IntRoduction to R

Forecasting Techniques 2017-2018

zabbeta@fsu.gr | katerina@fsu.gr

- ✓ Starting out in R
- ✓ Working with data
- ✓ Plotting & Forecasting

1. Starting Out In R

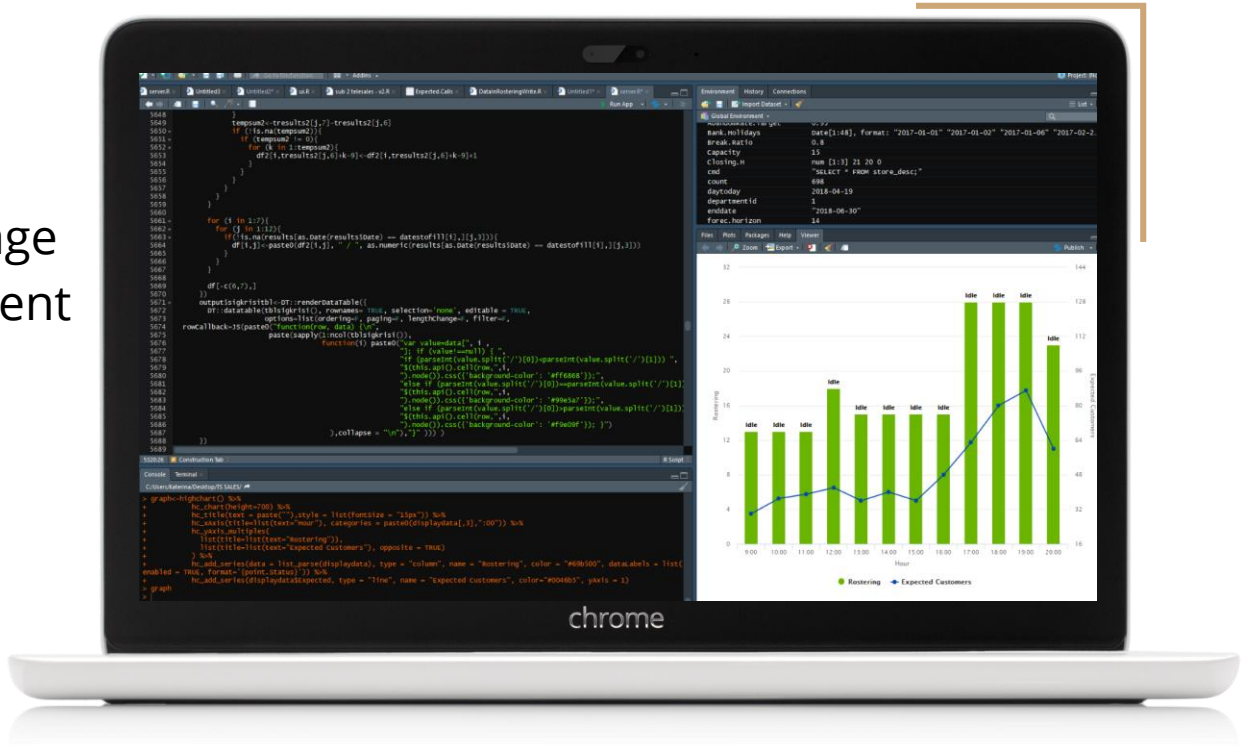
- ✓ R & RStudio
- ✓ Variables & Basics
- ✓ Data Types
- ✓ Functions

R + RStudio

- ✓ Programming language
- ✓ Interactive environment for statistics

RStudio :

- ✓ Text Editor
- ✓ Commands History
- ✓ Environment
- ✓ Plot Panel
- ✓ Help Panel



Variables & Basics

- ✓ Click on the “Console” panel, type 1+1 and press enter and R will display the result
- ✓ Create a new variable by assigning a value to it using <-
- ✓ Check variable’s value in the environment panel or by typing the name of it
- ✓ Comment using #

Basic Operations

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

2 * 2 == 4
## [1] TRUE
```

Variables

```
weight_kg <- 55
weight_kg
## [1] 55

# Weight in pounds
weight_lb <- 2.2 * weight_kg
weight_lb
## [1] 121
```

Data Types (1/4)

Vector:

- ✓ Vectors contain zero or more elements, all of the same basic type
- ✓ Elements can be named
- ✓ Indexing starts from 1

Types:

- ✓ Numeric
- ✓ Character
- ✓ Logical

Basic Operations (1/4)

```
myvec <- c(10,20,30,40,50)
myvec + 1
## [1] 11 21 31 41 51

myvec + myvec
## [1] 20 40 60 80 100

length(myvec)
## [1] 5

c(60, myvec)
## [1] 60 10 20 30 40 50

c(myvec, myvec)
## [1] 10 20 30 40 50 10 20 30 40 50
```


Data Types (1/4)

Vector:

- ✓ Vectors contain zero or more elements, all of the same basic type
- ✓ Elements can be named
- ✓ Indexing starts from 1

Types:

- ✓ Numeric
- ✓ Character
- ✓ Logical

Basic Operations (2/4)

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

myvec[2]
## [1] 20

myvec[2] <- 5
myvec
## [1] 10 5 30 40 50

#Can we use a vector to index another
#vector? Yes!
myind <- c(4,3,2)
myvec[myind]
## [1] 40 30 5
```

Data Types (1/4)

Vector:

- ✓ Vectors contain zero or more elements, all of the same basic type
- ✓ Elements can be named
- ✓ Indexing starts from 1

Types:

- ✓ Numeric
- ✓ Character
- ✓ Logical

Basic Operations (3/4)

```
#Sometimes we want a continuous
#slice from a vector
myvec[3:5]
## [1] 30 40 50

#We can take slices of character vectors
#as well
phrase <- c("I", "don't", "know", "I",
"know")
# First three words
phrase[1:3]
## [1] "I" "don't" "know"

# Last three words
phrase[3:5]
## [1] "know" "I" "know"
```

Data Types (2/4)

List:

- ✓ Lists contain zero or more elements, of any type
- ✓ List elements can and typically do have names
- ✓ Access an element:
  - `Mylist[[5]]`
  - `Mylist[["elementname"]]`
  - `Mylist$elementname`
- ✓ Creation:

```
list(a=1, b="two", c=FALSE)
```

Basic Operations

```
n = c(2, 3, 5)
s = c("aa", "bb", "cc", "dd", "ee")
b = c(TRUE, FALSE, TRUE, FALSE, FALSE)
x = list(n, s, b, 3)

x[2]
[[1]]
[1] "aa" "bb" "cc" "dd" "ee"

x[c(2, 4)]
[[1]]
[1] "aa" "bb" "cc" "dd" "ee"
[[2]]
[1] 3

x[[2]]
[1] "aa" "bb" "cc" "dd" "ee"
```


Data Types (3/4)

Matrix:

- ✓ 2-D tabular data structure in which all the elements are of the same type
- ✓ Matrix rows and columns may have names
- ✓ Access an element:
  - `Mat[3,5]`
  - `mat["arowname","acolumnname"]`
- ✓ Creation: `matrix( )`
- ✓ Casting: `as.matrix( )`

Basic Operations

```
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 # print the matrix
     [,1] [,2] [,3]
[1,]    2    4    3
[2,]    1    5    7

A[2, 3] # element at 2nd row, 3rd column
[1] 7

A[ ,3] # the 3rd column
[1] 3 7
```


Data Types (4/4)

Data Frame:

- ✓ 2-D tabular data structure in which the columns may have different types, but all the elements in each column must have the same type
- ✓ Accessing elements is the same as for matrices
- ✓ Creation: `data.frame(colname1=values1,colname2=values2,...)`
- ✓ Casting: `as.data.frame( )`

Basic Operations

```
n = c(2, 3, 5)
s = c("aa", "bb", "cc")
b = c(TRUE, FALSE, TRUE)
df = data.frame(n, s, b) # df is a data frame
```


Functions

- ✓ R has various built-in functions
- ✓ We can get help using ?function_name
- ✓ Creation:

```
my_function<-function(var1, var2)
{
     var3 <- var1 + var2
}
```

Basic Operations

```
#R has various functions
#such as sum( )
sum(myvec)
## [1] 135

#Let's look at the function rep
rep(42, 10)
## [1] 42 42 42 42 42 42 42 42 42 42

rep(c(1,2,3), 10)
## [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1
2 3 1 2 3 1 2 3 1 2 3 1 2 3

rep(c(1,2,3), each=3, times=5)
## [1] 1 1 1 2 2 2 3 3 3 1 1 1 2 2 2 3
3 3 1 1 1 2 2 2 3 3 3 1 1 1 2 2 2 3 3
## [36] 3 1 1 1 2 2 2 3 3 3
```

2. Working with Data

- ✓ Read Datasets
- ✓ Handle Data
- ✓ Export Datasets

Read Datasets

- ✓ Load Data from file using `read.csv()`
- ✓ read.csv( ) has many arguments that you may find useful
- ✓ read.csv( ) loads the data as a data frame

Basic Operations

```
dat <- read.csv("path/to/ts-1.csv", sep=",")
head(dat, 5)
  Year  Period  Original.Data
1 2005       1          19258
2 2005       2          19258
3 2005       3          19308
4 2005       4          19499
5 2005       5          19591

mean(dat$Original.Data) #get mean value
[1] 20518

#keep only the data points that are greater than mean
test<-dat[dat$Original.Data > mean(dat$Original.Data),]
head(test, 5)
   Year Period Original.Data
16 2006      4         20758
19 2006      7         20522
26 2007      2         20558
31 2007      7         20564
33 2007      9         20984
```


HandleData

- ✓ Load Data from file using `read.csv()`
- ✓ read.csv( ) has many arguments that you may find useful
- ✓ read.csv( ) loads the data as a data frame

Basic Operations

```
dat$newcol <- dat$Original.Data >
mean(dat$Original.Data)
head(dat, 5)
  Year Period Original.Data newcol
1 2005      1          19258  FALSE
2 2005      2          19258  FALSE
3 2005      3          19308  FALSE
4 2005      4          19499  FALSE
5 2005      5          19591  FALSE

#export new data frame
write.csv(dat, "path/to/ts-1-new.csv", row.names = F)
```

3. Plotting & Forecasting

- ✓ Plot Types
- ✓ Plotting Functions
- ✓ ggplot2
- ✓ Forecast Package

"The purpose of computing is insight, not numbers"
Richard Hamming

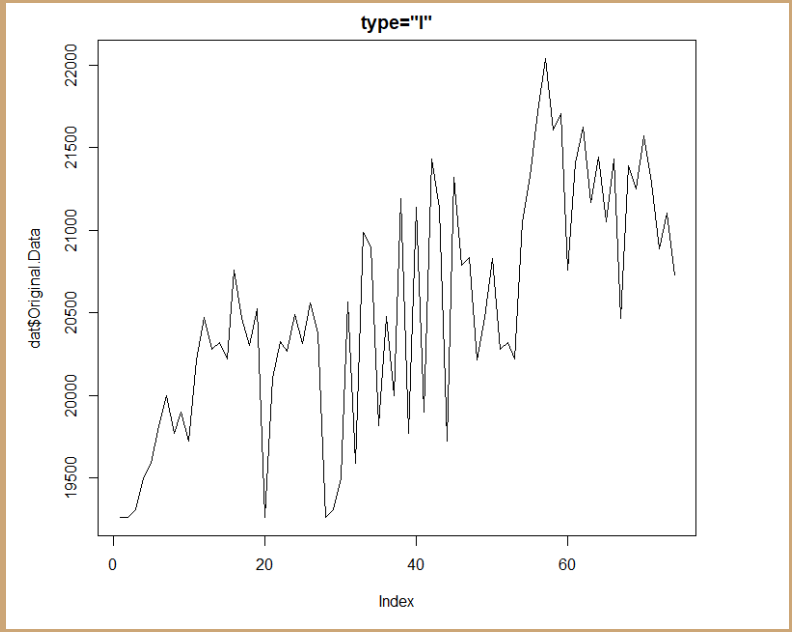
Plotting Functions

plot( ):

- ✓ “p”: Points
- ✓ “l”: Lines
- ✓ “b”: Both

Basic R plots (1/3)

```
plot(dat$Original.Data, type="l", main='type=l"')
```



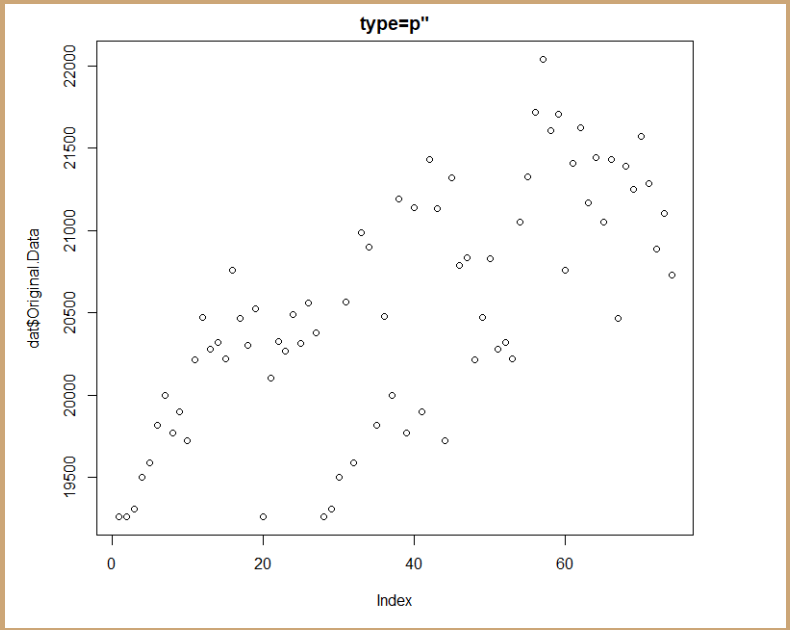
Plotting Functions

plot( ):

- ✓ “p”: Points
- ✓ “l”: Lines
- ✓ “b”: Both

Basic R plots (2/3)

```
plot(dat$Original.Data, type="p", main='type="p"')
```



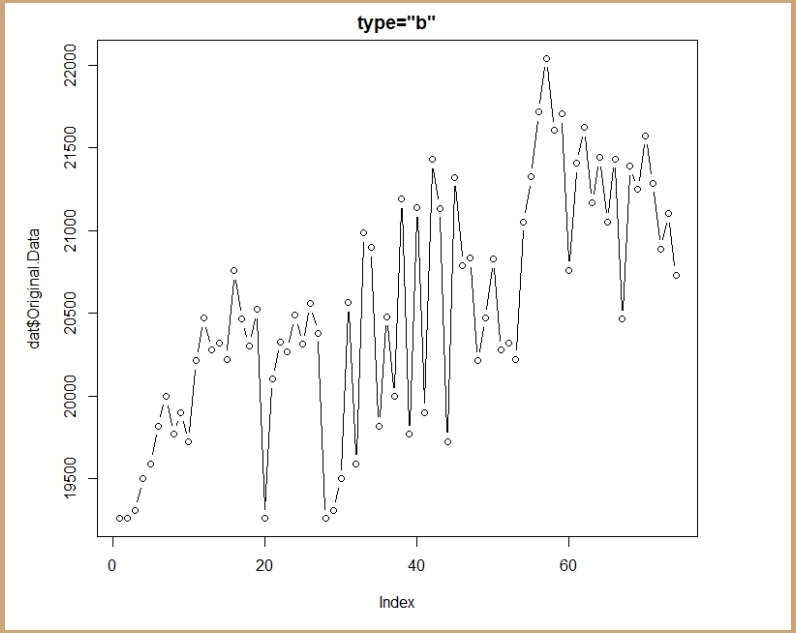
Plotting Functions

plot( ):

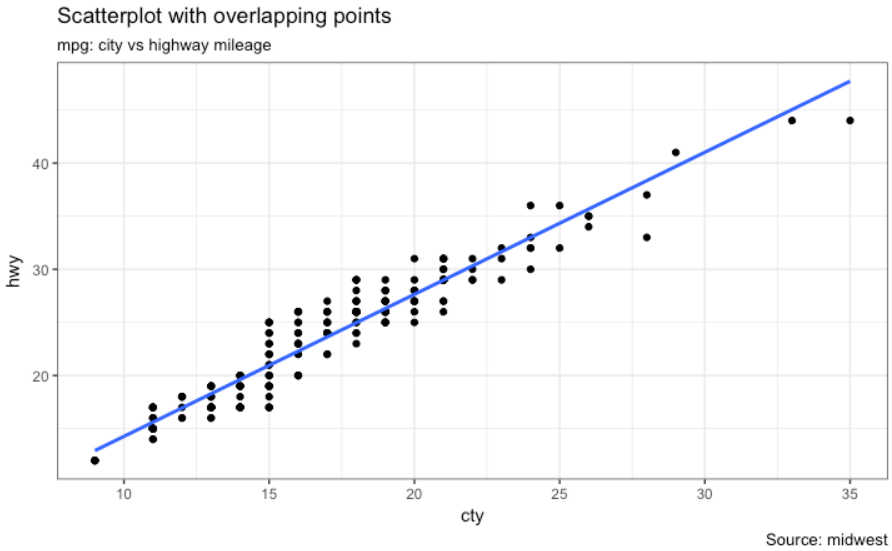
- ✓ “p”: Points
- ✓ “l”: Lines
- ✓ “b”: Both

Basic R plots (3/3)

```
plot(dat$Original.Data, type="b", main='type="b"')
```



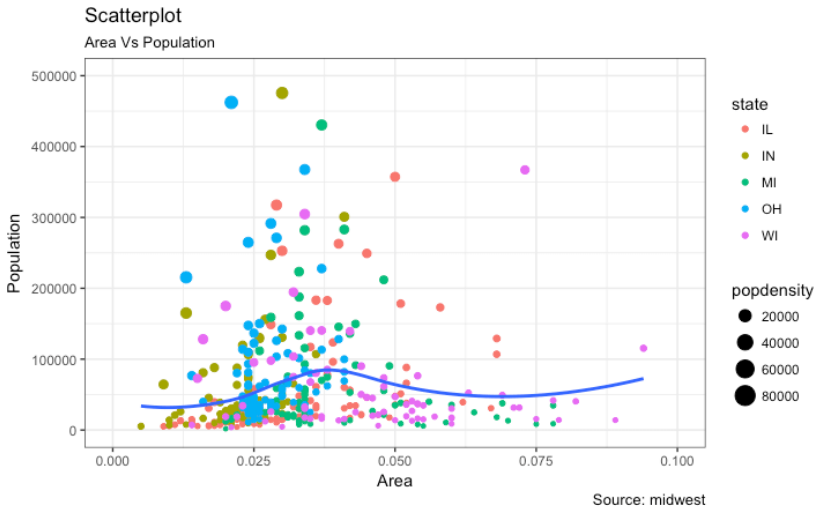
ggplot2 (1/2)



ggplot2

```
# load package and data
library(ggplot2)
data(mpg, package="ggplot2")

# alternate source: "http://goo.gl/uEeRGu")
theme_set(theme_bw())  # pre-set the bw theme
g <- ggplot(mpg, aes(cty, hwy))

# Scatterplot
g + geom_point() +
  geom_smooth(method="lm", se=F) +
  labs(subtitle="mpg: city vs highway mileage",
       y="hwy",
       x="cty",
       title="Scatterplot with overlapping points",
       caption="Source: midwest")
```


ggplot2 (2/2)



ggplot2

```
# load package and data
options(scipen=999)
library(ggplot2)
theme_set(theme_bw())  # pre-set the bw theme.
data("midwest", package = "ggplot2")

# Scatterplot
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) +
  xlim(c(0, 0.1)) +
  ylim(c(0, 500000)) +
  labs(subtitle="Area Vs Population",
       y="Population",
       x="Area",
       title="Scatterplot",
       caption = "Source: midwest")

plot(gg)
```


Packages

Getting Help with R:

- Search for terms
  help.search("forecasting")

- Detailed help
  help(forecast)

forecast

```
install.packages("fpp", dependencies=TRUE)
install.packages("forecast", dependencies=TRUE)
install.packages(“Matrix", dependencies=TRUE)
install.packages(“tseries", dependencies=TRUE)
install.packages(“ggplot2", dependencies=TRUE)
```

```
library(fpp)
library(forecast)
library(Matrix)
library(tseries)
Library(ggplot2)
```

Insert your Data

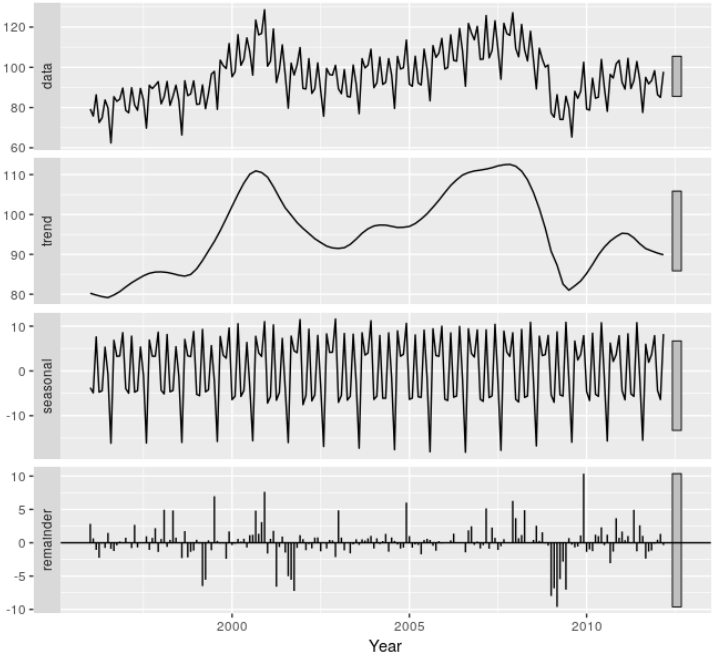
plot(x, type="l", main=example, col = "blue")

plot(xts.naive$mean, type="l", main=example, col = "blue")

Rscript

```
#make a new vector

x<-     c(7437115,  7483934,  7551117,
          7647675,  7743831,  7824909,
          7912398,  7996861)


read.table(file, sep = "", skip = 0)

read.csv(file, header = TRUE, sep = ",", dec = ".", fill = TRUE)

xts<-ts(x,frequency = 3) #more at help
```

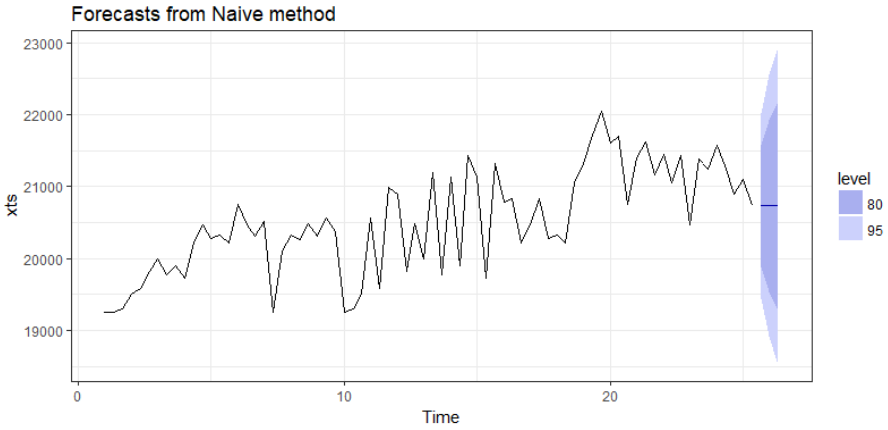
Decomposition



R script

```
#decomposition of xts
xts.decomp<-decompose(xts, type =
c("multiplicative") )

#take seasonal indices
xts.decomp$seasonal

#Take the deseasonilised ts and seasonalise
xts.deseason<-
seasadj(decompose(xts,"multiplicative"))

xts.seasonalindices<-xts.decomp$seasonal[(1:8)]

xts.deseason*(xts.seasonalindices)
```

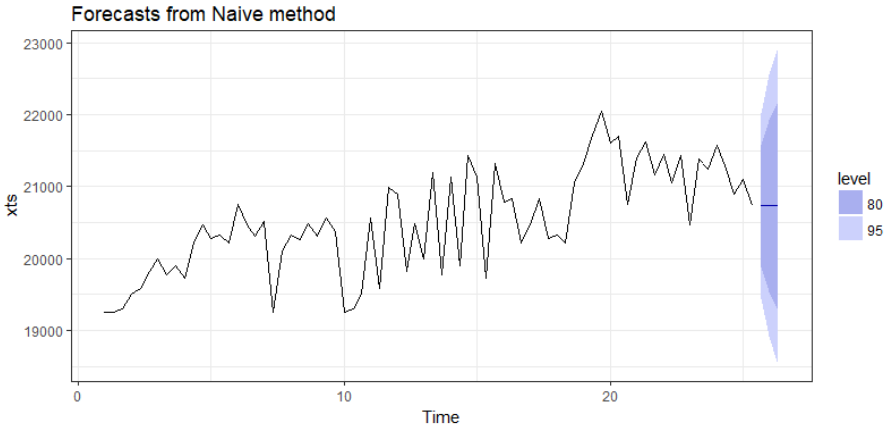
Forecast



Forecast - NAIVE

```
# load package
library(tseries)
library(forecast)

x<-dat$Original.Data
xts.naive<-naive(xts, h=3)
xts.naive$mean
plot(xts.naive)
autoplot(xts.naive)

summary(xts.naive)
```

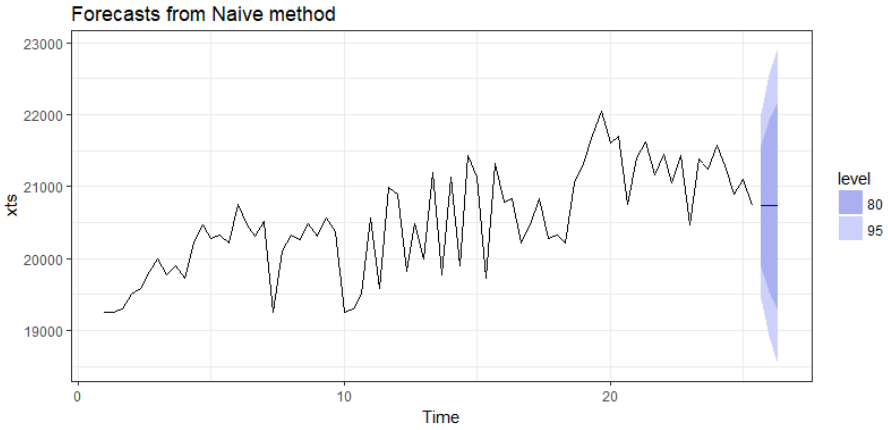

Forecast



Forecast - DAMPED

```
# load package
#damped
xts.damped<-holt(xts, h=3, level=c(80,95),
initial=c("optimal","simple"),damped=TRUE, alpha=NULL,
beta=NULL)
xts.damped$mean
xts.damped$residuals
xts.damped$fitted
```

Forecast



Forecast - LRL

```
forecasingHorizon<-3

aaperiod=c(1: length(xts))  #dianusma time (1, 2, 3,4,5,...,261)

aaperiodforecast=c((length(xts)+1):(length(xts)+forecasingHorizon))  #dianusma provlepsewn

lrlresults <- lm(formula = xts ~ aaperiod)  #ti mpainei sto arg formula tis lm

a=lrlresults$coefficients[1]
b=lrlresults$coefficients[2]

Insamplelrl<- a+b*aaperiod
Forecastlrl<- a+b*aaperiodforecast  #it is just a numeric vector mothing more
```

Keep your forecasts

EXPORT DATA

```
#make a table to export

forecasttable<-matrix(data=NA, nrow=5, ncol=3)
forecasttable[1,] <- xts.naive$mean
forecasttable[2,] <- xts.ses$mean
forecasttable[3,] <- xts.holt$mean
forecasttable[4,] <- xts.damped$mean
forecasttable[5,] <- Forecastlrl

write.table(forecasttable, file =
"C:/Users/zabbeta/Desktop/seminaR0.csv", append =
FALSE, quote = TRUE, sep = ",", row.names = TRUE,
col.names = TRUE,dec = ".")
```

Thank you!

Business Forecasting: Methods and Applications 2018-2019

zabbeta@fsu.gr | katerina@fsu.gr