]
```

```
##     ID Gender T.Shirt.Size
## 1 1000   Male        Small
## 2 1001 Female       Medium
## 3 1002 Female        Small
## 4 1003   Male        Large
## 5 1004 Female       Medium
## 6 1005   Male        Small
```

Subsetting is often done with logic conditions:

```
x
```

```
##  [1] 1.4 5.1 2.0 6.8 3.5 2.1 5.6 3.3 6.9 1.1
```

```
x > 4
```

```
##  [1] FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE
```

```
x[x > 4]
```

```
## [1] 5.1 6.8 5.6 6.9
```

and these can be more complicated:

```
x[x>4 & x<6]
```

```
## [1] 5.1 5.6
```

```
df[df$Age>20, ]
```

```
##     ID Gender Age T.Shirt.Size
## 2 1001 Female  23       Medium
## 4 1003   Male  21        Large
## 5 1004 Female  23       Medium
```

```
df[df$Gender=="Female", ]
```

```
##     ID Gender Age T.Shirt.Size
## 2 1001 Female  23       Medium
## 3 1002 Female  19        Small
## 5 1004 Female  23       Medium
```

Notice the use of $ to access the column names of a data frame.

Vector Arithmetic

R allows us to apply any mathematical function to a whole vector:

```
x = 1:10
2*x
```

```
##  [1]  2  4  6  8 10 12 14 16 18 20
```

```
x^2
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

```
log(x)
```

```
##  [1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379 1.7917595 1.9459101
##  [8] 2.0794415 2.1972246 2.3025851
```

```
sum(x)
```

```
## [1] 55
```

```
y = 21:30
```

```
x+y
```

```
##  [1] 22 24 26 28 30 32 34 36 38 40
```

```
x^2+y^2
```

```
##  [1]  442  488  538  592  650  712  778  848  922 1000
```

```
mean(x+y)
```

```
## [1] 31
```

Programming in R

One of the great strengths of R lies in the fact that it is a full fledged computer language. It includes all the usual things like if-else, for loops etc.

Most data analyses involve many steps:

- getting the data

- cleaning it
- processing it
- analyzing it
- presenting the results with graphs and tables

Having all of these steps together in one function is a very good idea. It allows for quick changes, it allows us to easily make the analysis available to everyone, it allows ourselves to remember what we did some years ago!

R Functions

Here is an example of a function that takes a data frame and standardizes each numeric column:

```r
standardize=function(x) {
  n=ncol(x)
  for(i in 1:n) {
    if(is.numeric(x[, i]))
      x[i, ]=(x[i,]-mean(x[,i]))/sd(x[, i])
  }
  x
}
```

Typically such a function would be saved as an ASCII file with the extension .R in a subdirectory of your project and *loaded* into R when needed.

Rmarkdown

The best way to use R is to create a Rmarkdown document. Such a document combines text, R routines, graphs, tables, etc. into one document. It uses a format that can then be turned into a html, a pdf, a Word, a Powerpoint etc. with the click of a button. For more on Rmarkdown see

An Introduction to Rmarkdown

This workshop was written entirely in Rmarkdown!

Now a function used in the project is just part of the Rmarkdown file.

Case Study: UPR Admissions

Consider the **upr** data set. This is the application data for all the students who applied and were accepted to UPR-Mayaguez between 2003 and 2013.

```
dim(upr)
```

```
## [1] 23666    16
```

tells us that there were 23666 applications and that for each student there are 16 pieces of information.

```
colnames(upr)
```

```
##  [1] "ID.Code"         "Year"            "Gender"          "Program.Code"
##  [5] "Highschool.GPA" "Aptitud.Verbal" "Aptitud.Matem"  "Aprov.Ingles"
##  [9] "Aprov.Matem"     "Aprov.Espanol"   "IGS"             "Freshmen.GPA"
## [13] "Graduated"       "Year.Grad."      "Grad..GPA"       "Class.Facultad"
```

shows us the variables

```
head(upr, 3)
```

```
##      ID.Code Year Gender Program.Code Highschool.GPA Aptitud.Verbal
## 1 00C2B4EF77 2005      M          502           3.97            647
## 2 00D66CF1BF 2003      M          502           3.80            597
## 3 00AB6118EB 2004      M         1203           4.00            567
##   Aptitud.Matem Aprov.Ingles Aprov.Matem Aprov.Espanol IGS Freshmen.GPA
## 1           621          626         672           551 342         3.67
## 2           726          618         718           575 343         2.75
## 3           691          424         616           609 342         3.62
##   Graduated Year.Grad. Grad..GPA Class.Facultad
## 1        Si       2012      3.33           INGE
## 2        No         NA        NA           INGE
## 3        No         NA        NA       CIENCIAS
```

shows us the first three cases.

Let’s say we want to find the number of males and females. We can use the table command for that:

```
table(upr$Gender)
```

```
##
##     F     M
## 11487 12179
```

Let’s answer a few questions regarding the **upr** admissions data:

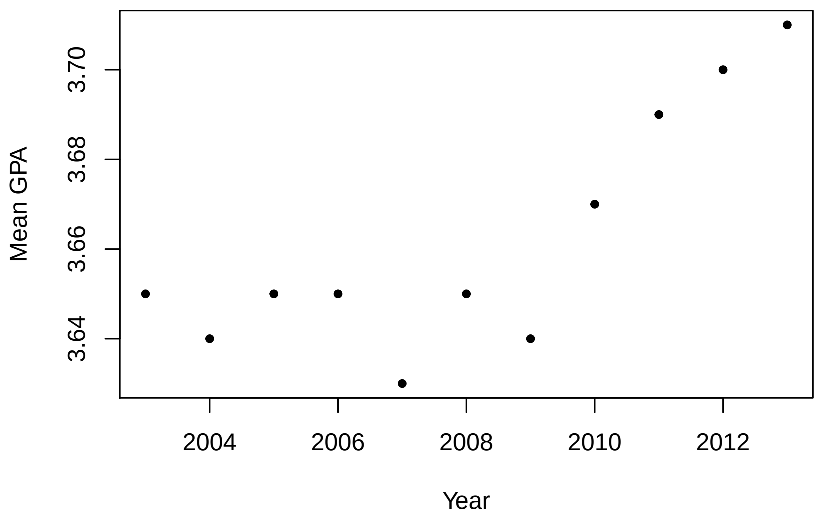
1. How did the number of applications change over the years?

```r
table(upr$Year)
```

```
##
## 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013
## 2253 2158 2300 2235 2464 2438 2417 2031 1772 1748 1850
```

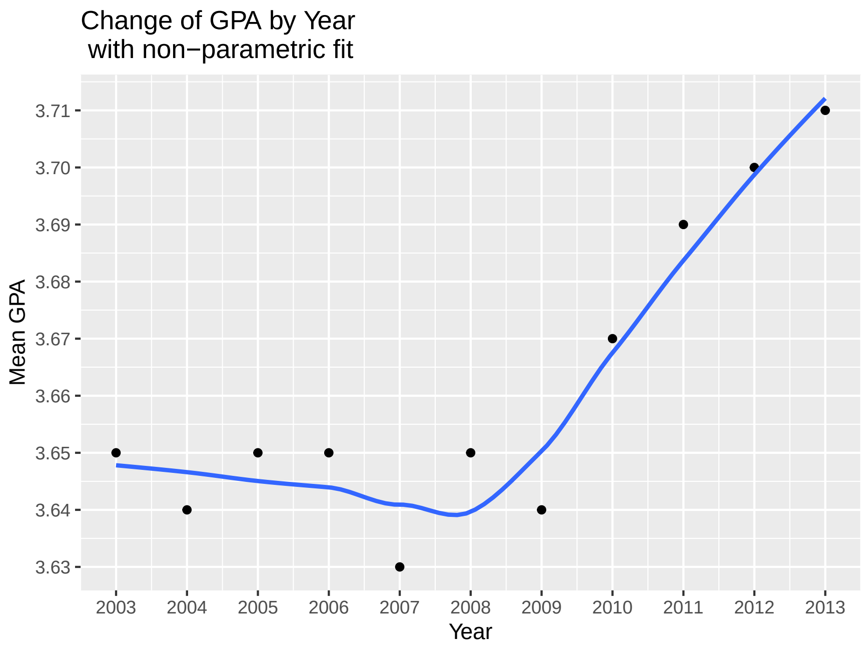
2. Did the High school GPAs change over the years?

```r
yr = unique(upr$Year)
mean.gpa.year = 0*yr
for(i in seq_along(yr)) {
  tmp.gpa = upr$Highschool.GPA[upr$Year==yr[i]]
  mean.gpa.year[i] = round(mean(tmp.gpa), 2)
}
plot(yr, mean.gpa.year,
     pch=20,
     xlab="Year",
     ylab="Mean GPA")
```



There are often many ways in R to do the same thing. Here is another, much nicer but also a bit more complicated solution:

```r
mean.gpa.year = round(tapply(upr$Highschool.GPA,
                             upr$Year,
                             mean), 2)
df = data.frame(GPA = mean.gpa.year,
                 Year = 2003:2013)
ggplot(data=df, aes(Year, GPA)) +
  geom_point() +
  scale_x_continuous(breaks = 2003:2013) +
  scale_y_continuous(breaks = 3.6+0:15/100) +
  labs(x="Year",
       y="Mean GPA",
       title="Change of GPA by Year\n with non-parametric fit") +
  geom_smooth(se=FALSE)
```



Here I make use of a number of nice features:

- the *apply* family of commands allows us the apply calculations to part of the data based on some criterion.
- *ggplot2* is a R library that creates very nice looking graphs
- I also add a *non-parametric regression curve* to the data to show the trend over time

3. How many female students applied between 2009 and 2012 to study Arts and who had a high school GPA over 3.5?

```
table(upr$Class.Facultad)
```

```
##
##     ADEM    ARTES     CIAG CIENCIAS     INGE
##     2492     4124     2326     7014     7710
```

```r
df = upr[upr$Gender=="F", ]
nrow(df)
```

```
## [1] 11487
```

```r
df = df[df$Year>=2009 & df$Year<=2012, ]
nrow(df)
```

```
## [1] 3760
```

```r
df = df[df$Highschool.GPA>3.5, ]
nrow(df)
```

```
## [1] 2948
```

```r
df = df[df$Class.Facultad=="ARTES", ]
nrow(df)
```

```
## [1] 508
```

Note in logic conditions we can use

- == equal to"
- < less than
- <= less or equal to
- > greater than
- >= greater or equal to

- & AND
- | OR

- ! NOT

Random Variates - Simulations

Simulations (aka numerical experiments on a computer) are quickly becoming a standard tool in most fields of science. R is very good at this!

Not surprisingly many standard probability distributions are part of base R. For each the format is

- dname = density

- pname = cumulative distribution function

- rname = random generation
- qname = quantile function

Note we will use the term *density* for both discrete and continuous random variables.

Example Poisson distribution

We have $X \sim \text{Pois}(\lambda)$ if

$$P(X = x) = \frac{\lambda^x}{x!}e^{-\lambda},\, x = 0, 1, ...$$

```r
#density
dpois(c(0, 8, 12, 20), lambda=10)
```

```
## [1] 0.00004539993 0.11259903215 0.09478033009 0.00186608131
```

```r
#cumulative distribution function
ppois(c(0, 8, 12, 20), 10)
```

```
## [1] 0.00004539993 0.33281967875 0.79155647639 0.99841173934
```

```r
#random generation
rpois(15, 10)
```

```
##  [1] 11  8  8  8  6  5  8  9 10  9  8  9  9 10  8
```

```r
#quantiles
qpois(1:4/5, 10)
```

```
## [1]  7  9 11 13
```

Example: what is the probability that an observation drawn from a normal distribution with mean 20 and standard deviation 5 is between 13 and 23?

- exact calculation:

```r
diff(pnorm(c(13, 23), 20, 5))
```

```
## [1] 0.6449902
```

- simulation:

```
x = rnorm(1e4, 20, 5)
length(x[x>=13 &x <=23])/1e4
```

```
## [1] 0.6483
```

Character Strings

Often data is not numeric but consists of things like names, codes etc. These are called *character strings*.

Here is the famous Gettysburg address of Abraham Lincoln:

```
text="Four score and seven years ago our fathers brought forth upon this continent, a ne
```

Let's say we want to know how many words the address had. Right now it is a single long string of characters. In order to break it up into words we can use

```
words=strsplit(text, split=" ")[[1]]
words[1:10]
```

```
##  [1] "Four"    "score"   "and"     "seven"   "years"   "ago"     "our"
##  [8] "fathers" "brought" "forth"
```

```
length(words)
```

```
## [1] 264
```

so it was actually a very short speech! Notice the space between the quotes, this tells R to split the text whenever it finds an empty space.

- how many letters does the address have?

Again we can use the strsplit command, but now we need to split after each letter:

```
letters=strsplit(text, "")[[1]]
letters[1:100]
```

```
##   [1] "F" "o" "u" "r" " " "s" "c" "o" "r" "e" " " "a" "n" "d" " " "s" "e" "v"
##  [19] "e" "n" " " "y" "e" "a" "r" "s" " " "a" "g" "o" " " "o" "u" "r" " " "f"
##  [37] "a" "t" "h" "e" "r" "s" " " "b" "r" "o" "u" "g" "h" "t" " " "f" "o" "r"
##  [55] "t" "h" " " "u" "p" "o" "n" " " "t" "h" "i" "s" " " "c" "o" "n" "t" "i"
##  [73] "n" "e" "n" "t" "," " " "a" " " "n" "e" "w" " " "n" "a" "t" "i" "o" "n"
##  [91] "," " " "c" "o" "n" "c" "e" "i" "v" "e"
```

So far so good, but now we also have the empty spaces. Let's remove them:

```
letters=letters[letters!=" "]
letters[1:100]
```

```
##   [1] "F" "o" "u" "r" "s" "c" "o" "r" "e" "a" "n" "d" "s" "e" "v" "e" "n" "y"
##  [19] "e" "a" "r" "s" "a" "g" "o" "o" "u" "r" "f" "a" "t" "h" "e" "r" "s" "b"
##  [37] "r" "o" "u" "g" "h" "t" "f" "o" "r" "t" "h" "u" "p" "o" "n" "t" "h" "i"
##  [55] "s" "c" "o" "n" "t" "i" "n" "e" "n" "t" "," "a" "n" "e" "w" "n" "a" "t"
##  [73] "i" "o" "n" "," "c" "o" "n" "c" "e" "i" "v" "e" "d" "i" "n" "L" "i" "b"
##  [91] "e" "r" "t" "y" "," "a" "n" "d" "d" "e"
```

We still have the punctuation marks, so let's remove those as well:

```
letters=letters[letters!="."]
letters=letters[letters!=","]
letters=letters[letters!="!"]
letters=letters[letters!="?"]
length(letters)
```

```
## [1] 1159
```

Now we could put all of these back together

```
paste0(letters, collapse="")
```

```
## [1] "Fourscoreandsevenyearsagoourfathersbroughtforthuponthiscontinentanewnationconcei
```

R has many commands to analyze textual data!

Basic Summaries and Graphs

We have talked about the **upr admissions** data before. Here are some simple things to do when looking at this kind of data:

Tables

```
Gender <- table(upr$Gender)
names(Gender) <- c("Female", "Male")
Percentage <- round(Gender/sum(Gender)*100, 1)
cbind(Gender, Percentage) # Put them together as columns
```

```
##        Gender Percentage
## Female  11487       48.5
## Male    12179       51.5
```

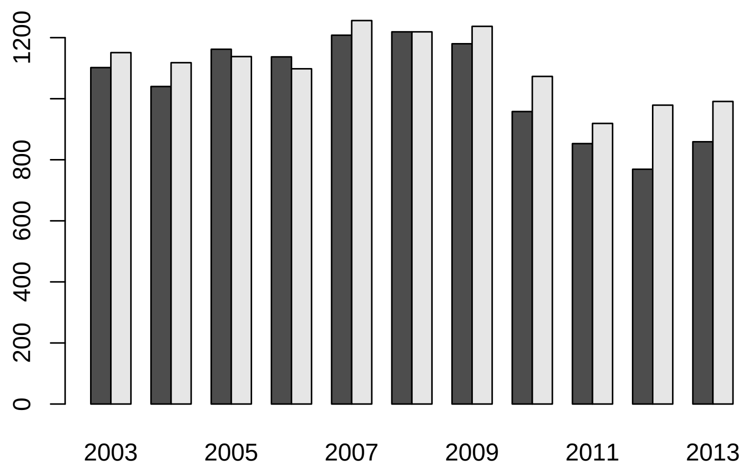
Contingency Tables

```r
table(upr$Year, upr$Gender)
```

```
##
##           F    M
##   2003 1102 1151
##   2004 1040 1118
##   2005 1162 1138
##   2006 1137 1098
##   2007 1208 1256
##   2008 1219 1219
##   2009 1180 1237
##   2010  958 1073
##   2011  853  919
##   2012  769  979
##   2013  859  991
```

Bar Charts

```r
barplot(table(upr$Gender, upr$Year), beside = TRUE)
```



Numerical Summaries

```
mean(upr$Freshmen.GPA)
```

```
## [1] NA
```

this gives an error because for some students the Freshmen GPA is missing (students that dropped out almost immediately, before getting any grades). There is an easy way to deal with such **missing values**:

```
round(mean(upr$Freshmen.GPA, na.rm=TRUE), 3)
```

```
## [1] 2.733
```

```
round(median(upr$Freshmen.GPA, na.rm=TRUE), 3)
```

```
## [1] 2.83
```

```
round(sd(upr$Freshmen.GPA, na.rm=TRUE), 3) # Standard Deviation
```

```
## [1] 0.779
```

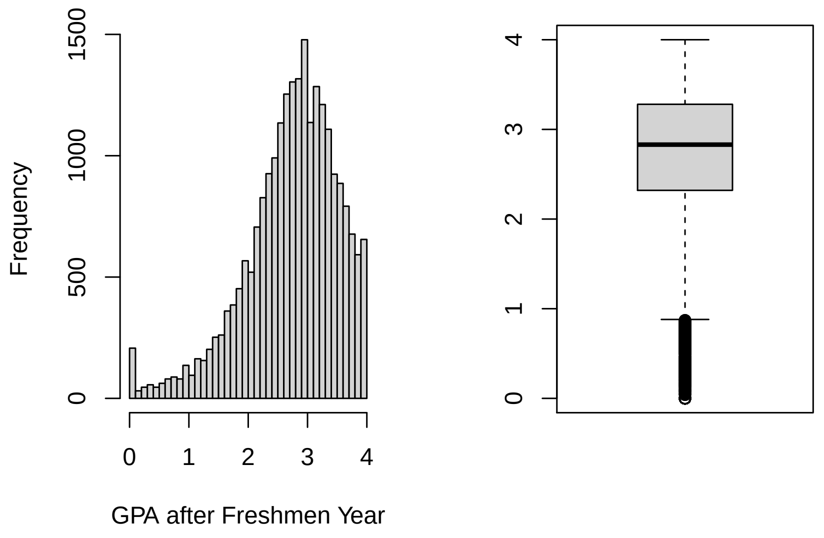
```
round(quantile(upr$Freshmen.GPA,
               probs = c(0.1, 0.25, 0.75, 0.9),
               na.rm=TRUE), 3) # Quantiles and Quartiles
```

```
##  10%  25%  75%  90%
## 1.71 2.32 3.28 3.65
```

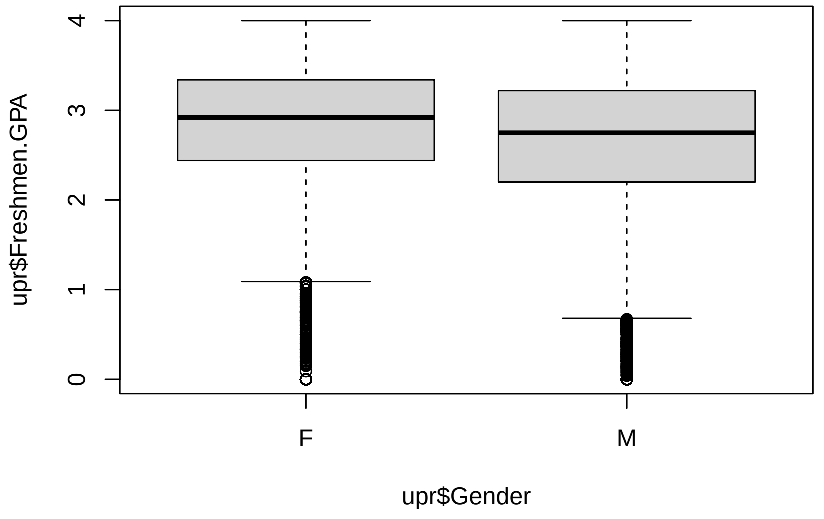
Notice also that I have rounded all the answers. Proper rounding is an important thing to do!

Histogram and Boxplot

```
par(mfrow=c(1, 2))
hist(upr$Freshmen.GPA,
     breaks=50,
     main="",
     xlab="GPA after Freshmen Year")
boxplot(upr$Freshmen.GPA)
```



```r
boxplot(upr$Freshmen.GPA~upr$Gender)
```

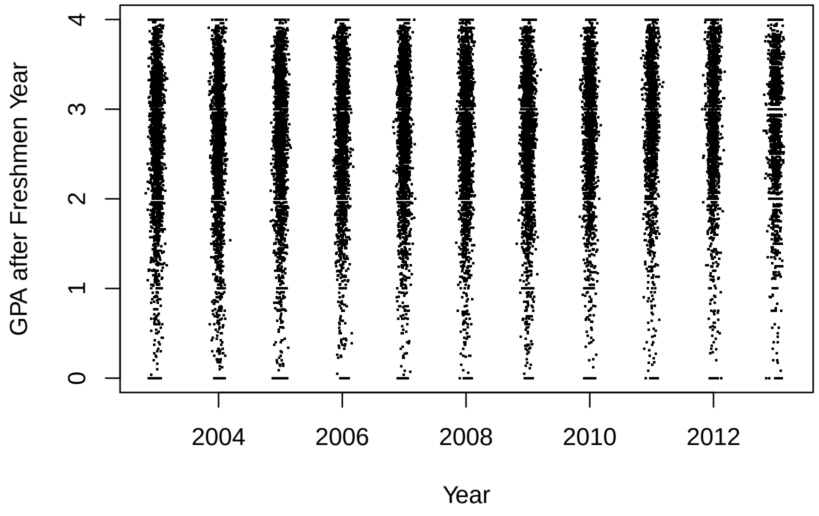


Two Quantitative Variables - Scatterplot

```r
round(cor(upr$Year, upr$Freshmen.GPA,
    use="complete.obs"), 3)
```

```
## [1] 0.097
```

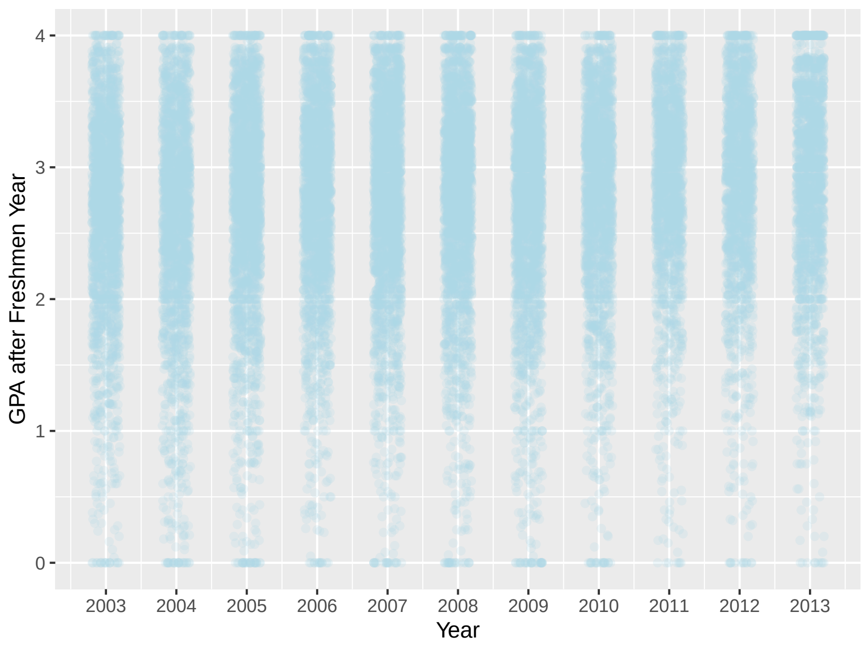
```r
plot(upr$Year + rnorm(length(upr$Year), 0, 0.05),
     upr$Freshmen.GPA,
     xlab="Year",
     pch=".",
     ylab="GPA after Freshmen Year")
```



ggplot2

R can also make much nicer looking graphs using the ggplot2 library, however these are not trivial to make. Here is the same graph, now with ggplot2:

```r
ggplot(data=upr, aes(Year, Freshmen.GPA)) +
  geom_jitter(alpha=0.2, width=0.2,height=0, color="lightblue") +
  labs(x="Year", y="GPA after Freshmen Year") +
  scale_x_continuous(breaks=2003:2013)
```



Inference for a Population Mean

The basic R command for inference for a population mean is *t.test*:

Example: Simon Newcomb's Measurements of the Speed of Light

Simon Newcomb made a series of measurements of the speed of light between July and September 1880. He measured the time in seconds that a light signal took to pass from his laboratory on the Potomac River to a mirror at the base of the Washington Monument and back, a total distance of 7400m. His first measurement was 0.000024828 seconds, or 24,828 nanoseconds (10^9 nanoseconds = 1 second).

We want to find a 95% confidence interval the speed of light.

```
t.test(newcomb$Measurement)
```

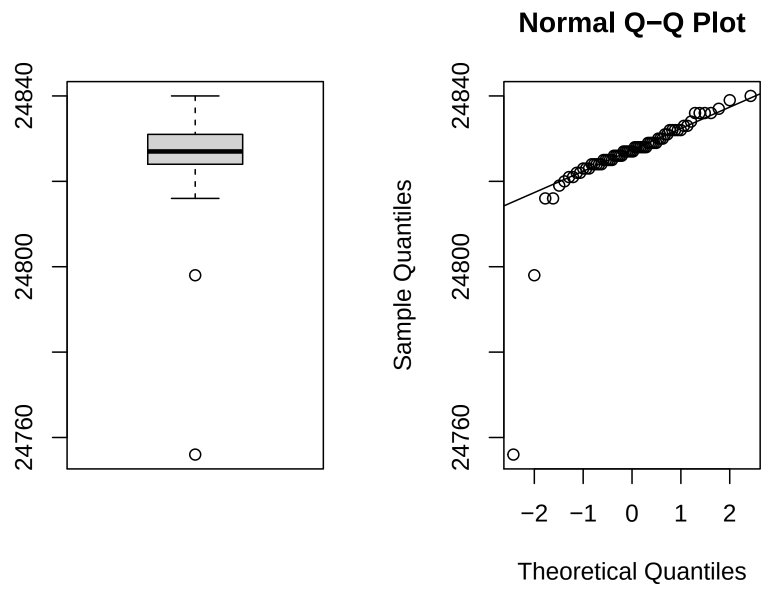
```
##
##  One Sample t-test
##
## data:  newcomb$Measurement
## t = 18770, df = 65, p-value < 0.00000000000000022
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  24823.57 24828.85
## sample estimates:
## mean of x
##  24826.21
```

The assumptions for this method are:

- data comes from a normal distribution
- or data set is large enough

Let's check:

```
par(mfrow=c(1, 2))
boxplot(newcomb$Measurement)
qqnorm(newcomb$Measurement)
qqline(newcomb$Measurement)
```



It seems there is at least one serious outlier on the lower end. This should not happen if the data came from a normal distribution.

We could proceed in one of two ways:

- eliminate outlier:

```
sort(newcomb$Measurement)[1:5]
```

```
## [1] 24756 24798 24816 24816 24819
```

```
x <- newcomb$Measurement[newcomb$Measurement>24800]
t.test(x)$conf.int
```

```
## [1] 24826.48 24829.02
## attr(,"conf.level")
## [1] 0.95
```

- do analysis based on median:

```r
median(newcomb$Measurement)
```

```
## [1] 24827
```

but now we need to find a confidence interval for the median. That can be done with the *non-parametric Wilcoxon Rank Sum* method:

```r
wilcox.test(newcomb$Measurement, conf.int = TRUE)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  newcomb$Measurement
## V = 2211, p-value = 0.000000000001633
## alternative hypothesis: true location is not equal to 0
## 95 percent confidence interval:
##  24826.0 24828.5
## sample estimates:
## (pseudo)median
##        24827.5
```

Example: Resting Period of Monarch Butterflies

Some Monarch butterflies fly early in the day, others somewhat later. After the flight they have to rest for a short period. It has been theorized that the resting period (RIP) of butterflies flying early in the morning is shorter because this is a thermoregulatory mechanism, and it is cooler in the mornings. The mean RIP of all Monarch butterflies is 133 sec. Test the theory at the 10% level.

Research by Anson Lui, Resting period of early and late flying Monarch butterflies Danaeus plexippus, 1997

1. Parameter: mean μ

2. Method: 1-sample t

3. Assumptions: normal data or large sample

4. $\alpha = 0.1$

5. $H_0 : \mu = 133$ (RIP is the same for early morning flying butterflies as all others)

6. $H_0: \mu < 133$ (RIP is the shorter for early morning flying butterflies)

7.

```
t.test(butterflies$RIP.sec.,
       mu=133,
       alternative = "less")$p.value
```

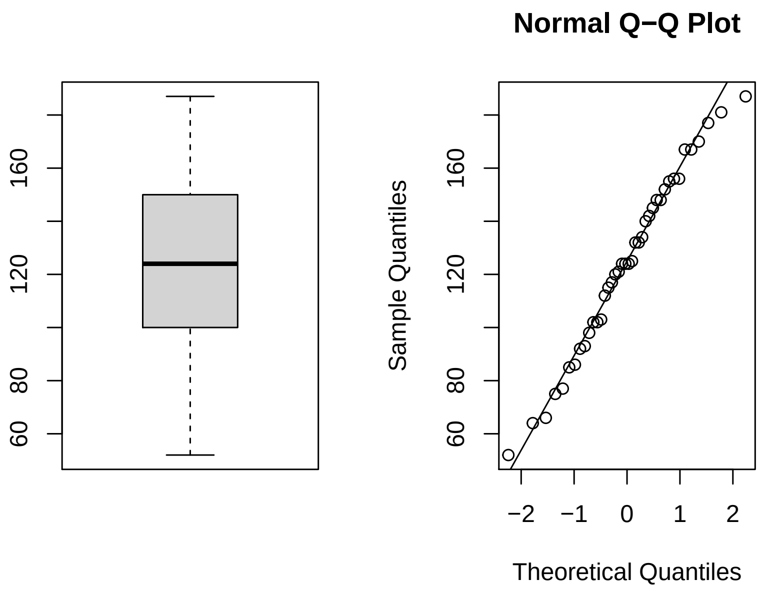
```
## [1] 0.05583963
```

8. $p = 0.0558 < \alpha = 0.1$, so we reject the null hypothesis

9. It appears the resting time is somewhat shorter, but the conclusion is not a strong one.

Checking the assumption:

```
par(mfrow=c(1, 2))
boxplot(butterflies$RIP.sec.)
qqnorm(butterflies$RIP.sec.)
qqline(butterflies$RIP.sec.)
```



looks good.

Inference for a Population Proportion

The R routine for inference for a proportion (or a probability or a percentage) is *binom.test*. This implements a method by Clopper and Pearson (1934). This method is exact and has no assumptions.

Note The formula discussed in many introductory statistic courses for the confidence interval is

$$\hat{p} \pm \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

where $\hat{p}$ is the proportion of success. This leads to confidence intervals that are now known to be quite wrong, and so this method should not be used anymore. The same is true for the corresponding hypothesis test. This method (actually a slight improvement due to Wilson (1927)) is implemented in R by *prop.test*.

Example: Jon Kerrichs Coin

The South African Jon Kerrich spent some time in a German prisoner of war camp during world war I. He used his time to flip a coin 10000 times, resulting in 5067 heads.

Test at the 5% level of significance whether 5067 heads in 10000 flips are compatible with a fair coin.

1. Parameter: proportion π
2. Method: exact binomial
3. Assumptions: None
4. $\alpha = 0.05$
5. $H_0 : \pi = 0.5$ (50% of flips result in "Heads", coin is fair)
6. $H_a : \pi \neq 0.5$ (coin is not fair)
7.

```
binom.test(x = 5067, n = 10000)$p.value
```

```
## [1] 0.1835155
```

8. $p = 0.1835 > \alpha = 0.05$, so we fail to reject the null hypothesis.
9. it appears Jon Kerrich's coin was indeed fair.

Example: Sample Size for Polling

Say some polling institute wants to conduct a poll for the next election for president. They will then find a 95% confidence interval and they want this interval to have an error of 3 percentage points (aka ±0.03). What sample size do they need?

In American politics the two parties are always very close, so in a poll with n people about n/2 will vote for one or the other party. Let's do a little trial and error:

```
n <- 100
diff(as.numeric(binom.test(n/2, n)$conf.int)/2)
```

```
## [1] 0.1016789
```

Now that is to large, so

```
n <- 200
diff(as.numeric(binom.test(n/2, n)$conf.int)/2)
```

```
## [1] 0.07134157
```

```
n <- 400
diff(as.numeric(binom.test(n/2, n)$conf.int)/2)
```

```
## [1] 0.05009211
```

```
n <- 800
diff(as.numeric(binom.test(n/2, n)$conf.int)/2)
```

```
## [1] 0.03521797
```

```
n <- 1200
diff(as.numeric(binom.test(n/2, n)$conf.int)/2)
```

```
## [1] 0.02867679
```

```
n <- 1100
diff(as.numeric(binom.test(n/2, n)$conf.int)/2)
```

```
## [1] 0.02996843
```

```
n <- 1050
diff(as.numeric(binom.test(n/2, n)$conf.int)/2)
```

```
## [1] 0.03068294
```

There is something quite remarkable about this result!

Correlation

**Example:* UPR Admissions data

What are the correlations between the various variables?

```r
head(upr, 2)
```

```
##      ID.Code Year Gender Program.Code Highschool.GPA Aptitud.Verbal
## 1 00C2B4EF77 2005      M          502           3.97            647
## 2 00D66CF1BF 2003      M          502           3.80            597
##   Aptitud.Matem Aprov.Ingles Aprov.Matem Aprov.Espanol IGS Freshmen.GPA
## 1           621          626         672           551 342         3.67
## 2           726          618         718           575 343         2.75
##   Graduated Year.Grad. Grad..GPA Class.Facultad
## 1        Si       2012      3.33           INGE
## 2        No         NA        NA           INGE
```

Let's take out the those variables that are either not numerical or not useful for predicting success, either because we don't have their value at the time of the admissions process (Freshmen.GPA) or for legal reasons (Gender)

```r
x <- upr[, -c(1:4, 11:16)]
head(x, 2)
```

```
##   Highschool.GPA Aptitud.Verbal Aptitud.Matem Aprov.Ingles Aprov.Matem
## 1           3.97            647           621          626         672
## 2           3.80            597           726          618         718
##   Aprov.Espanol
## 1           551
## 2           575
```

```r
round(cor(x, use = "complete.obs") ,3)
```

```
##                Highschool.GPA Aptitud.Verbal Aptitud.Matem Aprov.Ingles
## Highschool.GPA          1.000          0.176         0.156        0.049
## Aptitud.Verbal          0.176          1.000         0.461        0.513
## Aptitud.Matem           0.156          0.461         1.000        0.456
## Aprov.Ingles            0.049          0.513         0.456        1.000
## Aprov.Matem             0.216          0.474         0.819        0.481
## Aprov.Espanol           0.247          0.602         0.389        0.428
##                Aprov.Matem Aprov.Espanol
## Highschool.GPA       0.216         0.247
## Aptitud.Verbal       0.474         0.602
## Aptitud.Matem        0.819         0.389
## Aprov.Ingles         0.481         0.428
## Aprov.Matem          1.000        0.404
## Aprov.Espanol        0.404        1.000
```

notice the surprisingly low correlations between Highschool.GPA and any of the diagnostic exams.

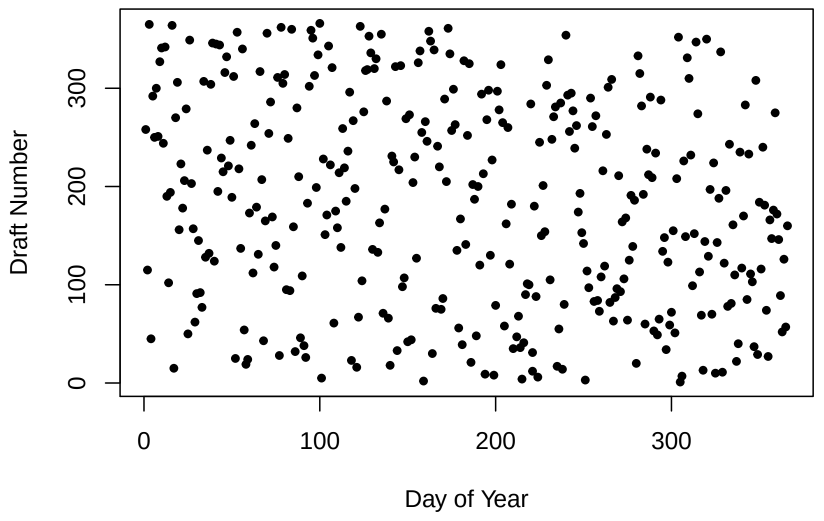
Example: The 1970's Military Draft

In 1970, Congress instituted a random selection process for the military draft. All 366 possible birth dates were placed in plastic capsules in a rotating drum and were selected one by one. The first date drawn from the drum received draft number one and eligible men born on that date were drafted first. In a truly random lottery there should be no relationship between the date and the draft number.

Question: **was the draft was really "random"?**

Here we have two quantitative variables, so we start with the scatterplot:

```r
plot(draft$Draft.Number, draft$Day.of.Year,
     pch=20,
     xlab="Day of Year",
     ylab="Draft Number")
```



and this does not look like there is a problem with independence.

However:

1) Parameter: Pearson's correlation coefficient ρ
2) Method: Test for Pearson's correlation coefficient ρ

3) Assumptions: relationship is linear and that there are no outliers.
4) $\alpha = 0.05$
5) $H_0 : \rho = 0$ (no relationship between Day of Year and Draft Number)
6) $H_a : \rho \neq 0$ (some relationship between Day of Year and Draft Number)
7)

```
cor.test(draft$Draft.Number, draft$Day.of.Year)$p.value
```

```
## [1] 0.00001263829
```

8) $p = 0.0000 < \alpha = 0.05$, so we reject the null hypothesis,
9) There is a statistically significant relationship between Day of Year and Draft Number.

Categorical Data Analysis - Tests for Independence

Example: Drownings in Los Angeles

Data is from O'Carroll PW, Alkon E, Weiss B. Drowning mortality in Los Angeles County, 1976 to 1984, JAMA, 1988 Jul 15;260(3):380-3.

Drowning is the fourth leading cause of unintentional injury death in Los Angeles County. They examined data collected by the Los Angeles County Coroner's Office on drownings that occurred in the county from 1976 through 1984. There were 1587 drownings (1130 males and 457 females) during this nine-year period

```
drownings
```

```
##                         Male Female
## Private Swimming Pool    488    219
## Bathtub                  115    132
## Ocean                    231     40
## Freshwater bodies        155     19
## Hottubs                   16     15
## Reservoirs                32      2
## Other Pools               46     14
## Pails, basins, toilets     7      4
## Other                     40     12
```

Here we have two categorical variables (Method of Drowning and Gender), both categorical. We want to know whether the variables are independent. The most popular method of analysis for this type of problem is **Pearson's chi square test of independence**. It is done with the command *chisq.test* and it has the assumption of no expected counts less than 5.

1. Parameters of interest: measure of association

2. Method of analysis: chi-square test of independence

3. Assumptions of Method: all expected counts greater than 5

4. Type I error probability α=0.05

5. H_0: Classifications are independent = there is no difference in the method of drowning between men and women.

6. H_a: Classifications are dependent = there is some difference in the method of drowning between men and women.

7.

```r
chisq.test(drownings)
```

```
## Warning in chisq.test(drownings): Chi-squared approximation may be incorrect

##
##  Pearson's Chi-squared test
##
## data:  drownings
## X-squared = 144.48, df = 8, p-value < 0.00000000000000022
```

8. p = 0.000 < α=0.05, we reject the null hypothesis, there is a statistically significant difference between men and women and where they drown.

Let's see whether there is a problem with the assumptions:

```r
round(chisq.test(drownings)$expected, 1)
```

```
## Warning in chisq.test(drownings): Chi-squared approximation may be incorrect

##                          Male Female
## Private Swimming Pool   503.4  203.6
## Bathtub                 175.9   71.1
## Ocean                   193.0   78.0
## Freshwater bodies       123.9   50.1
## Hottubs                  22.1    8.9
## Reservoirs               24.2    9.8
## Other Pools              42.7   17.3
## Pails, basins, toilets    7.8    3.2
## Other                    37.0   15.0
```

and we see that the expected counts of Pails, basins, toilets and Female is 3.2. In real life this would be considered ok, but it would also be easy to fix:

```
newmale <- c(drownings[1:7, 1], 7+40)
newfemale <- c(drownings[1:7, 2], 4+12)
newdrown <- cbind(newmale, newfemale)
newdrown
```

```
##                        newmale newfemale
## Private Swimming Pool      488       219
## Bathtub                    115       132
## Ocean                      231        40
## Freshwater bodies          155        19
## Hottubs                     16        15
## Reservoirs                  32         2
## Other Pools                 46        14
##                             47        16
```

```
out <- chisq.test(newdrown)
round(out$expected, 1)
```

```
##                        newmale newfemale
## Private Swimming Pool    503.4     203.6
## Bathtub                  175.9      71.1
## Ocean                    193.0      78.0
## Freshwater bodies        123.9      50.1
## Hottubs                   22.1       8.9
## Reservoirs                24.2       9.8
## Other Pools               42.7      17.3
##                           44.9      18.1
```

```
round(out$p.value, 4)
```

```
## [1] 0
```

Comparing the Means of Several Populations - ANOVA

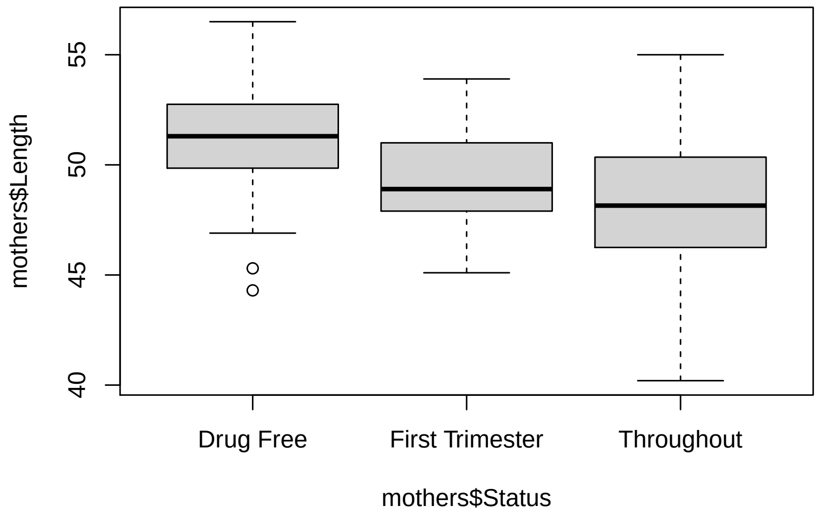
Example: Mothers Cocaine Use and Babies Health

Chasnoff and others obtained several measures and responses for newborn babies whose mothers were classified by degree of cocain use.

The study was conducted in the Perinatal Center for Chemical Dependence at Northwestern University Medical School. The measurement given here is the length of the newborn.

Source: Cocaine abuse during pregnancy: correlation between prenatal care and perinatal outcome
Authors: SN MacGregor, LG Keith, JA Bachicha, and IJ Chasnoff
Obstetrics and Gynecology 1989;74:882-885

```
boxplot(mothers$Length~mothers$Status)
```



```
out <- matrix(0, 3, 3)
colnames(out) <- c("Size", "Mean", "SD")
rownames(out) <- unique(mothers$Status)
out[, 1] <- tapply(mothers$Length,
                   mothers$Status, length)
out[, 2] <- round(tapply(mothers$Length,
                         mothers$Status, mean), 2)
out[, 3] <- round(tapply(mothers$Length,
                         mothers$Status, sd), 2)
out
```

```
##                 Size Mean  SD
## Drug Free         39 51.1 2.9
## First Trimester   19 49.3 2.5
## Throughout        36 48.0 3.6
```

The standard method for this problem is called **ANOVA** (Analysis of Variance) and is run with the *aov* command.

1. Parameters of interest: group means
2. Method of analysis: ANOVA
3. Assumptions of Method: residuals have a normal distribution, groups have equal variance
4. Type I error probability α=0.05
5. Null hypothesis H_0: $\mu_1 = \mu_2 = \mu_3$ (groups have the same means)
6. Alternative hypothesis H_a: $\mu_i \neq \mu_j$ (at least two groups have different means)
7.

```
fit <- aov(mothers$Length~mothers$Status)
summary(fit)
```

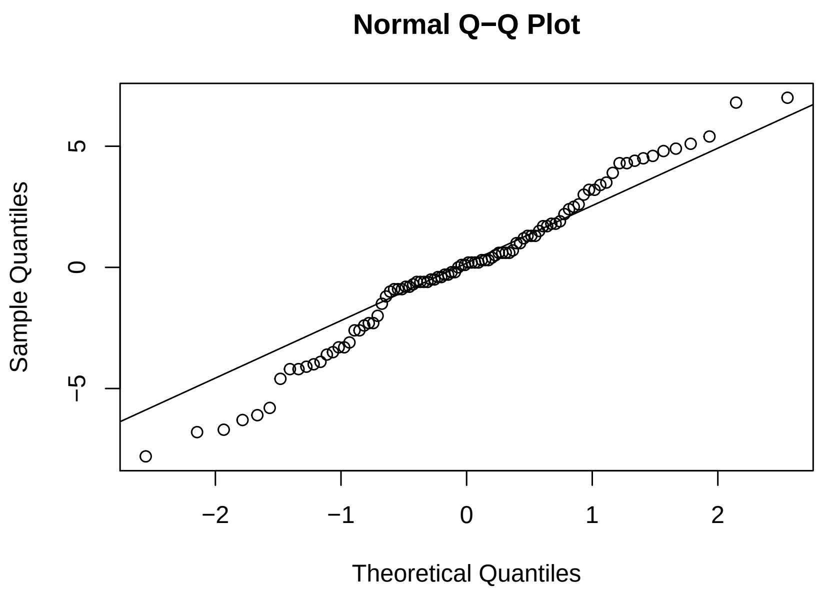
```
##                 Df Sum Sq Mean Sq F value   Pr(>F)
## mothers$Status  2  181.4   90.69   9.319 0.000208
## Residuals      91  885.6    9.73
```

8. $0.0002 < 0.05$, there is some evidence that the group means are not the same, the babies whose mothers used cocain tend to be a little shorter (less healthy?)

In step 3 we have the assumptions

a. residuals have a normal distribution

```
qqnorm(fit$res)
qqline(fit$res)
```



looks fine

b. groups have equal variance

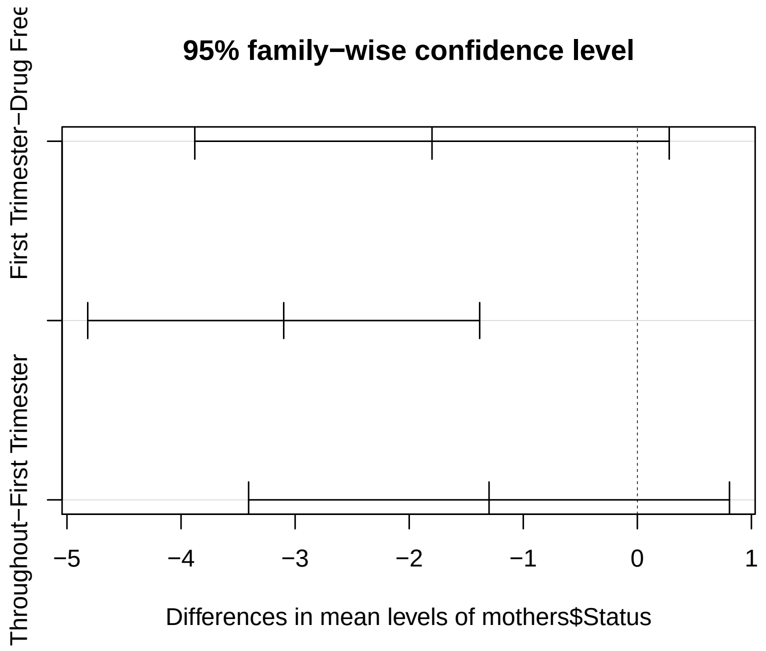
the boxplot shows that the withing group variances re quite similar.

Often if the null of no difference is rejected, one wants to go a step further and do a **pairwise comparison**:

- is Drug Free different from First Trimester?
- is First Trimester different from Throughout?

There are a number of methods known for this problem, a popular one is by **Tukey**:

```
tuk <- TukeyHSD(fit)
plot(tuk)
```



this draws confidence intervals for the difference in means of all pairs. If an interval does not contain 0, the corresponding pair is statistically significantly different.

Here that is the case only for Drug Free - Throughout, so the other two pairs are not statistically significantly different. Remember, however that *failing to reject H_0* is NOT the same as *accepting H_0*. The fact that those pairs are not statistically significantly different is almost certainly due to a lack of sample size.

Regression

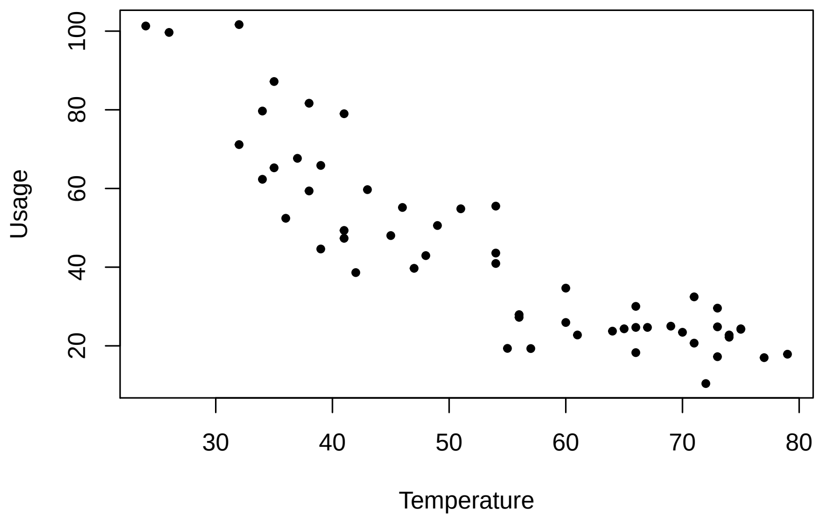
Example: Predicting the Usage of Electricity

In Westchester County, north of New York City, Consolidated Edison bills residential customers for electricity on a monthly basis. The company wants to predict residential usage, in order to plan purchases of fuel and budget revenue flow. The data includes information on usage (in kilowatt-hours per day) and average monthly temperature for 55 consecutive months for an all-electric home. Data on consumption of electricity and the temperature in Westchester County, NY.

```
head(elusage)
```

```
##   Month Year  Usage Temperature
## 1     8 1989 24.828          73
## 2     9 1989 24.688          67
## 3    10 1989 19.310          57
## 4    11 1989 59.706          43
## 5    12 1989 99.667          26
## 6     1 1990 49.333          41
```

```r
plot(elusage$Temperature, elusage$Usage,
     pch=20,
     xlab="Temperature",
     ylab="Usage")
```



We want to find a function Usage $= f(\text{Temperature})$.

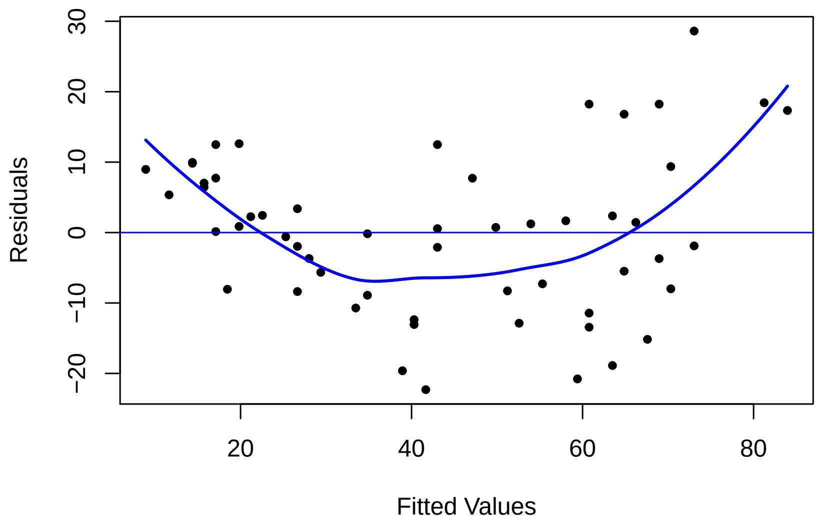
1. Linear Model

```r
fit <- lm(Usage~Temperature, data=elusage)
summary(fit)
```

```
##
## Call:
## lm(formula = Usage ~ Temperature, data = elusage)
##
## Residuals:
##     Min       1Q   Median       3Q      Max
## -22.305   -8.163    0.559    7.723   28.611
##
## Coefficients:
##              Estimate Std. Error t value            Pr(>|t|)
## (Intercept) 116.71619    5.56494   20.97 <0.0000000000000002
## Temperature  -1.36461    0.09941  -13.73 <0.0000000000000002
##
## Residual standard error: 11.35 on 53 degrees of freedom
## Multiple R-squared:  0.7805, Adjusted R-squared:  0.7763
## F-statistic: 188.4 on 1 and 53 DF,  p-value: < 0.00000000000000022
```

How do we know that this is a good model? The main diagnostic is the Residual vs Fits plot:

```
draw.fit <- function(fit) {
  plot(fitted.values(fit), resid(fit),
     pch=20,
     xlab="Fitted Values",
     ylab="Residuals")
  abline(h=0, col="blue")
  fit.loess <- loess(resid(fit)~fitted.values(fit))
  x.range <- seq(min(fitted.values(fit)),
                 max(fitted.values(fit)),
                 length=100)
  y <- predict(fit.loess, newdata=x.range)
  lines(x.range, y, lwd=2, col="blue")
}
draw.fit(fit)
```

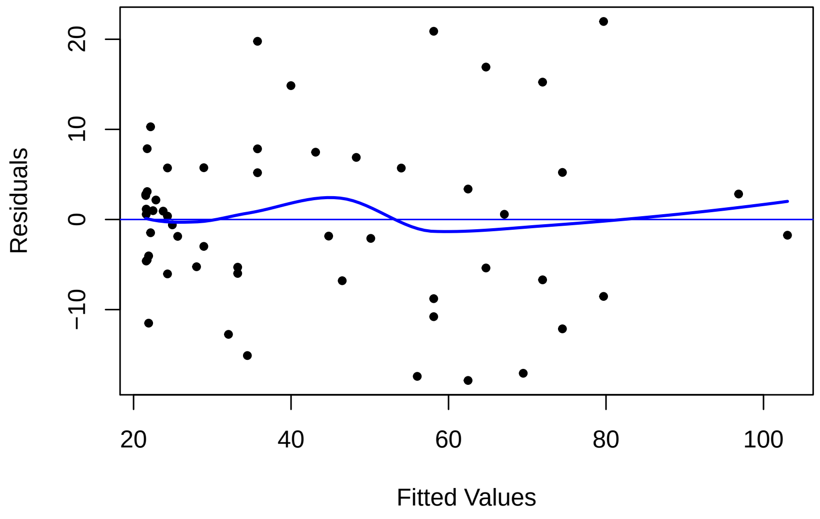


the idea is that if the model is good, the residuals and the fitted values should be independent, and so this graph should not show any pattern. Adding the non-parametric fit and a horizontal line shows that this is not the case here.

- polynomial model

Let's add a quadratic term to the model

```
Temperature2 <- elusage$Temperature^2
quad.fit <- lm(Usage~Temperature + Temperature2,
               data=elusage)
draw.fit(quad.fit)
```

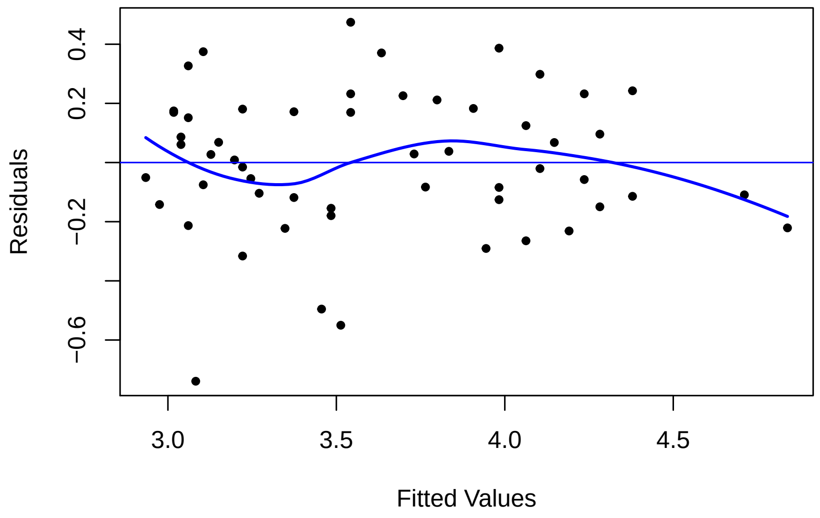


and that is much better.

- Transformations

here we use some transformation (functions) of the data:

```
log.usage <- log(elusage$Usage)
log.temp <- log(elusage$Temperature)
log.fit <- lm(log.usage~log.temp)
draw.fit(log.fit)
```



and that is also not to bad.

How do we choose among these models? A standard measure of the quality of the fit is the **Coefficient of Determination**. It is defined as

$$R^2 = \text{cor(Observed Values, Predicted Values)}^2 100\%$$

Here we find

```
#Linear Model
round(100*unlist(summary(fit)$r.squared), 1)
```

```
## [1] 78
```

```
#Quadratic Model
round(100*unlist(summary(quad.fit)$r.squared), 1)
```

```
## [1] 84.7
```

```
#Log Transform Model
round(100*unlist(summary(log.fit)$r.squared), 1)
```

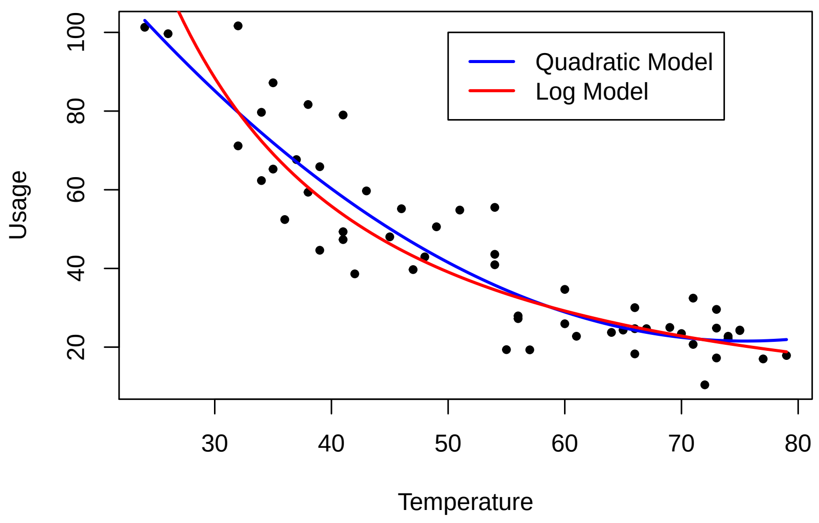
```
## [1] 81.1
```

but we need to be careful here: the linear model is a special case of the quadratic model, and so it's R^2 can never be smaller. There are other ways to choose between such *nested* *models*, for example the F test, but here this is not an issue because the linear model is bad anyway.

Now the R^2 of the quadratic model is 84.7% and that of the log transform model is 81.1%, so the quadratic one is better.

Finally let's have a look what those models look like:

```r
x <- seq(min(elusage$Temperature),
        max(elusage$Temperature), length=100)
y.quad <- predict(quad.fit,
                  newdata=data.frame(Temperature=x,
                                     Temperature2 = x^2))
y.log <- exp(predict(log.fit,
                 newdata=data.frame(log.temp=log(x))))
plot(elusage$Temperature, elusage$Usage,
     pch=20,
     xlab="Temperature",
     ylab="Usage")
lines(x, y.quad, lwd=2, col="blue")
lines(x, y.log, lwd=2, col="red")
legend(50, 100, c("Quadratic Model", "Log Model"),
       lty=c(1, 1), lwd=2,
       col=c("blue", "red"))
```


Quadratic Model
Log Model
Usage
Temperature