National Technical University of Athens
School of Electrical and Computer Engineering
Forecasting & Strategy Unit

# *Forecasting Techniques*

The R software

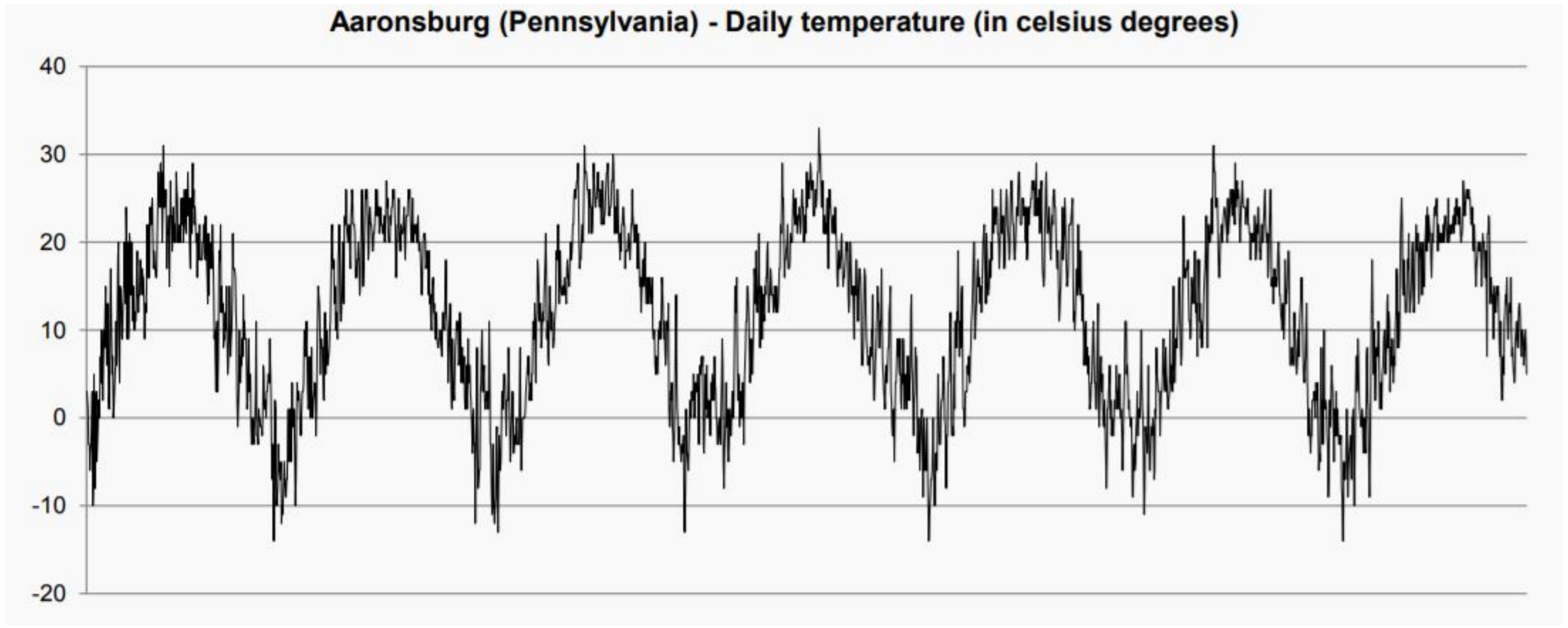
Part #1

FSU

# Why visualize and analyze your data?

Each series displays its own particular **characteristics**

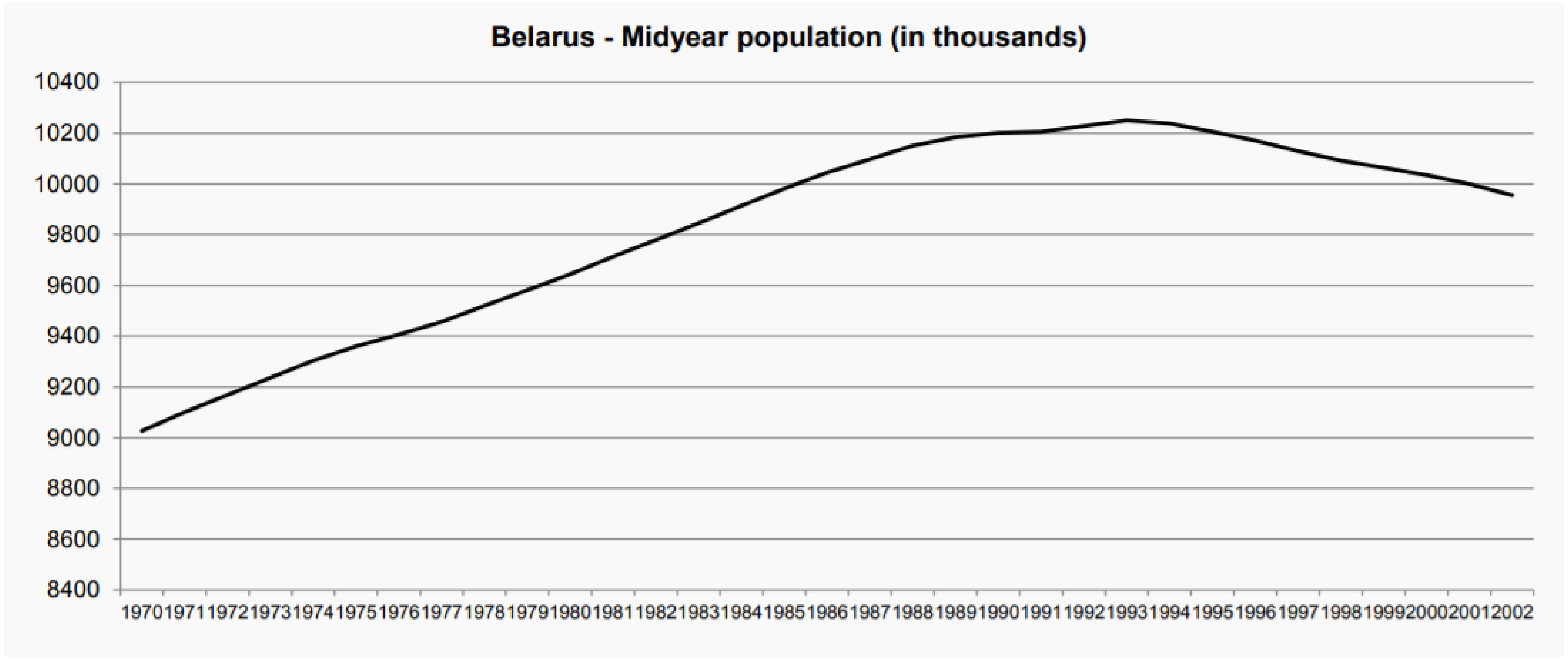
- **Seasonality** (at year, week or day level)



## Why visualize and analyze your data?

Each series displays its own particular **characteristics**

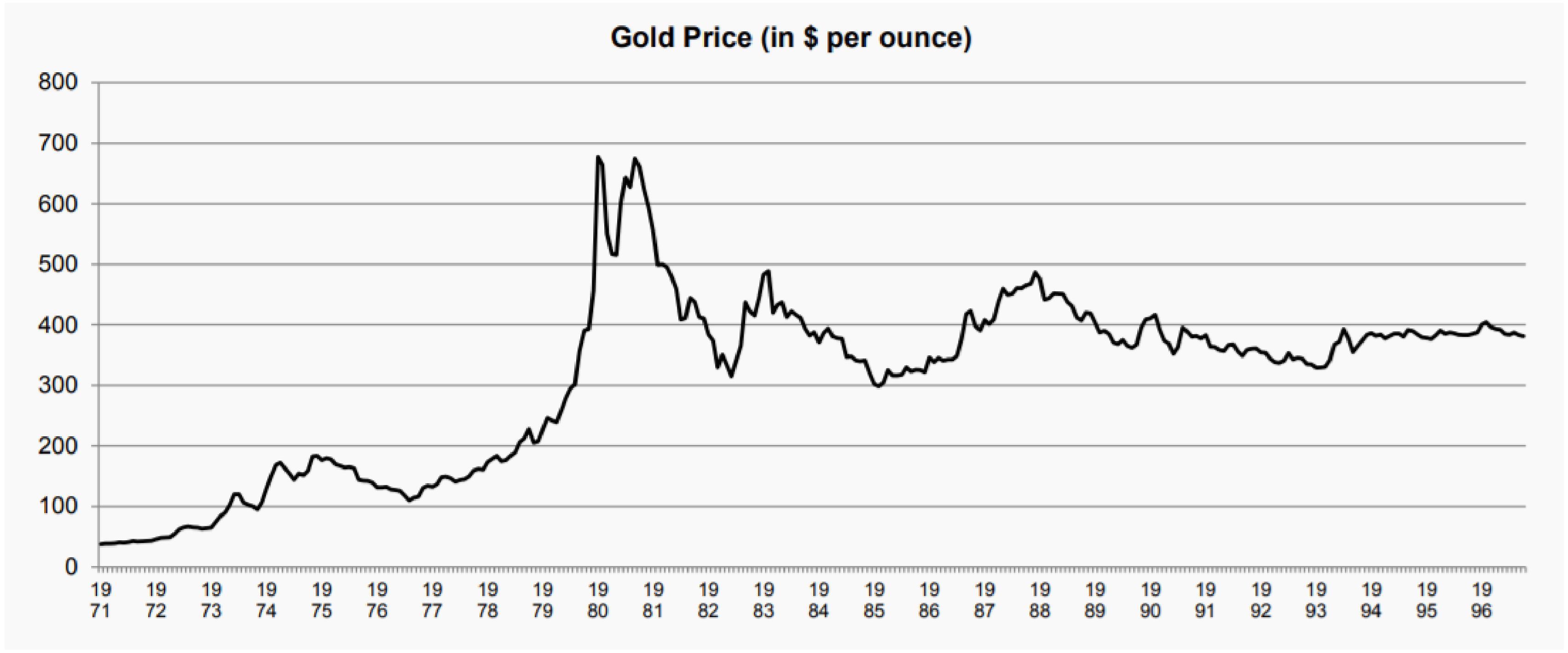
- **Trend** (Linear vs. non-linear and constant vs. changing over time)



## Why visualize and analyze your data?

Each series displays its own particular **characteristics**

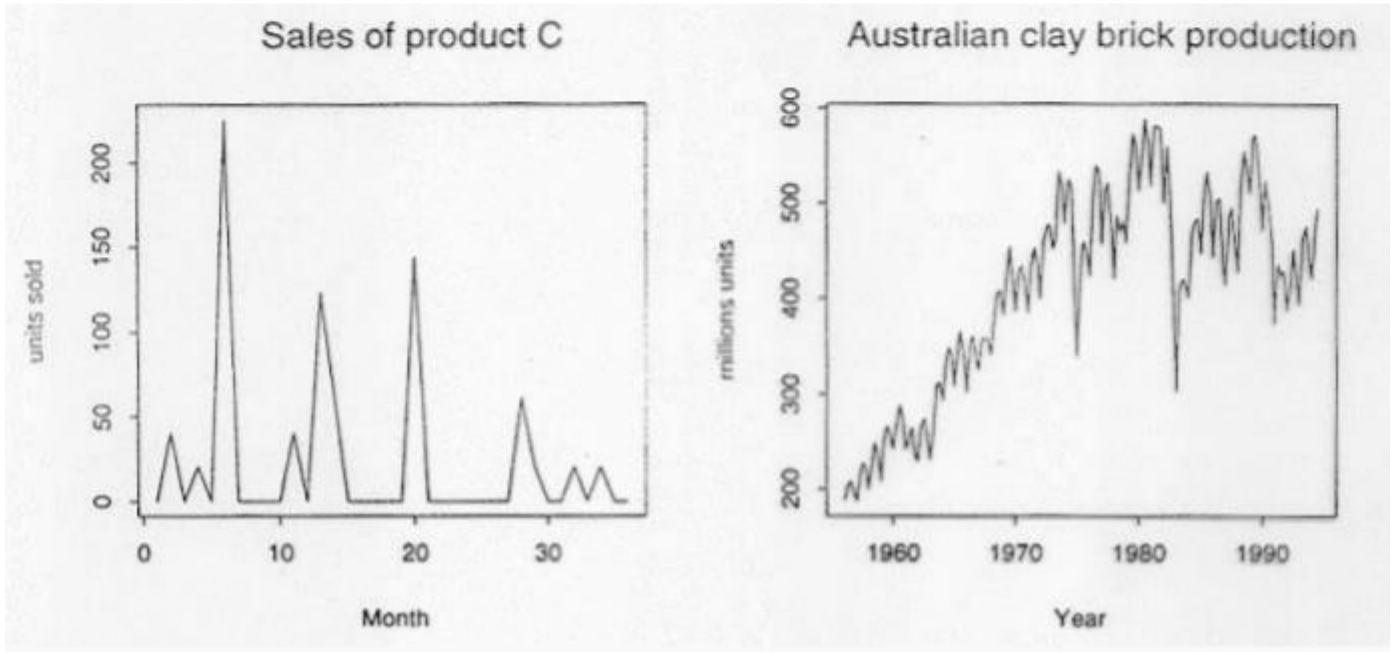
- **Cycle** (which length and intensity may differ over time)



# Why visualize and analyze your data?

Each series displays its own particular **characteristics**

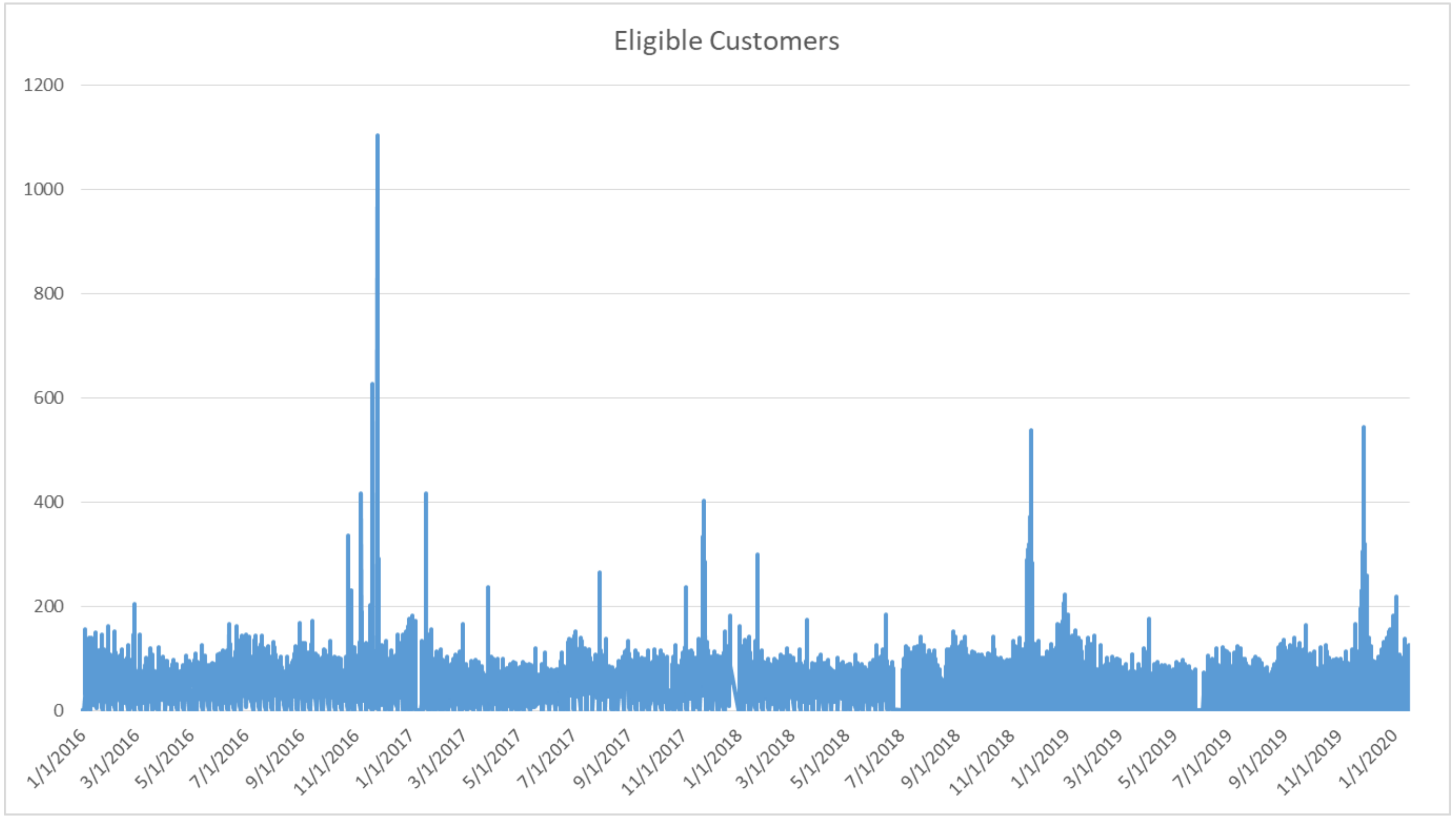
- **Randomness** (as well as outliers and level shifts)



## Why visualize and analyze your data?

Each series displays its own particular **characteristics**

- **Missing values**
- **Special events**



## Why visualize and analyze your data?

Pre-processing typically improves forecasting accuracy

*Spiliotis, E., Assimakopoulos, V., Nikolopoulos, K. (2019). Forecasting with a hybrid method utilizing data smoothing, a variation of the Theta method and shrinkage of seasonal factors. International Journal of Production Economics, 209, 92-102*

There are "Horses for Courses": Each forecasting method is more tailored to some types of data

*Petropoulos, F., Makridakis, S., Assimakopoulos, V., & Nikolopoulos, K. (2014). 'Horses for Courses' in demand forecasting, European Journal of Operational Research, 237 (1), 152-163*

## Why visualize and analyze your data?

You have to understand how the values of the series change over time and which factors affect these changes to select

***the most appropriate forecasting approach***

## The R software package

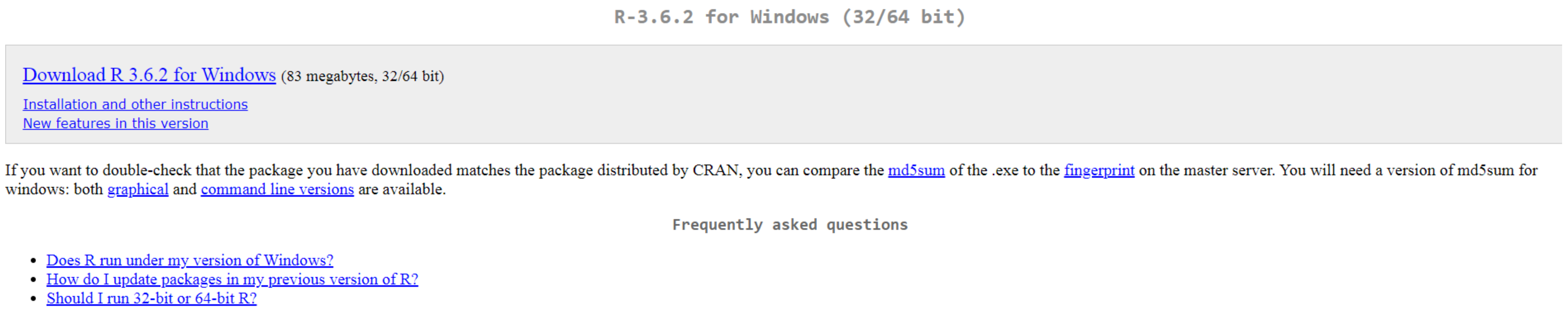
- R is an **open-source** language which has become one of the most **popular** tools for data and predictive analytics
- It compiles and runs on a **wide variety of UNIX platforms**, Windows and MacOS
- R is supported by a **growing community** of more than 2.5 million users and thousands of developers worldwide
- It provides a wide variety of **statistical** and **graphical techniques**, and is highly **extensible**
- This includes packages specifically designed for **supporting forecasting solutions**
- Except the **built-in functions** of R's base package, the users can use hundreds of others available for free or **write their own**

## The CRAN

- CRAN is a **network of ftp and web servers** around the world that store identical, up-to-date, versions of code and documentation for R
- Anyone can create an R package, i.e., a set of functions built in R, and contribute to CRAN so that everyone can use it
- CRAN just makes sure that it will be compatible to different platforms

| | | | |
|---|---|---|---|
| 15,390 | 8,753 | 346 | 31,533,083 |
| active packages | package maintainers | updates last week | downloads last week |

# Installing R



R-3.6.2 for Windows (32/64 bit)

Download R 3.6.2 for Windows (83 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server. You will need a version of md5sum for windows: both graphical and command line versions are available.

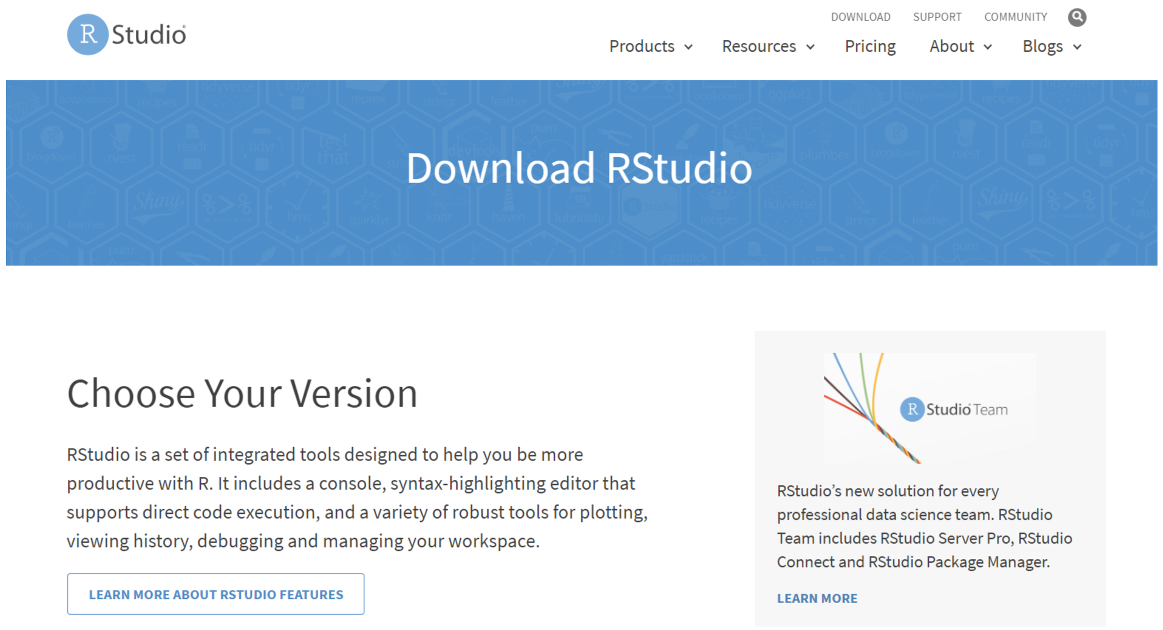
Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

https://cran.r-project.org/bin/windows/base/

- R language is continuously updated. The same stands for the R packages.
- Most of the packages developed using past versions of R will also run under its newer R versions. The opposite is not always true.

## Installing Rstudio



Choose Your Version

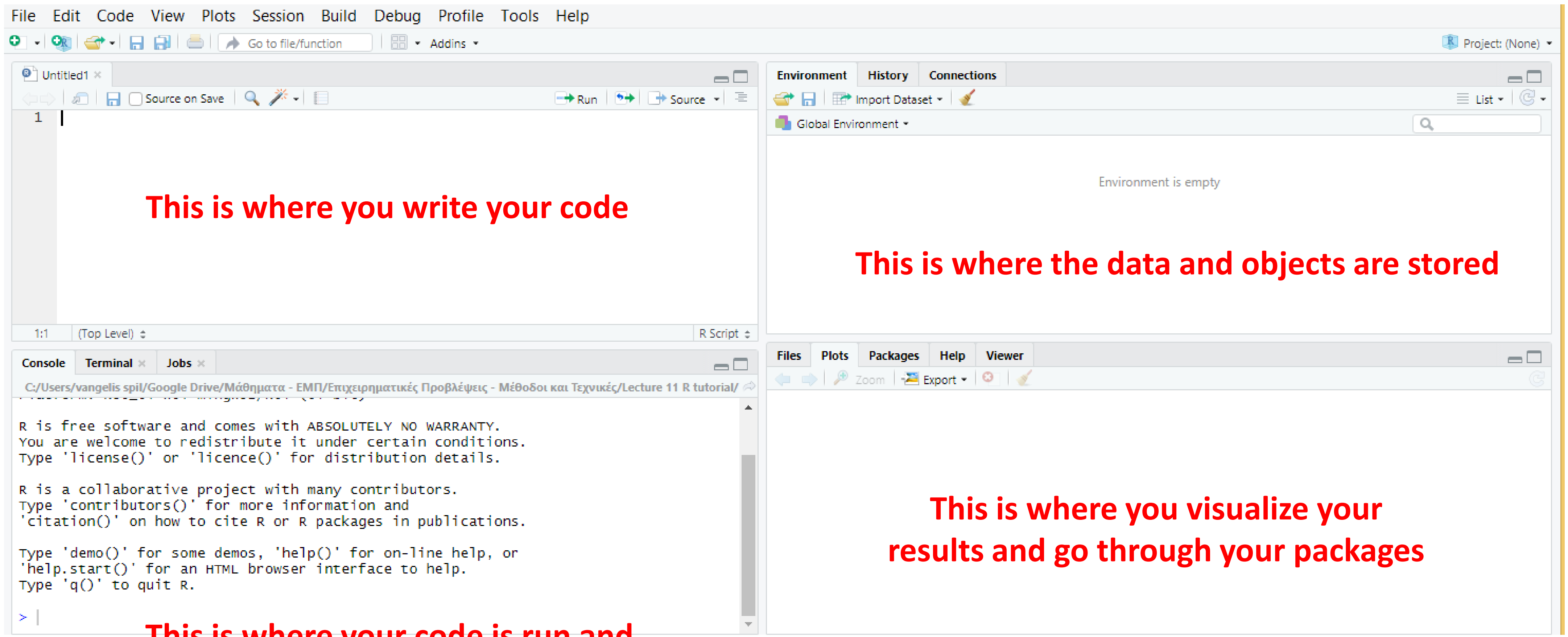
RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, and a variety of robust tools for plotting, viewing history, debugging and managing your workspace.

RStudio's new solution for every professional data science team. RStudio Team includes RStudio Server Pro, RStudio Connect and RStudio Package Manager.

- Rstudio is a free and open tool that helps you use R, visualize your results, and handle your data
- You can still use R without Rstudio by utilizing its terminal. But R studio is much more convenient.

https://rstudio.com/products/rstudio/download/#download

# Using R


File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Untitled1
Source on Save
Run
Source
1:1
(Top Level)
R Script
Environment History Connections
Import Dataset
List
Global Environment
Environment is empty
Console Terminal Jobs
C:/Users/vangelis spil/Google Drive/Μάθηματα - ΕΜΠ/Επιχειρηματικές Προβλέψεις - Μέθοδοι και Τεχνικές/Lecture 11 R tutorial/
Files Plots Packages Help Viewer
Zoom
Export

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

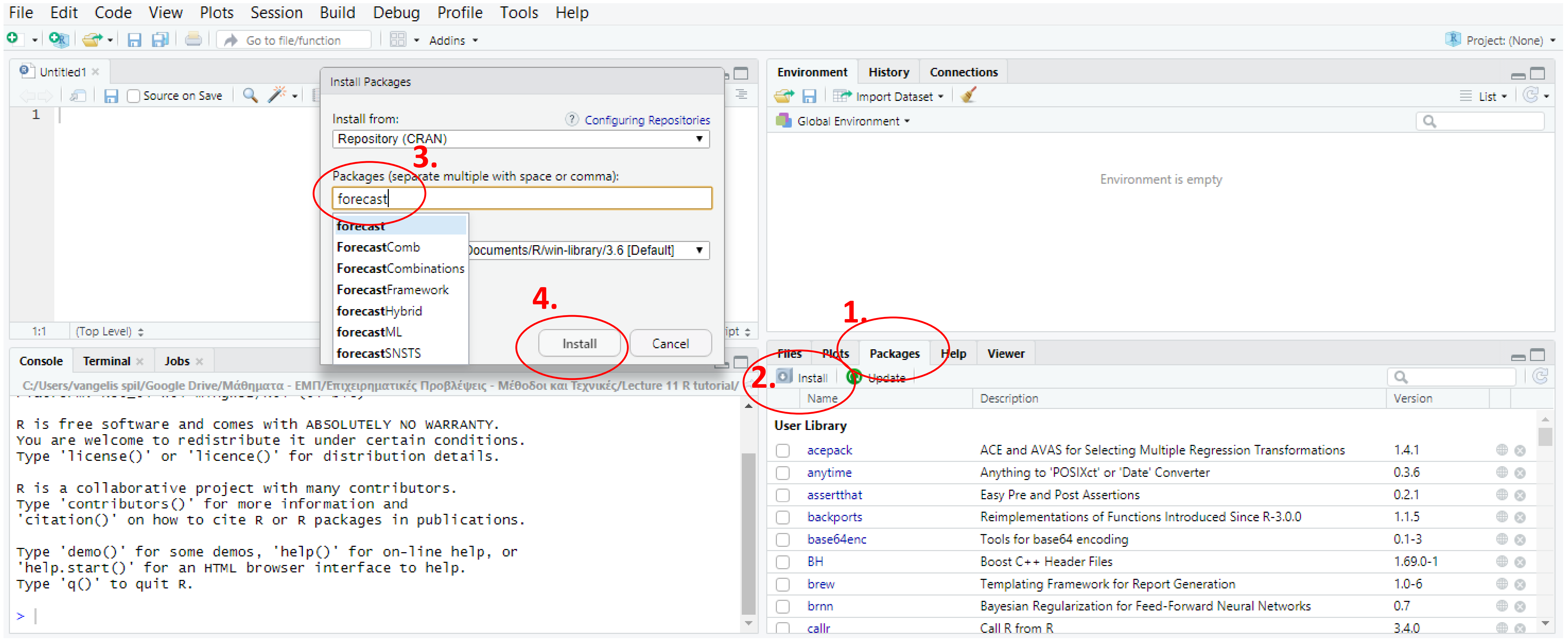
**This is where you write your code**

**This is where the data and objects are stored**

**This is where your code is run and results are printed**

**This is where you visualize your results and go through your packages**

# Installing R packages

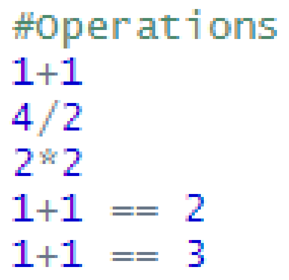


C:/Users/vangelis spil/Google Drive/Μάθηματα - ΕΜΠ/Επιχειρηματικές Προβλέψεις - Μέθοδοι και Τεχνικές/Lecture 11 R tutorial/

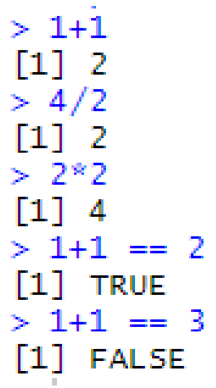
```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

| Name | Description | Version |
|---|---|---|
| **User Library** | | |
| acepack | ACE and AVAS for Selecting Multiple Regression Transformations | 1.4.1 |
| anytime | Anything to 'POSIXct' or 'Date' Converter | 0.3.6 |
| assertthat | Easy Pre and Post Assertions | 0.2.1 |
| backports | Reimplementations of Functions Introduced Since R-3.0.0 | 1.1.5 |
| base64enc | Tools for base64 encoding | 0.1-3 |
| BH | Boost C++ Header Files | 1.69.0-1 |
| brew | Templating Framework for Report Generation | 1.0-6 |
| brnn | Bayesian Regularization for Feed-Forward Neural Networks | 0.7 |
| callr | Call R from R | 3.4.0 |

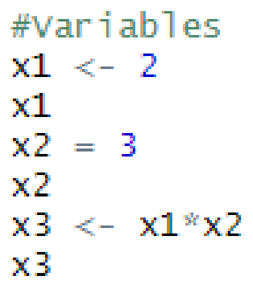
## The basics: Operations & Variables



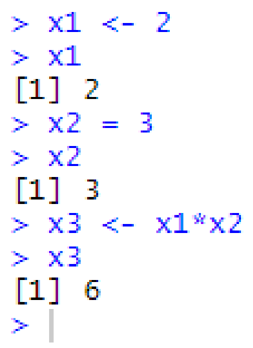
```
#Operations
1+1
4/2
2*2
1+1 == 2
1+1 == 3
```



```
> 1+1
[1] 2
> 4/2
[1] 2
> 2*2
[1] 4
> 1+1 == 2
[1] TRUE
> 1+1 == 3
[1] FALSE
```

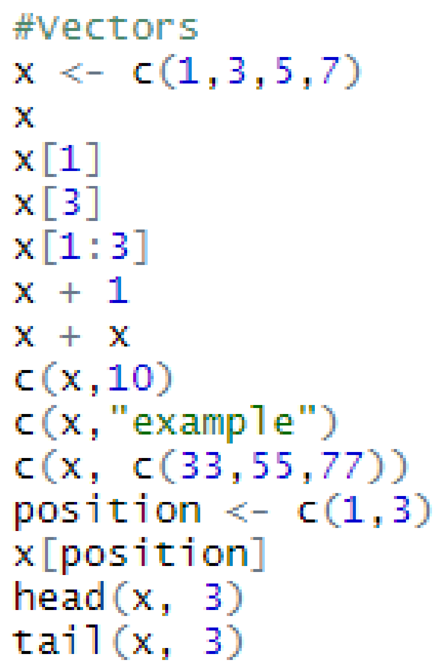


```
#Variables
x1 <- 2
x1
x2 = 3
x2
x3 <- x1*x2
x3
```

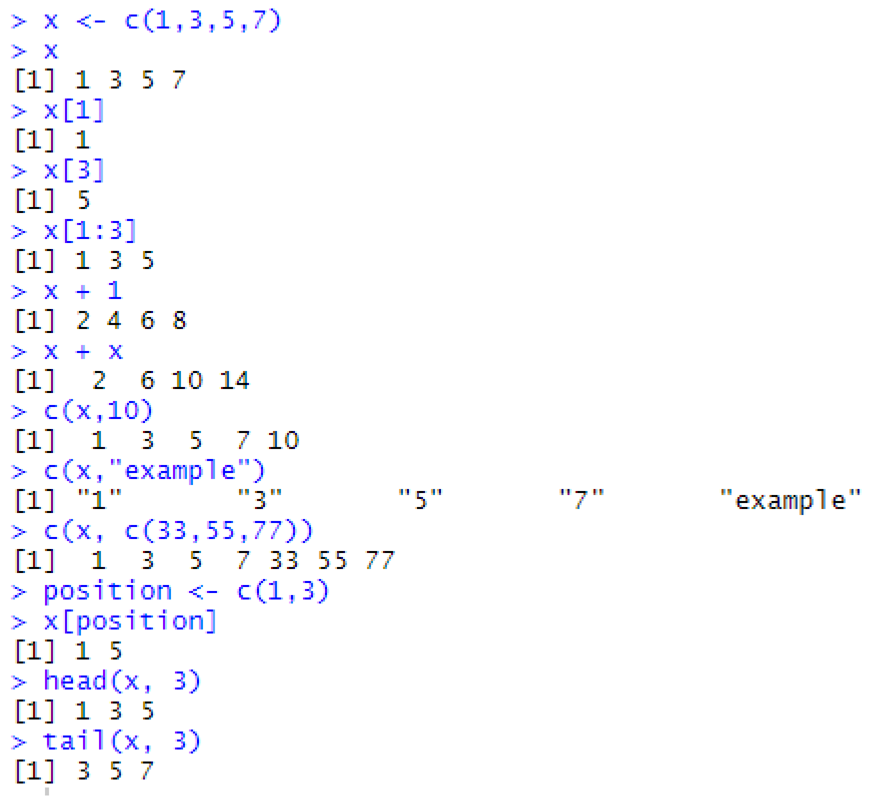


```
> x1 <- 2
> x1
[1] 2
> x2 = 3
> x2
[1] 3
> x3 <- x1*x2
> x3
[1] 6
>
```

# The basics: Vectors

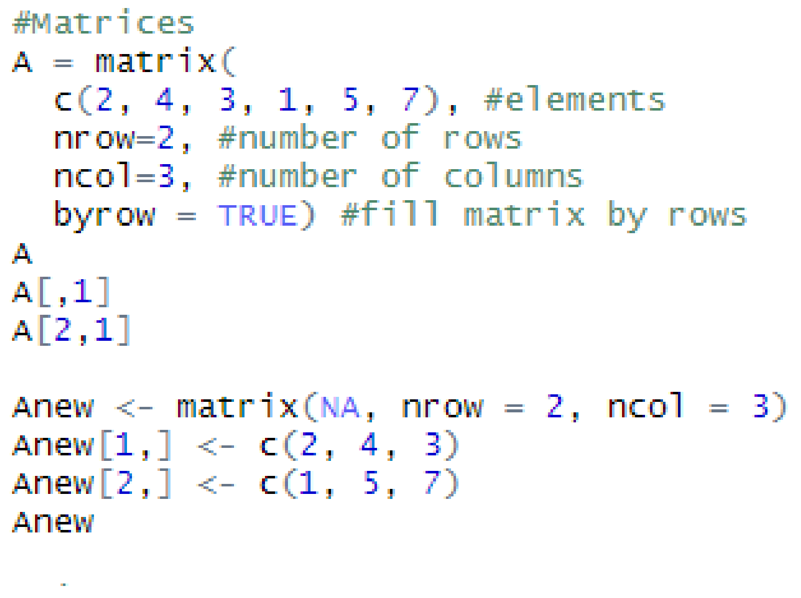


```
#Vectors
x <- c(1,3,5,7)
x
x[1]
x[3]
x[1:3]
x + 1
x + x
c(x,10)
c(x,"example")
c(x, c(33,55,77))
position <- c(1,3)
x[position]
head(x, 3)
tail(x, 3)
```

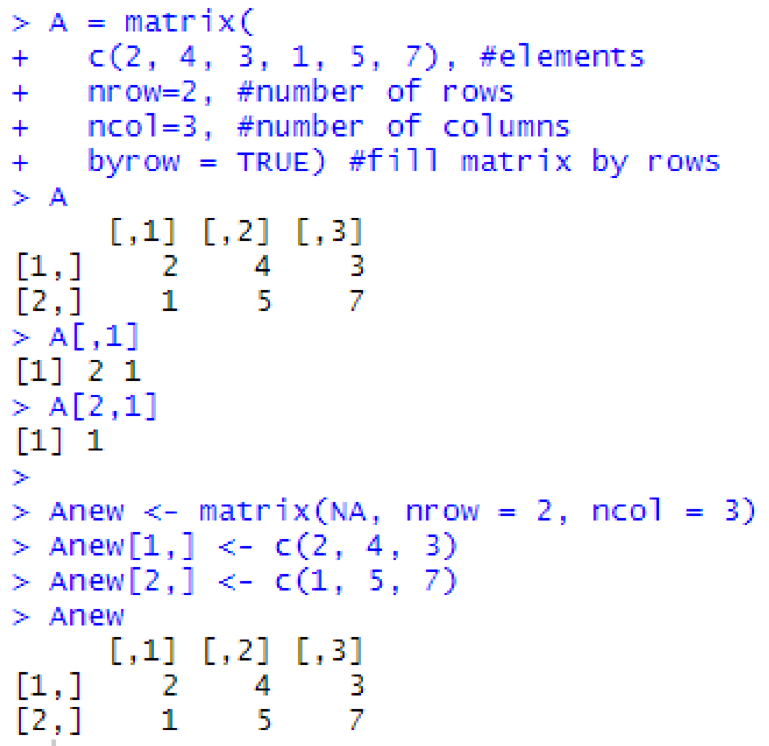


```
> x <- c(1,3,5,7)
> x
[1] 1 3 5 7
> x[1]
[1] 1
> x[3]
[1] 5
> x[1:3]
[1] 1 3 5
> x + 1
[1] 2 4 6 8
> x + x
[1]  2  6 10 14
> c(x,10)
[1]  1  3  5  7 10
> c(x,"example")
[1] "1"       "3"       "5"       "7"       "example"
> c(x, c(33,55,77))
[1]  1  3  5  7 33 55 77
> position <- c(1,3)
> x[position]
[1] 1 5
> head(x, 3)
[1] 1 3 5
> tail(x, 3)
[1] 3 5 7
```

## The basics: Matrices

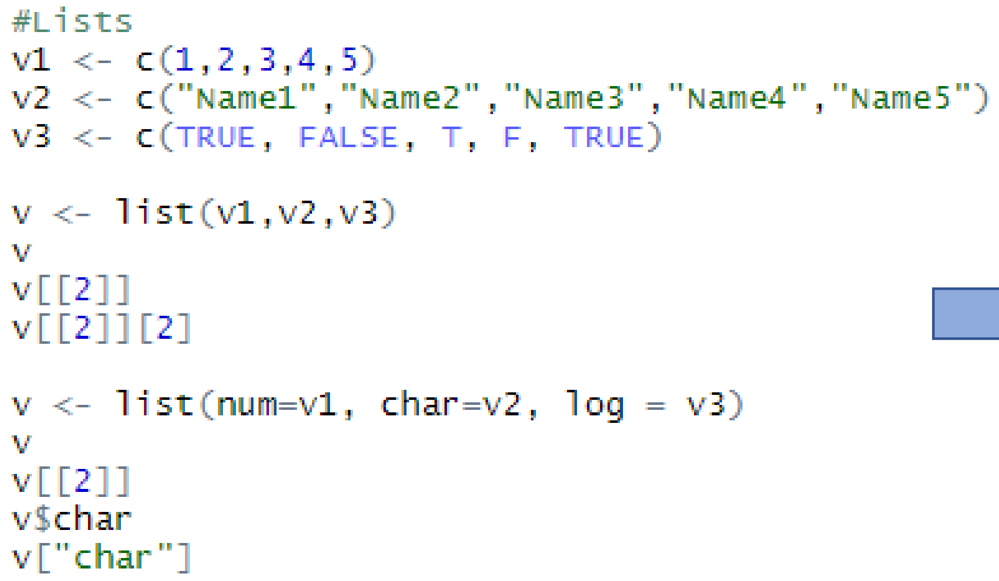


```
#Matrices
A = matrix(
  c(2, 4, 3, 1, 5, 7), #elements
  nrow=2, #number of rows
  ncol=3, #number of columns
  byrow = TRUE) #fill matrix by rows
A
A[,1]
A[2,1]

Anew <- matrix(NA, nrow = 2, ncol = 3)
Anew[1,] <- c(2, 4, 3)
Anew[2,] <- c(1, 5, 7)
Anew
```

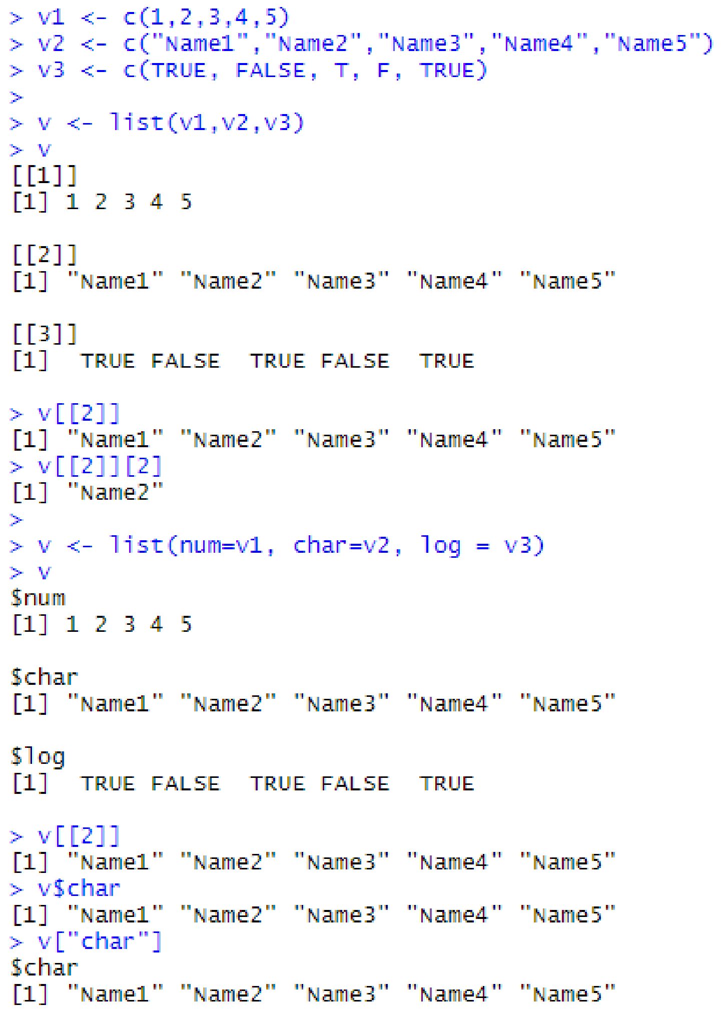


```
> A = matrix(
+   c(2, 4, 3, 1, 5, 7), #elements
+   nrow=2, #number of rows
+   ncol=3, #number of columns
+   byrow = TRUE) #fill matrix by rows
> A
     [,1] [,2] [,3]
[1,]    2    4    3
[2,]    1    5    7
> A[,1]
[1] 2 1
> A[2,1]
[1] 1
>
> Anew <- matrix(NA, nrow = 2, ncol = 3)
> Anew[1,] <- c(2, 4, 3)
> Anew[2,] <- c(1, 5, 7)
> Anew
     [,1] [,2] [,3]
[1,]    2    4    3
[2,]    1    5    7
```

# The basics: Lists

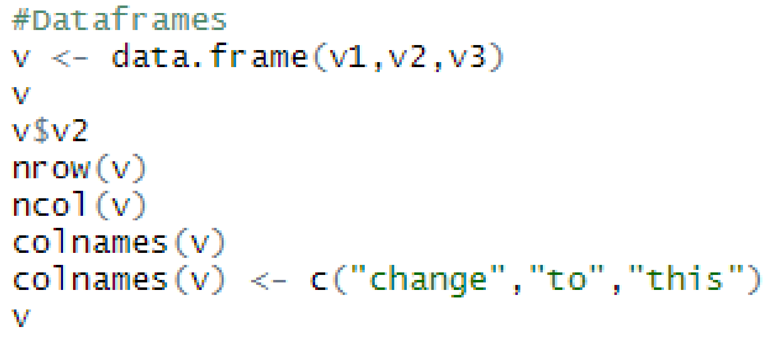


```r
#Lists
v1 <- c(1,2,3,4,5)
v2 <- c("Name1","Name2","Name3","Name4","Name5")
v3 <- c(TRUE, FALSE, T, F, TRUE)

v <- list(v1,v2,v3)
v
v[[2]]
v[[2]][2]

v <- list(num=v1, char=v2, log = v3)
v
v[[2]]
v$char
v["char"]
```

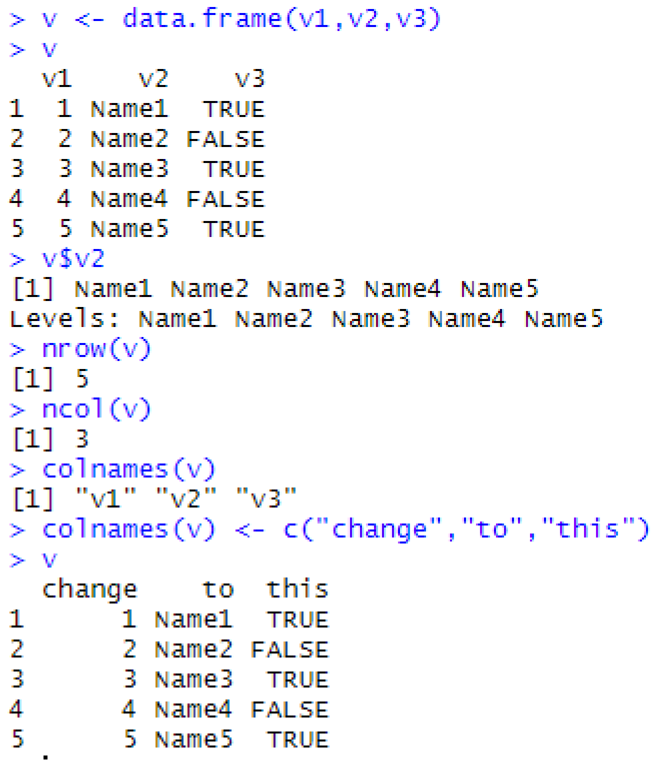


```
> v1 <- c(1,2,3,4,5)
> v2 <- c("Name1","Name2","Name3","Name4","Name5")
> v3 <- c(TRUE, FALSE, T, F, TRUE)
>
> v <- list(v1,v2,v3)
> v
[[1]]
[1] 1 2 3 4 5

[[2]]
[1] "Name1" "Name2" "Name3" "Name4" "Name5"

[[3]]
[1]  TRUE FALSE  TRUE FALSE  TRUE

> v[[2]]
[1] "Name1" "Name2" "Name3" "Name4" "Name5"
> v[[2]][2]
[1] "Name2"
>
> v <- list(num=v1, char=v2, log = v3)
> v
$num
[1] 1 2 3 4 5

$char
[1] "Name1" "Name2" "Name3" "Name4" "Name5"

$log
[1]  TRUE FALSE  TRUE FALSE  TRUE

> v[[2]]
[1] "Name1" "Name2" "Name3" "Name4" "Name5"
> v$char
[1] "Name1" "Name2" "Name3" "Name4" "Name5"
> v["char"]
$char
[1] "Name1" "Name2" "Name3" "Name4" "Name5"
```

## The basics: Data frames

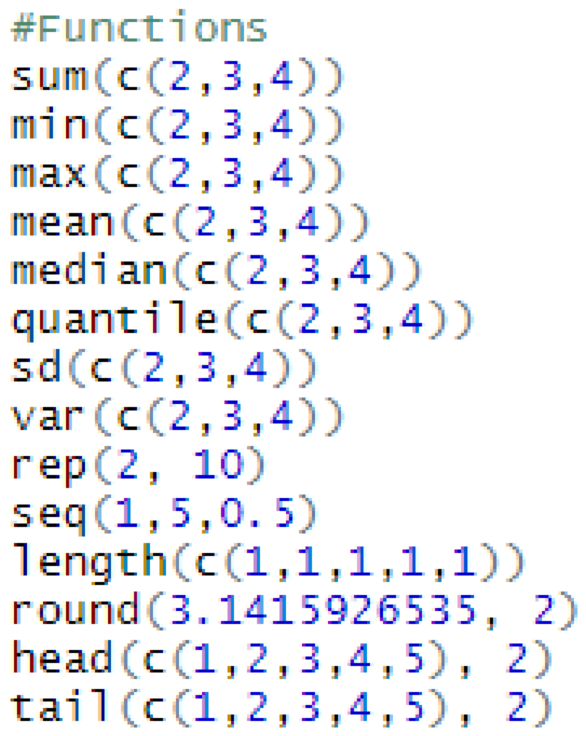


```
#Dataframes
v <- data.frame(v1,v2,v3)
v
v$v2
nrow(v)
ncol(v)
colnames(v)
colnames(v) <- c("change","to","this")
v
```

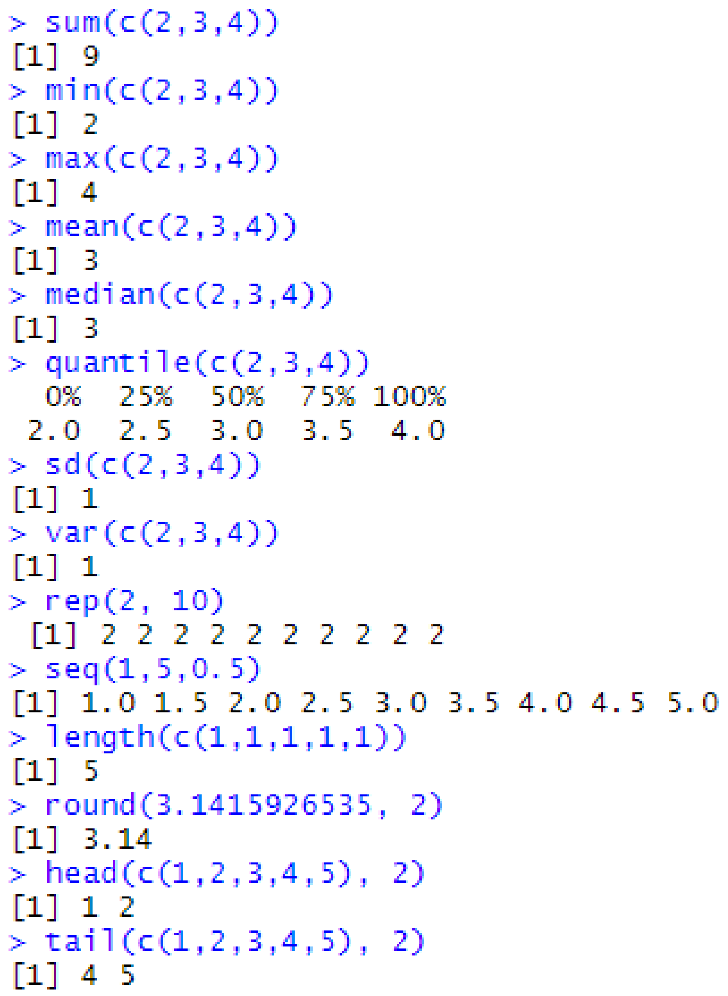


```
> v <- data.frame(v1,v2,v3)
> v
  v1    v2    v3
1  1 Name1  TRUE
2  2 Name2 FALSE
3  3 Name3  TRUE
4  4 Name4 FALSE
5  5 Name5  TRUE
> v$v2
[1] Name1 Name2 Name3 Name4 Name5
Levels: Name1 Name2 Name3 Name4 Name5
> nrow(v)
[1] 5
> ncol(v)
[1] 3
> colnames(v)
[1] "v1" "v2" "v3"
> colnames(v) <- c("change","to","this")
> v
  change    to  this
1      1 Name1  TRUE
2      2 Name2 FALSE
3      3 Name3  TRUE
4      4 Name4 FALSE
5      5 Name5  TRUE
```

## The basics: Functions



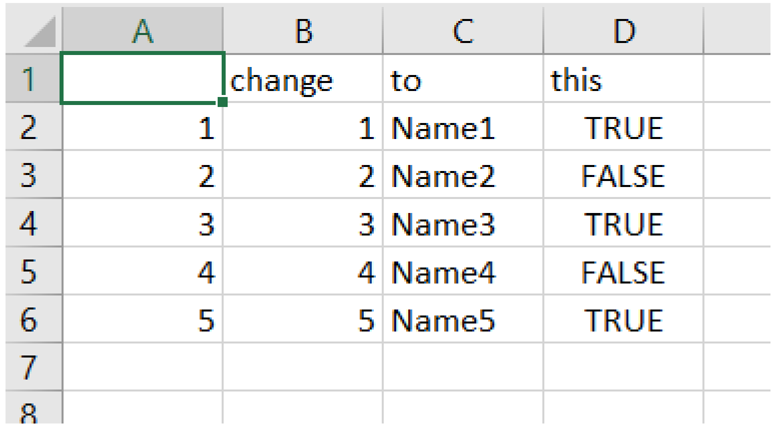
```
#Functions
sum(c(2,3,4))
min(c(2,3,4))
max(c(2,3,4))
mean(c(2,3,4))
median(c(2,3,4))
quantile(c(2,3,4))
sd(c(2,3,4))
var(c(2,3,4))
rep(2, 10)
seq(1,5,0.5)
length(c(1,1,1,1,1))
round(3.1415926535, 2)
head(c(1,2,3,4,5), 2)
tail(c(1,2,3,4,5), 2)
```



```
> sum(c(2,3,4))
[1] 9
> min(c(2,3,4))
[1] 2
> max(c(2,3,4))
[1] 4
> mean(c(2,3,4))
[1] 3
> median(c(2,3,4))
[1] 3
> quantile(c(2,3,4))
  0%  25%  50%  75% 100% 
 2.0  2.5  3.0  3.5  4.0 
> sd(c(2,3,4))
[1] 1
> var(c(2,3,4))
[1] 1
> rep(2, 10)
 [1] 2 2 2 2 2 2 2 2 2 2
> seq(1,5,0.5)
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
> length(c(1,1,1,1,1))
[1] 5
> round(3.1415926535, 2)
[1] 3.14
> head(c(1,2,3,4,5), 2)
[1] 1 2
> tail(c(1,2,3,4,5), 2)
[1] 4 5
```

# The basics: Files (input & output)

```
write.csv(v, "C:/Users/vangelis spil/Desktop/ft_project_insample_1.csv")
```



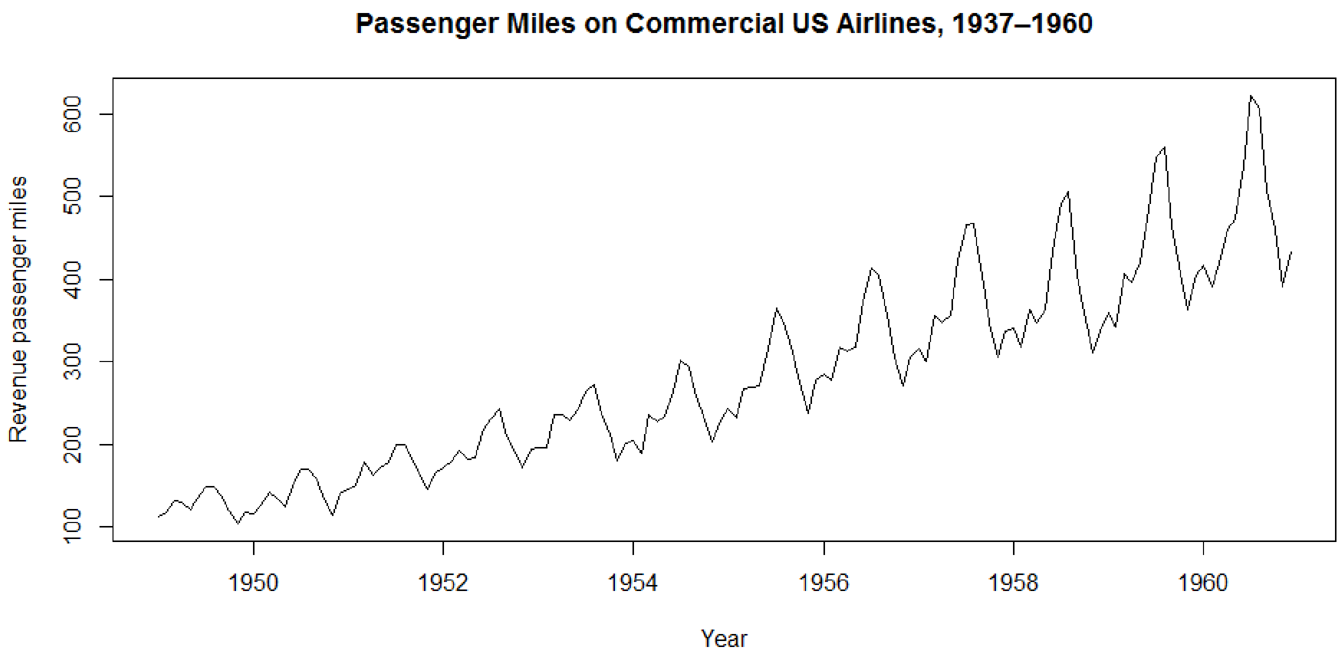
| | A | B | C | D |
|---|---|---|---|---|
| 1 | | change | to | this |
| 2 | 1 | 1 | Name1 | TRUE |
| 3 | 2 | 2 | Name2 | FALSE |
| 4 | 3 | 3 | Name3 | TRUE |
| 5 | 4 | 4 | Name4 | FALSE |
| 6 | 5 | 5 | Name5 | TRUE |
| 7 | | | | |
| 8 | | | | |

```
#Files
read.csv("C:/Users/vangelis spil/Desktop/ft_project_insample_1.csv")
input <- read.csv("C:/Users/vangelis spil/Desktop/ft_project_insample_1.csv", stringsAsFactors = F)
```

## Visualization

```
#Plot
time_series <- AirPassengers
plot(time_series, type="l", main="Passenger Miles on Commercial US Airlines, 1937-1960",
     ylab = "Revenue passenger miles", xlab = "Year")
```

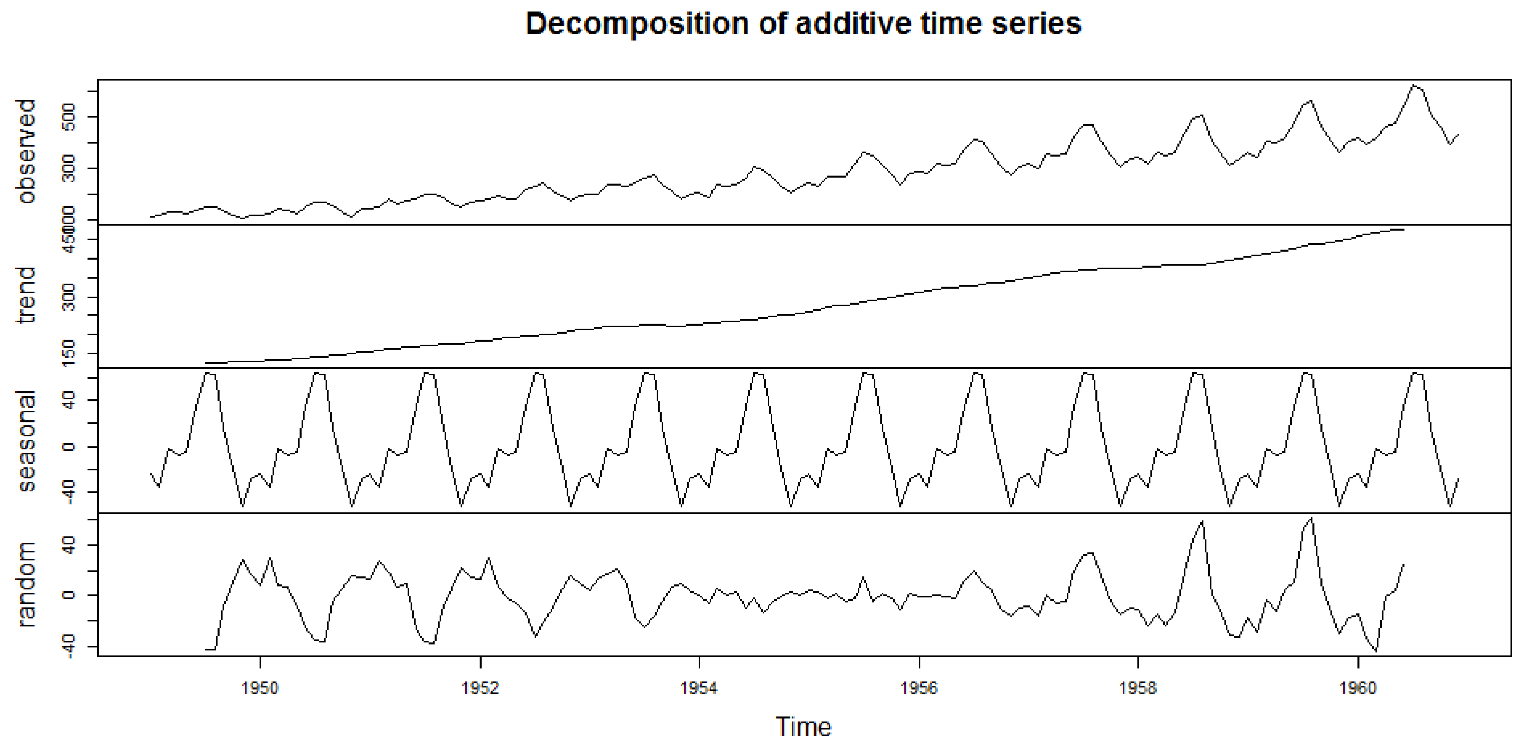
Powerful alternative to basic R plots



# Decomposition

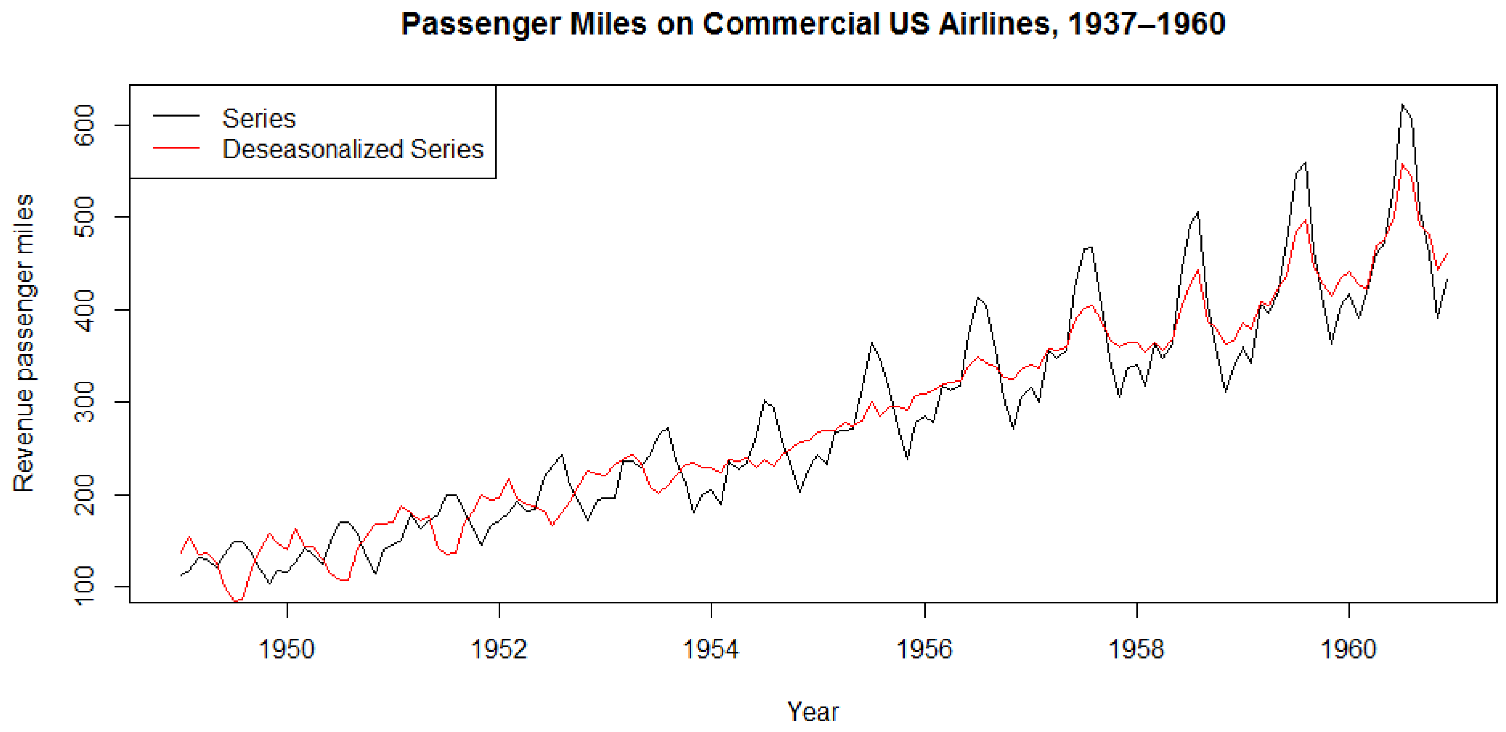
```
#Decompose
dec <- decompose(time_series, type="additive")
plot(dec)
plot(dec$seasonal[1:frequency(time_series)], type="l",
     ylab = "Index", xlab = "Period")
```

**Data = Trend + Seasonal + Random**



# Additive seasonal adjustments

```
#Seasonally adjust
d_time_series <- time_series - dec$seasonal
plot(time_series, type="l", main="Passenger Miles on Commercial US Airlines, 1937-1960",
     ylab = "Revenue passenger miles", xlab = "Year")
lines(d_time_series, col="red")
legend("topleft",
       legend = c("Series", "Deseasonalized Series"),
       col = c("black", "red"), lty=1)
```

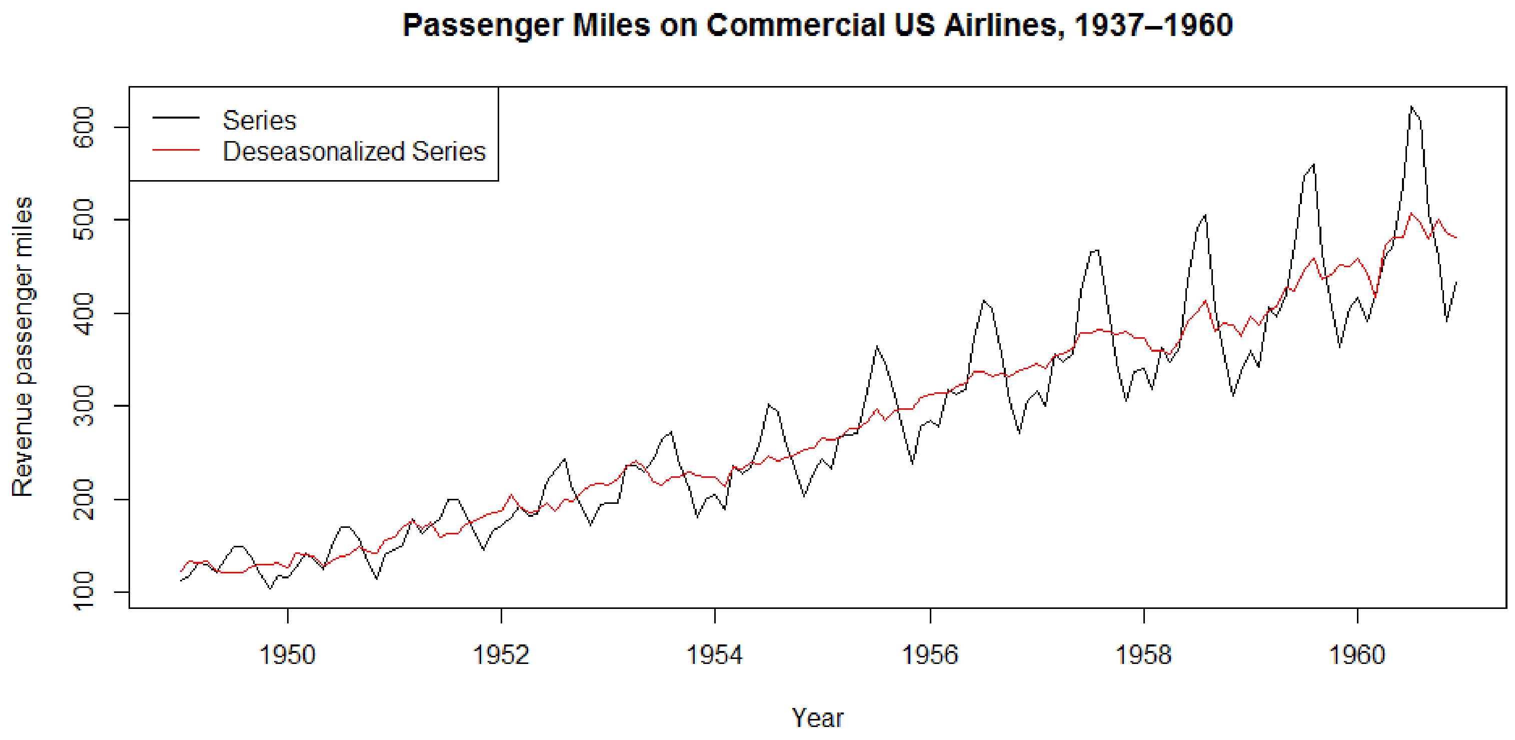


- Seasonal intensity changes over time, being subject to trend **(Heteroscedasticity)**

# Multiplicative seasonal adjustments (1/2)

```
#Seasonally adjust
dec <- decompose(time_series, type="multiplicative")
d_time_series <- time_series / dec$seasonal
plot(time_series, type="l", main="Passenger Miles on Commercial US Airlines, 1937-1960",
     ylab = "Revenue passenger miles", xlab = "Year")
lines(d_time_series, col="red")
legend("topleft",
       legend = c("Series", "Deseasonalized Series"),
       col = c("black", "red"), lty=1)
```

**Data = Trend * Seasonal * Random**

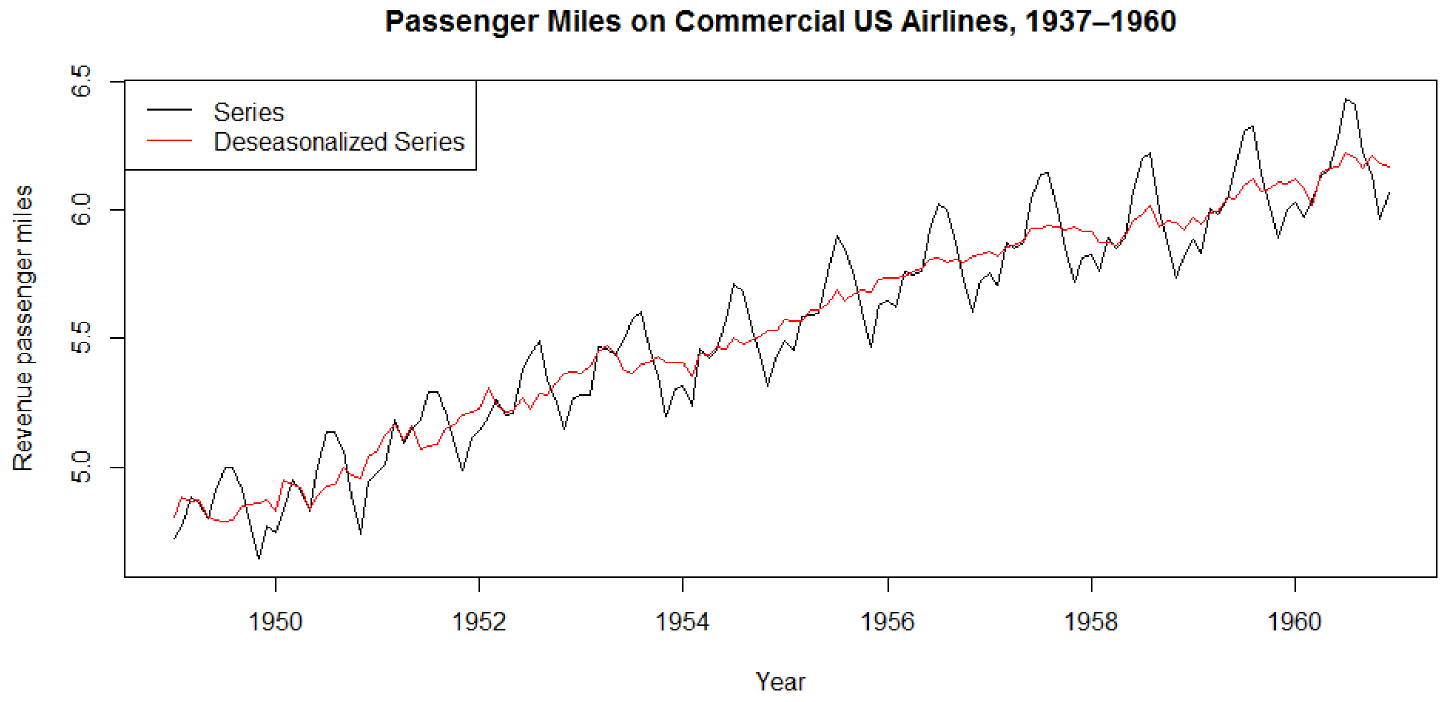


- The series is characterized by **multiplicative** seasonality

# Multiplicative seasonal adjustments (2/2)

```
#Seasonally adjust
log_time_series <- log(time_series)
dec <- decompose(log_time_series, type="additive")
d_time_series <- log_time_series - dec$seasonal
plot(log_time_series, type="l", main="Passenger Miles on Commercial US Airlines, 1937-1960",
     ylab = "Revenue passenger miles", xlab = "Year")
lines(d_time_series, col="red")
legend("topleft",
       legend = c("Series", "Deseasonalized Series"),
       col = c("black", "red"), lty=1)
```

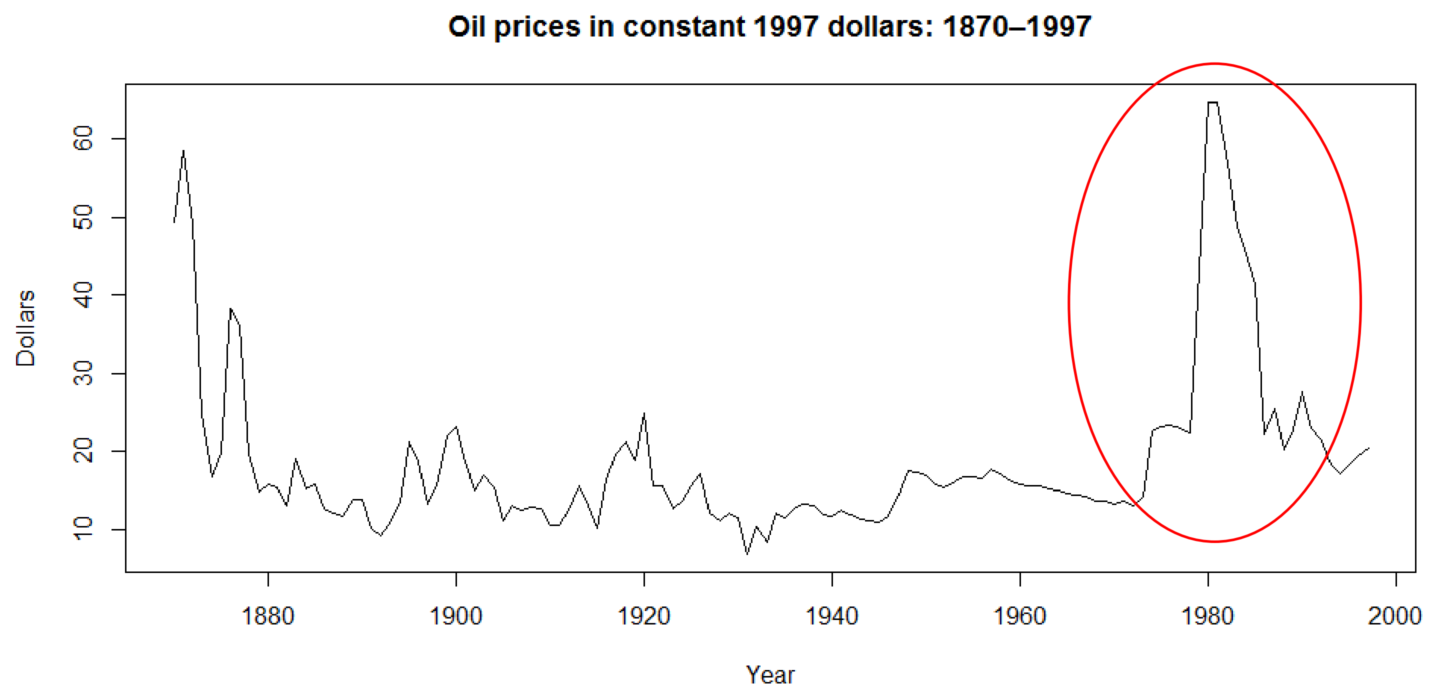
- Alternatively, we can **log-scale** our data and use the **additive** seasonality



# Distribution of data (1/2)

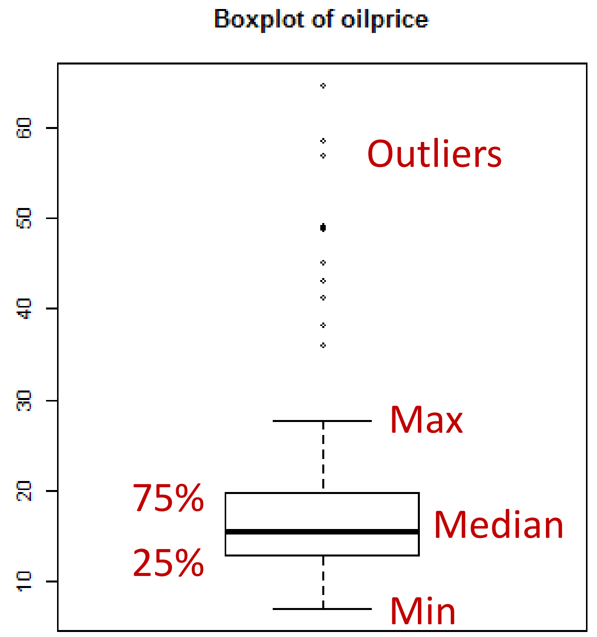
```
library(fpp)
plot(oilprice, type="l", ylab="Dollars", xlab = "Year",
     main="Oil prices in constant 1997 dollars: 1870-1997")
```

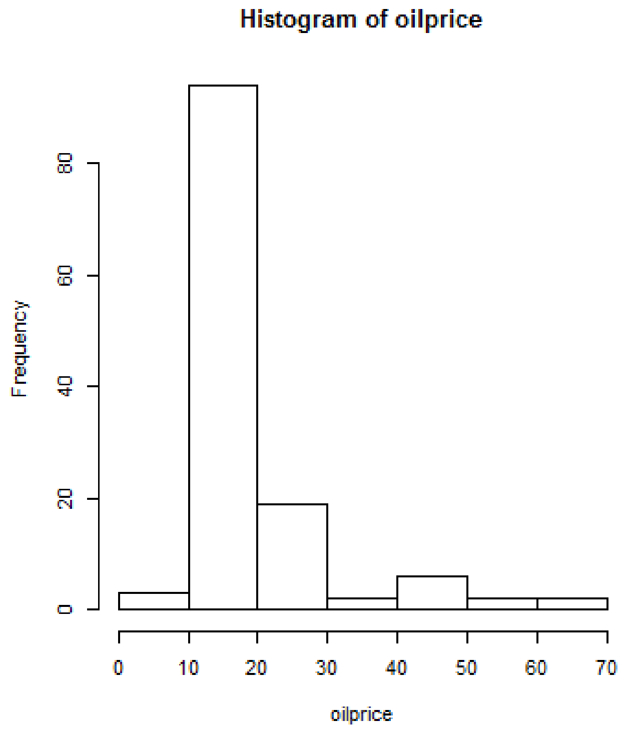
- The 1980s oil glut was a serious surplus of crude oil caused by falling demand following the 1970s energy crisis

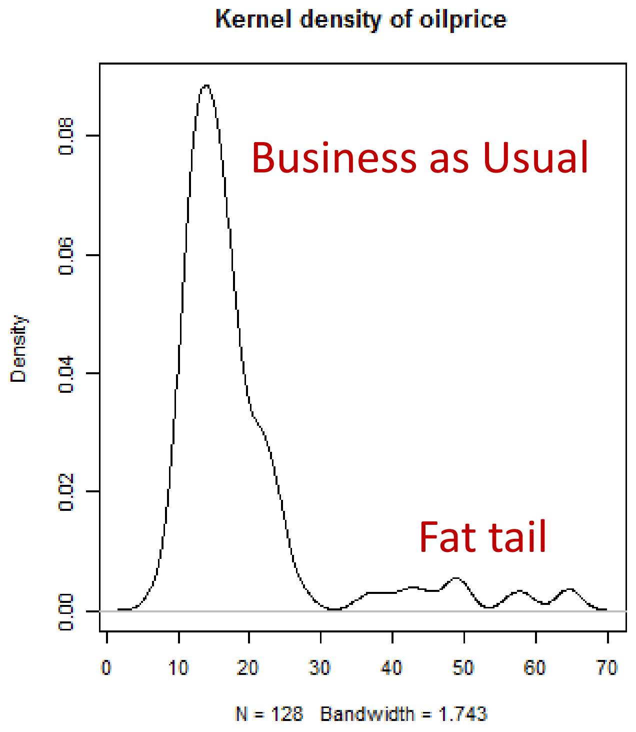


# Distribution of data (2/2)

```
library(fpp)
plot(oilprice, type="l", ylab="Dollars", xlab = "Year",
     main="Oil prices in constant 1997 dollars: 1870-1997")

par(mfrow=c(1,3))
boxplot(oilprice, main="Boxplot of oilprice")
hist(oilprice)
plot(density(oilprice), main="Kernel density of oilprice")
```



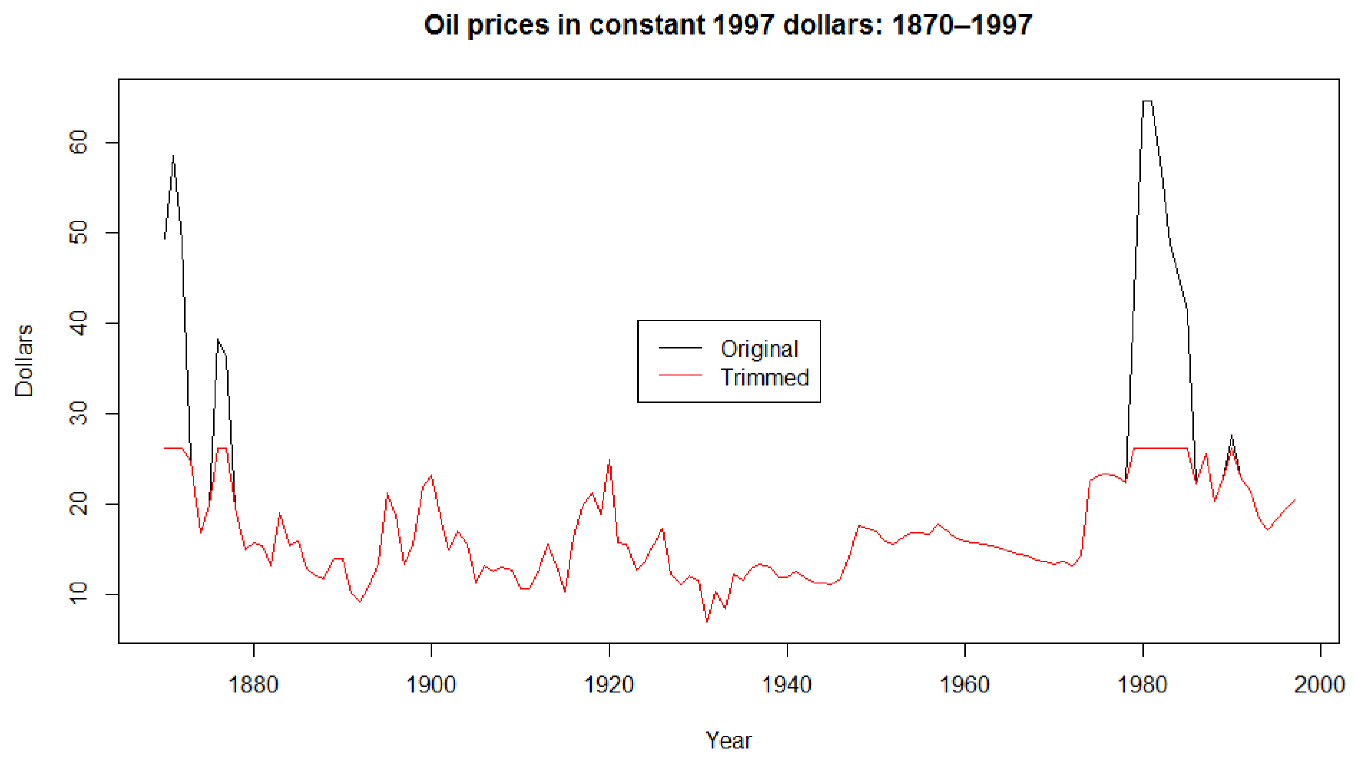




## Deal with outliers – By trimming

```
#Remove outliers
limit <- quantile(oilprice, 0.90)
n_oilprice <- oilprice
n_oilprice[n_oilprice>limit] <- limit
plot(oilprice, type="l", ylab="Dollars", xlab = "Year",
     main="Oil prices in constant 1997 dollars: 1870–1997")
lines(n_oilprice, col="red")
legend("center",
       legend = c("Original", "Trimmed"),
       col = c("black", "red"), lty=1)
```

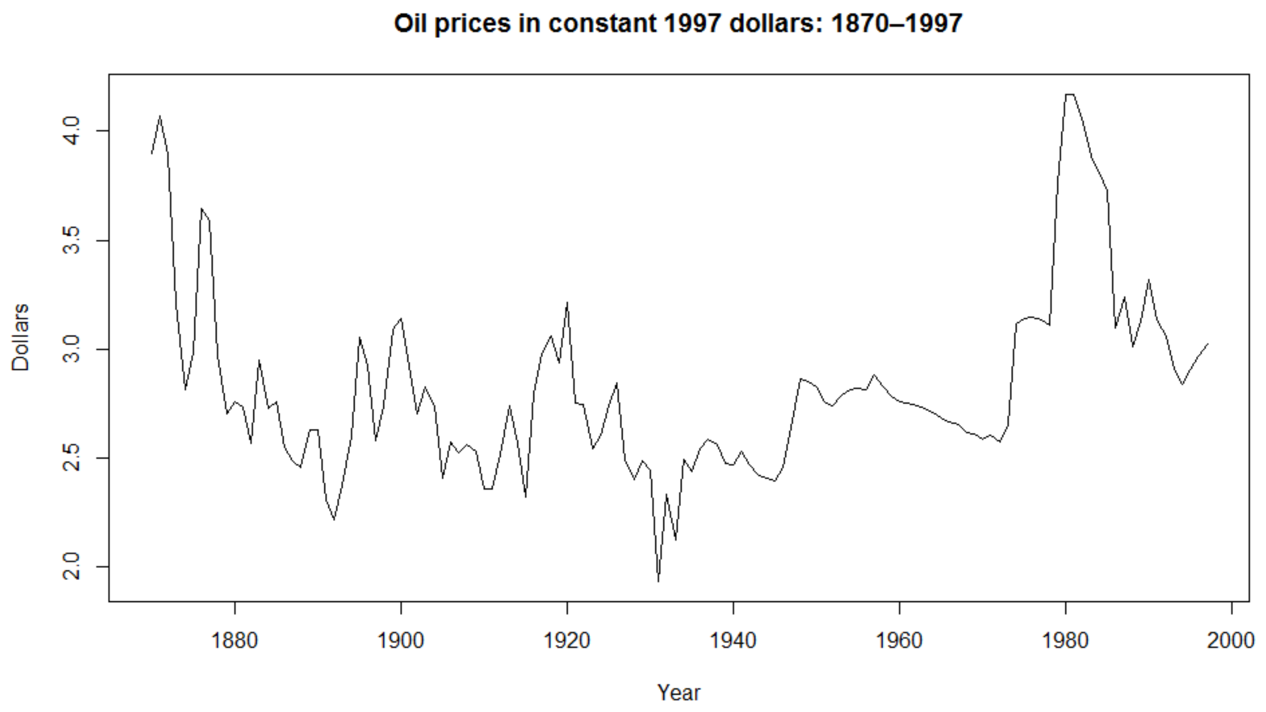
- The limit is **arbitrarily** set to the top 10% of the observed values
- The same can be down for low prices



# Deal with outliers – By reducing variance

```
plot(log(oilprice), type="l", ylab="Dollars", xlab = "Year",
     main="Oil prices in constant 1997 dollars: 1870-1997")

sd(oilprice)*100/mean(oilprice)
sd(log(oilprice))*100/mean(log(oilprice))
```

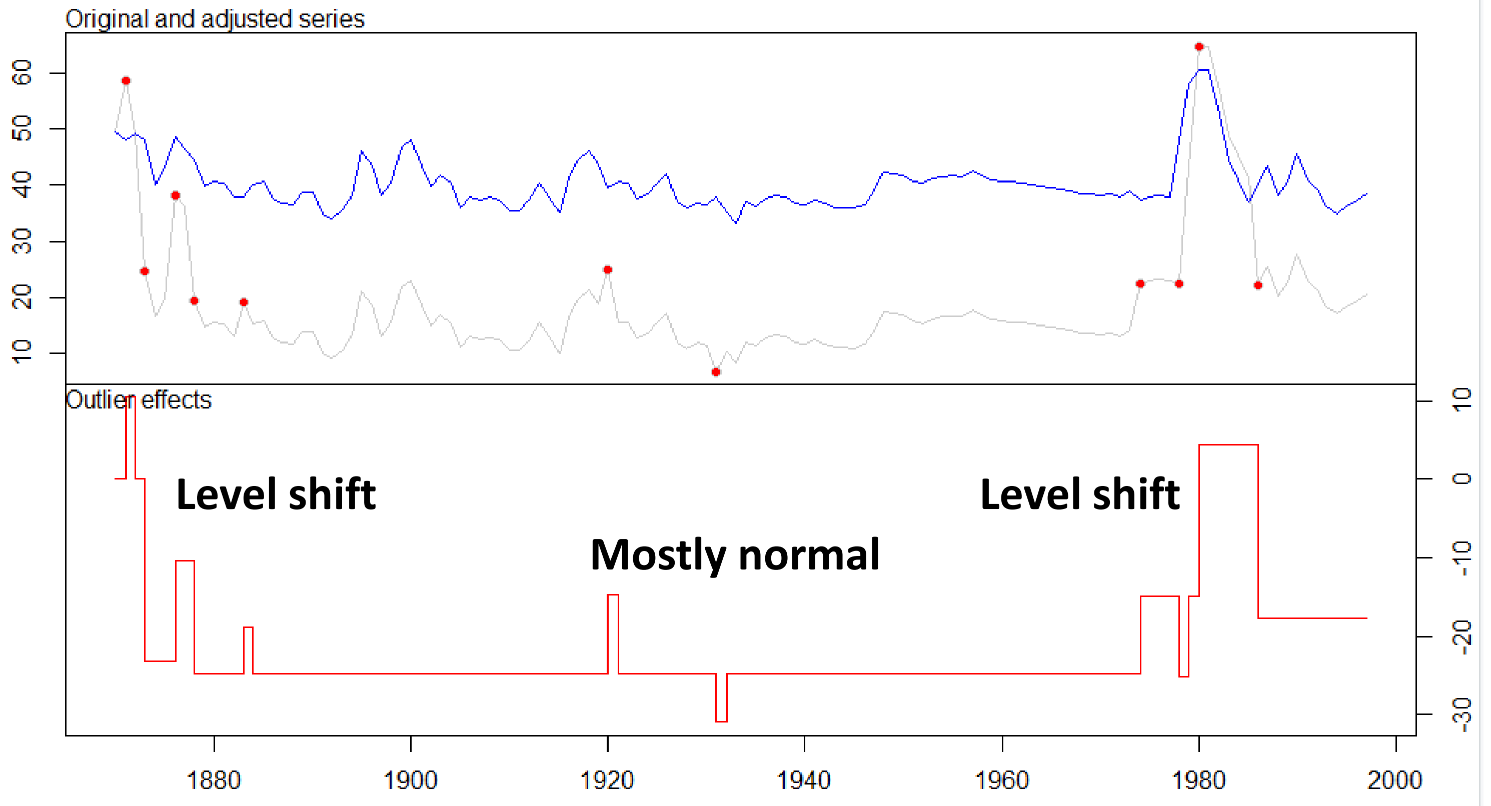
- Although the outliers are still visible, their extent has been significantly reduced
- **Coefficient of Variation (CV)**
  - ✓ Before: 58.65%
  - ✓ After: 15.05%



# Deal with outliers – By fitting a forecasting model

```
library(tsoutliers)
product.outlier<-tso(oilprice,types=c("AO","LS"))
plot(product.outlier)
```

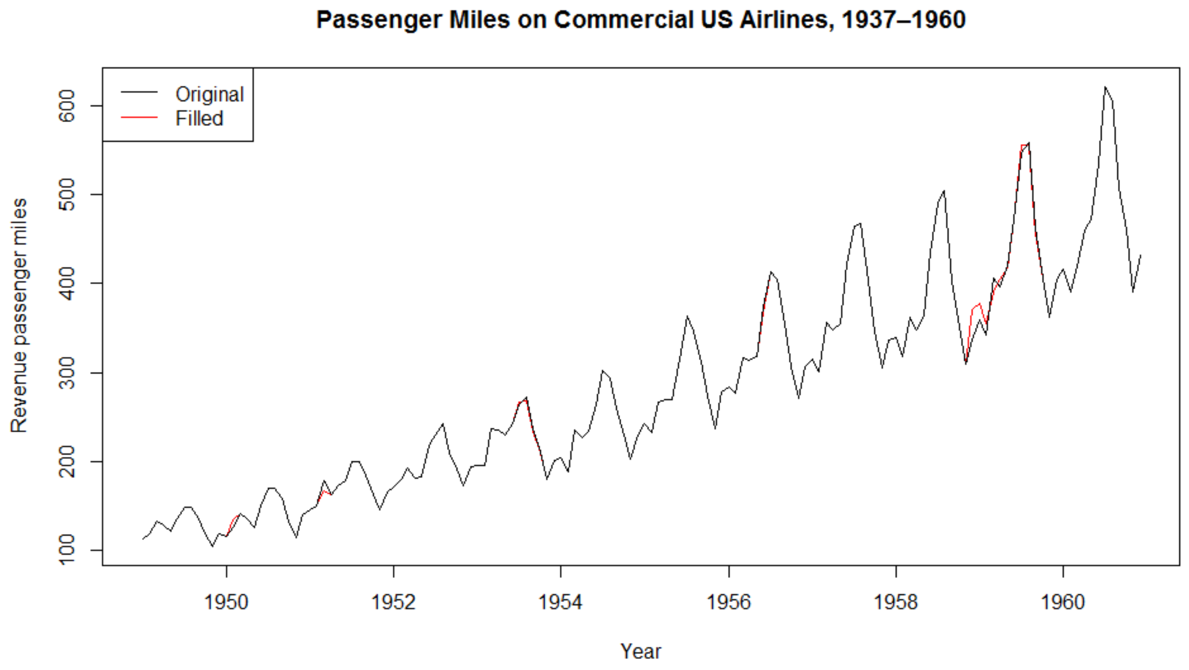
```
Outliers:
   type ind time coefhat   tstat
1    AO   2 1871  10.430   5.534
2    LS   4 1873 -23.272  -8.297
3    LS   7 1876  12.928   5.218
4    LS   9 1878 -14.492  -5.598
5    AO  14 1883   5.931   5.099
6    AO  51 1920  10.110   7.190
7    AO  62 1931  -6.220  -4.451
8    LS 105 1974   9.971   5.112
9    AO 109 1978 -10.458  -7.448
10   LS 111 1980  19.207   8.069
11   LS 117 1986 -22.165 -10.652
```



## Deal with missing values

```
#Fill missing values
missing <- AirPassengers
missing[c(14,27,55,56,57,58,90,120,121,122,
         123,124,125,127,128,129)] <- NA

for (i in (frequency(missing)+1):(length(missing)-12)){
  if (is.na(missing[i])==TRUE){
    missing[i] <- mean(c(missing[i-frequency(missing)], missing[i+frequency(missing)]))
  }
}

plot(missing, type="l", main="Passenger Miles on Commercial US Airlines, 1937-1960",
     ylab = "Revenue passenger miles", xlab = "Year", col="red")
lines(AirPassengers, col="black")
legend("topleft",
       legend = c("Original", "Filled"),
       col = c("black", "red"), lty=1)
```

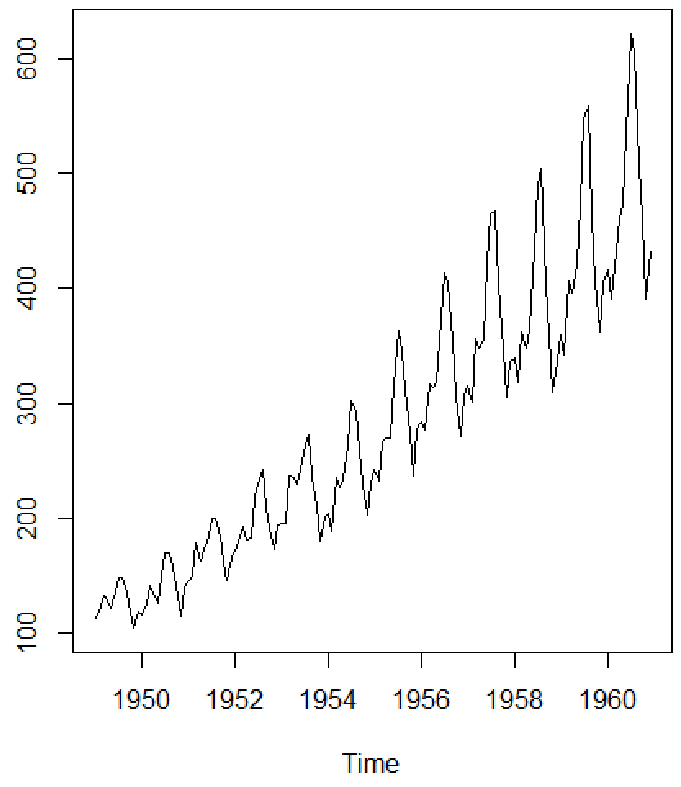
- **Non-seasonal:** Average of the previous and the following observations
- **Seasonal & non-trended**: Average of all the observations of the same period
- **Seasonal & trended**: Average of the previous and following observations of the same period

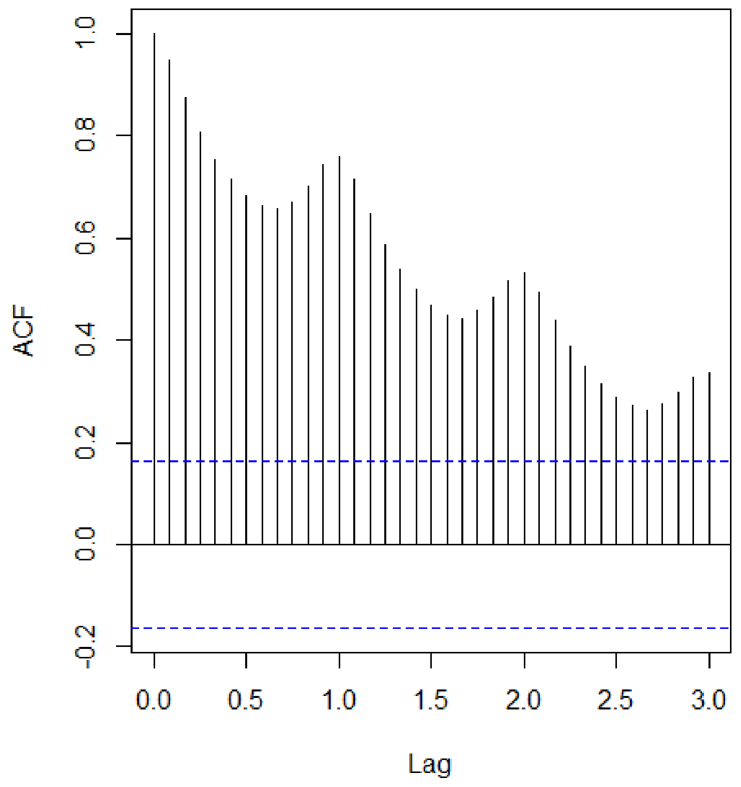


# Autocorrelation (1/4)

```
#Correlation between observations
par(mfrow=c(1,2))

plot(time_series)
acf(time_series,lag.max=36)
```

- Correlation decreases through time -> **Trend**
- Correlation oscillates every 12 periods -> **Seasonality**

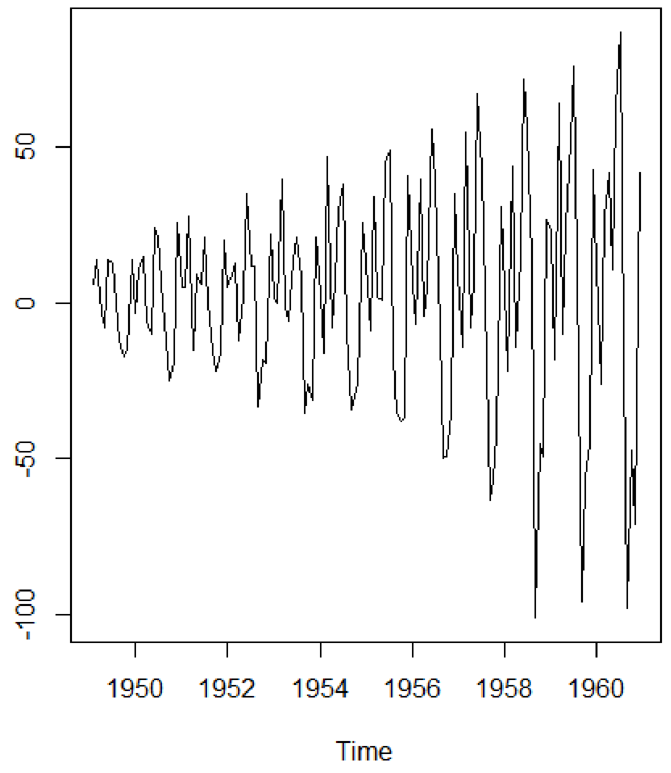


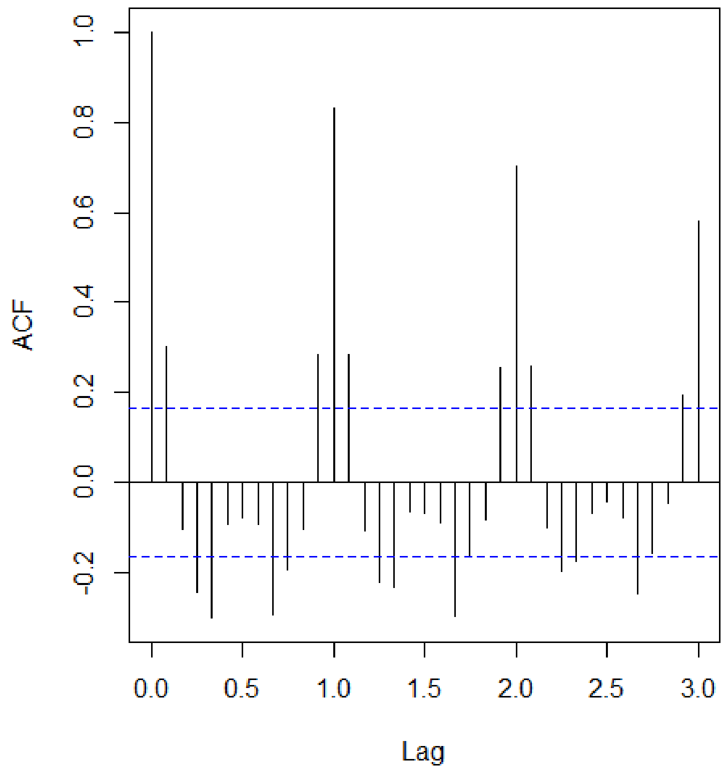


# Autocorrelation (2/4)

```
plot(diff(time_series,1))
acf(diff(time_series,1),lag.max=36)
```

- **First differences:** Trend has been removed– Seasonality is still strong

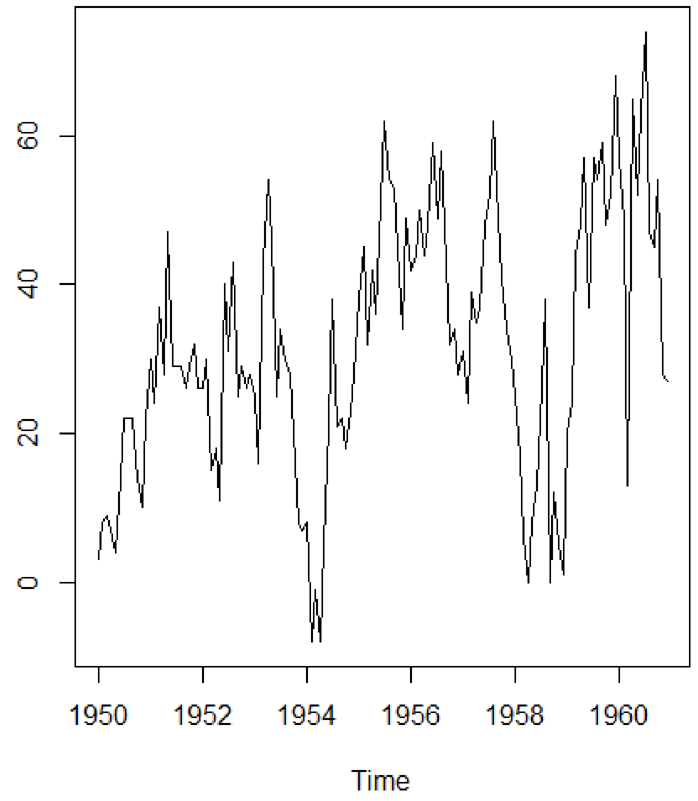


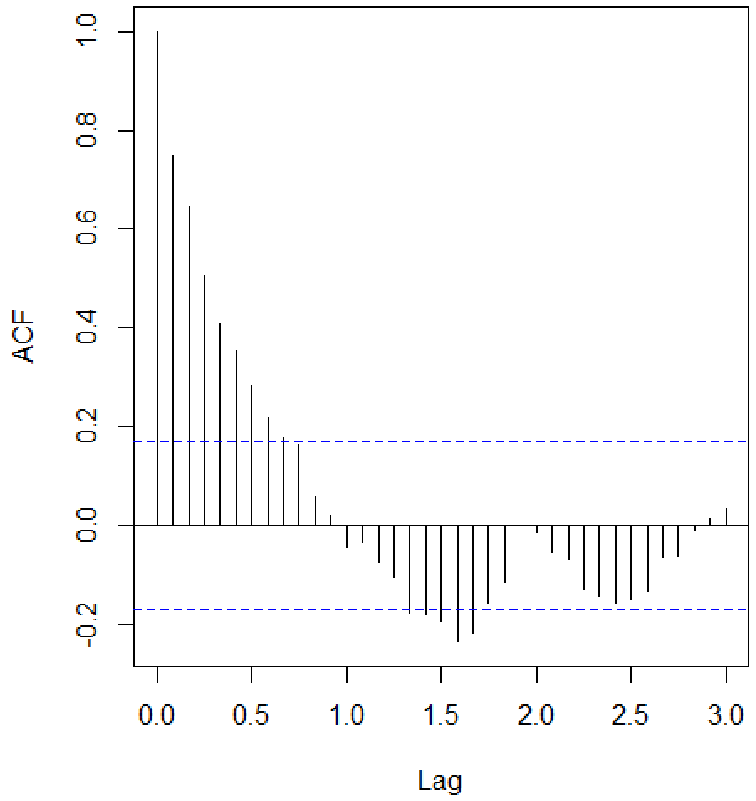


## Autocorrelation (3/4)

```
plot(diff(time_series,12))
acf(diff(time_series,12),lag.max=36)
```

- **Seasonal differences:** Seasonality has been removed– Trend is still observable

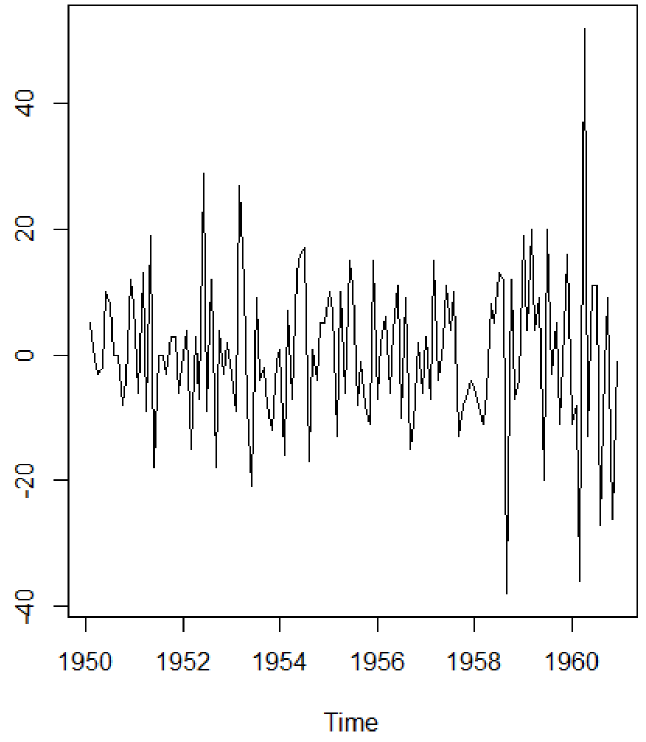


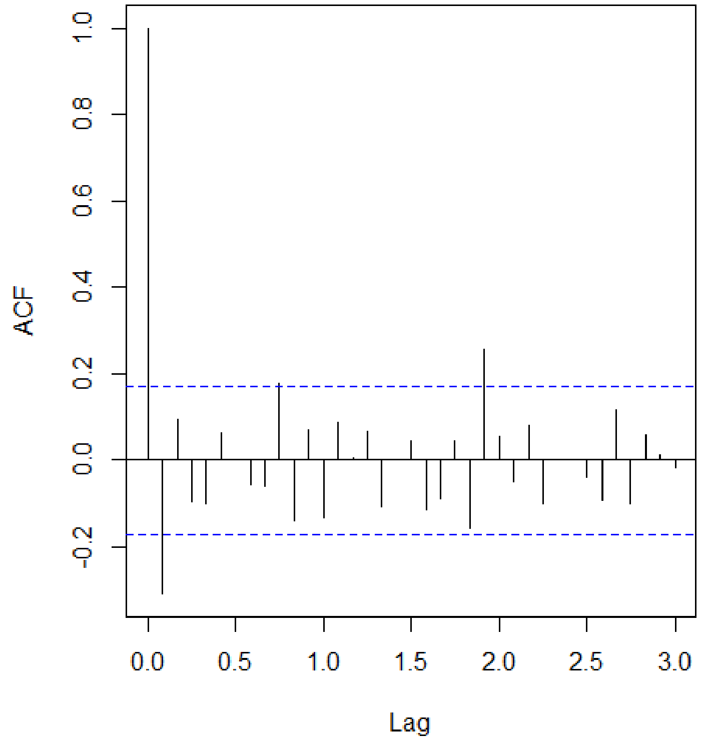


# Autocorrelation (4/4)

```
plot(diff(diff(time_series,12),1))
acf(diff(diff(time_series,12),1),lag.max=36)
```

- **First and Seasonal differences:** We get a stationary series (mean and deviation constant through time)





- Differentiation is another way for **decomposing** a series
- Useful for making time series data ready-to-be used by **ML forecasting methods** (*typically assume stationarity*)