Introduction to R

Professor Hiroki Sayama & Dieudonne Ouedraogo

3/2/2017

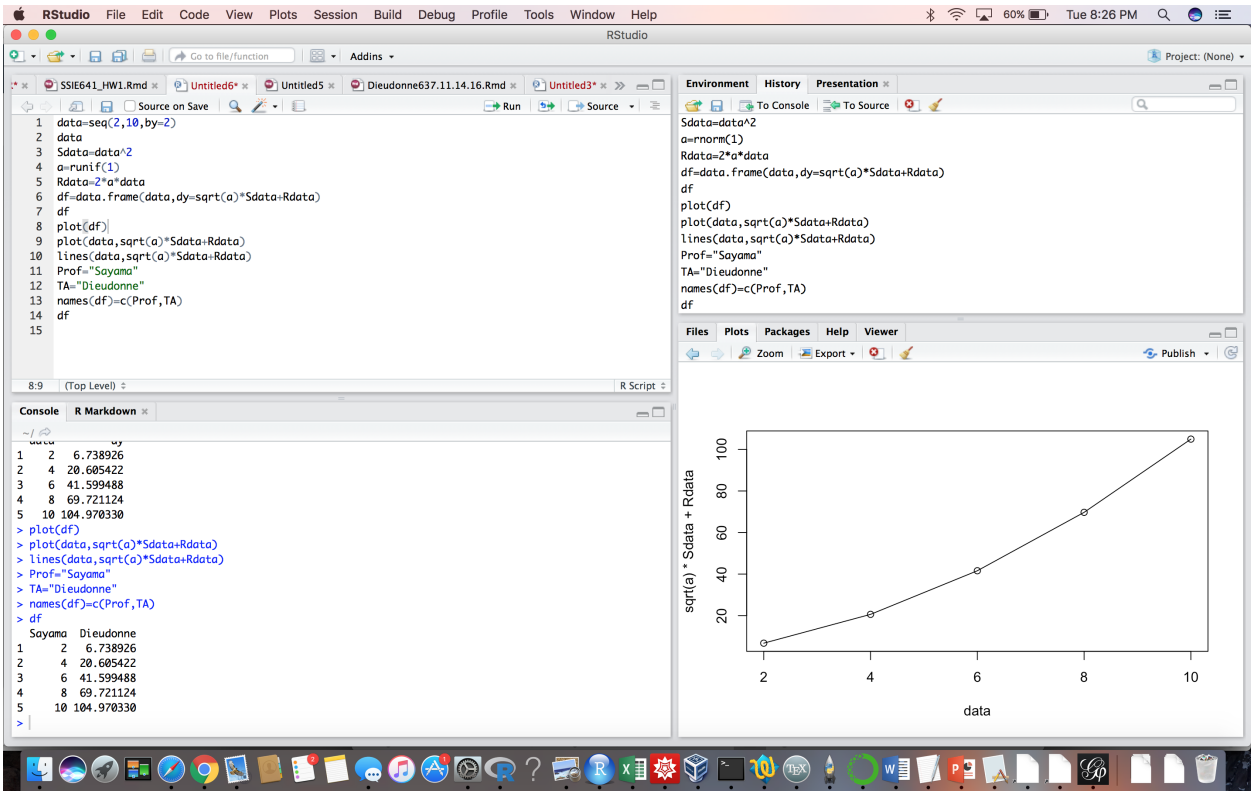


Figure 1: Basic Commands

(1) Introduction and Installation

R is a powerful language mainly used for data and statistical analysis and for graphics.

To install R on your computer go to the home website of R

https://www.r-project.org

Install RStudio

http://www.rstudio.org/

Launch Rstudio

The RStudio consists of several windows

Bottom left: Command Window Here you can type simple commands after the ">" prompt and R will then execute your command. This is the most important window, because this is where R actually does stuff.

Top left: Script or Editor Window
You can type edit and save your code's script here. Just typing a command in the editor window is not enough, it has to get into the command window before R executes the command. If you want to run a line from the script window (or the whole script), you can click Run or press CTRL+ENTER to send it to the command window.

Top right: Workspace and history window. In the workspace window you can see which data and values R has in its memory. You can view and edit the values by clicking on them. The history window shows what has been typed before.

Bottom right: Files,Packages ,Help and Plots packages window. Here you can open files, view plots (also previous plots), install and load packages or use the help function. You can change the size of the windows by drag- ging the grey bars between the windows.

(2) Libraries

R has a very robust base set of functions ,but it also possesses many libraries for extra features,they are organized in so-called packages. With the standard installation, most common packages are installed. Example ggplot2,cars,. . . ..

(3) Assigning in R

You can use <- ,-> or = ,the recommended one is ,<-

(4) vectors and matrices

```
#Vectors
v1=c(1,2,38,9)
v1
```

```
## [1]  1  2 38  9
```

```
v2=seq(1,40,5)
v2
```

```
## [1]  1  6 11 16 21 26 31 36
```

```
v3=seq(1,40,by=3)
v3
```

```
##  [1]  1  4  7 10 13 16 19 22 25 28 31 34 37 40
```

```
v4=c("Sayama","SSIE500","TA","Dieudonne")
v4
```

```
## [1] "Sayama"    "SSIE500"   "TA"        "Dieudonne"
```

```
paste( v4[1],v4[2])
```

```
## [1] "Sayama SSIE500"
```

```
#Matrix
m1=matrix(c(2,1,3,4,5,6,7,8),ncol=4)
m1
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    3    5    7
## [2,]    1    4    6    8
```

```
m2=matrix(c(2,1,3,4,5,6,7,8),byrow=T,nrow=2)
m2
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2    1    3    4
## [2,]    5    6    7    8
```

(5) Data frames

```
df= data.frame(A = c(10,8,17),B= c(9,30,22), C = c(13,19,6))
df
```

```
##    A  B  C
## 1 10  9 13
## 2  8 30 19
## 3 17 22  6
```

```
mean(df$A)
```

```
## [1] 11.66667
```

```
mean(df[["A"]])
```

```
## [1] 11.66667
```

```
df[,3]
```

```
## [1] 13 19  6
```

```
df[2,2]
```

```
## [1] 30
```

(6) List

Another basic structure in R is a list. The main advantage of lists is that the "columns" (they're not really ordered in columns any more, but are more a collection of vectors) don't have to be of the same length, unlike matrices and data frames.

```
N= list(Sayama=1, Dieudonne=c(1,2),
David=seq(0, 4, length=5),Tom=5,Travis=8)
N$Sayama
```

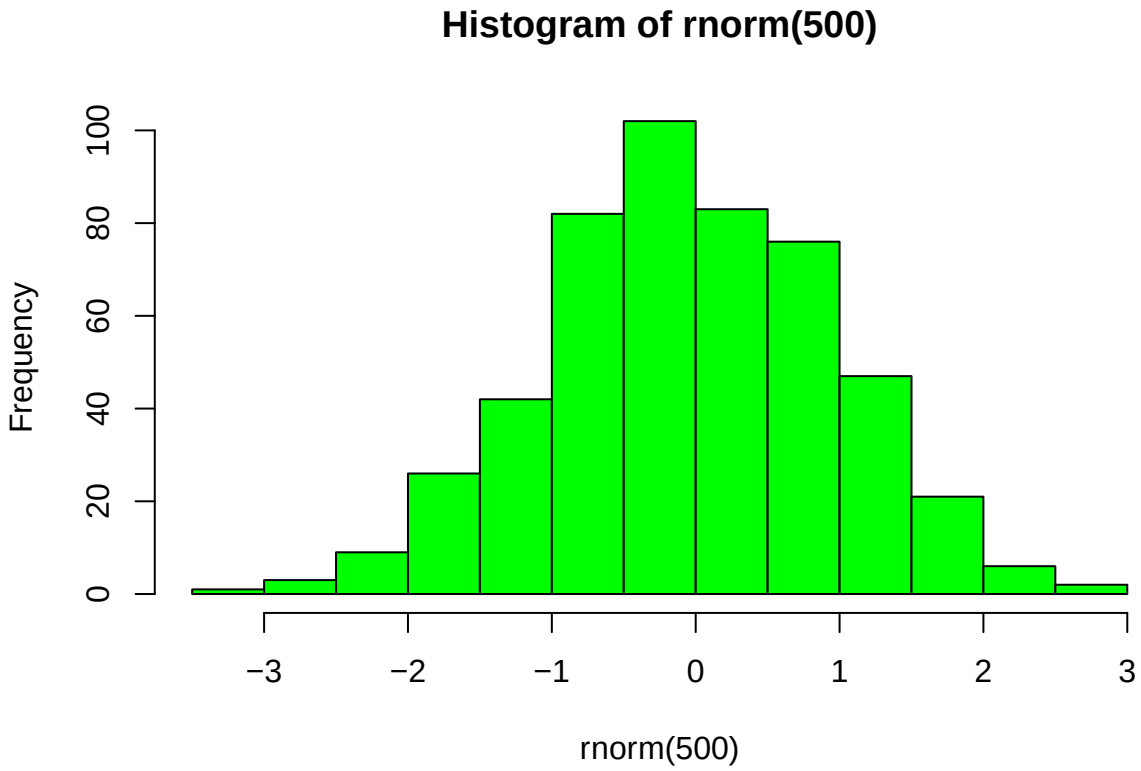
```
## [1] 1
```

```
N$David+6
```

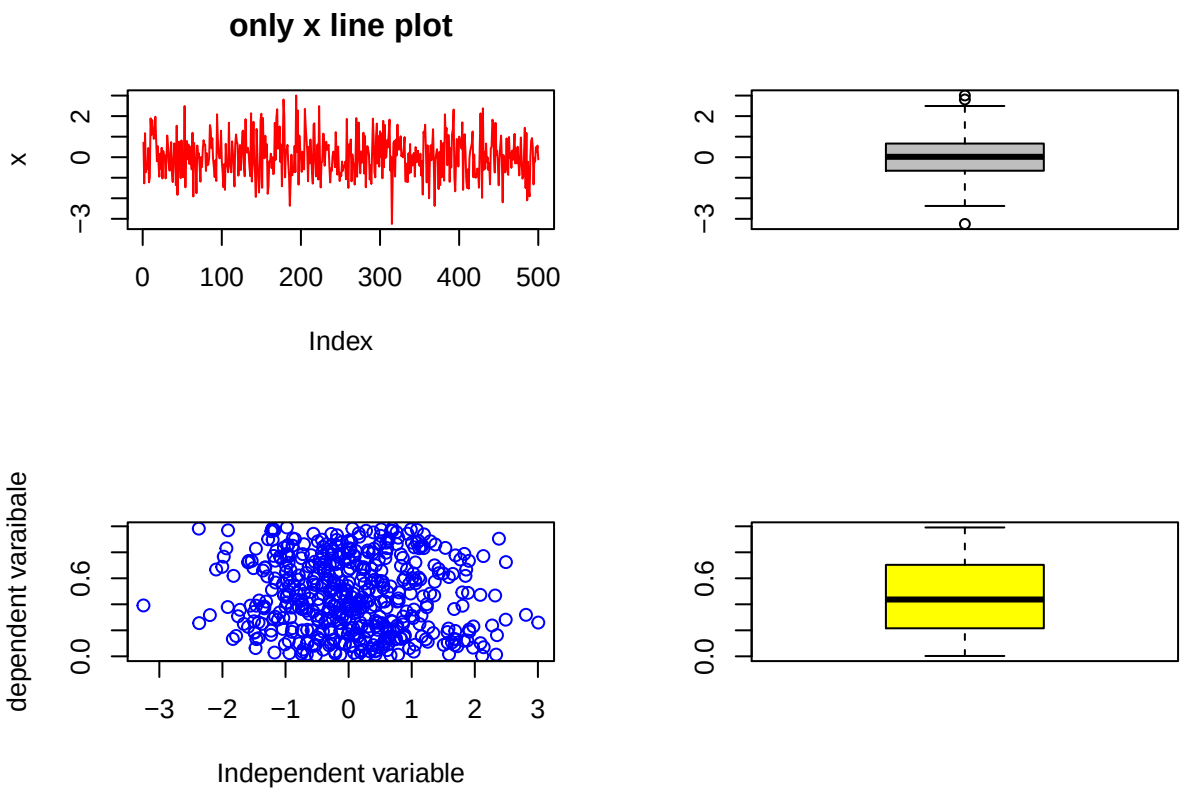
```
## [1]  6  7  8  9 10
```

(7) Graphics

```
hist(rnorm(500),col="green")
```




```
y=runif(500)
x=rnorm(500)
layout(matrix(c(1,2,3,4),2,2))
plot(x, type="l", col="red",main="only x line plot")
plot(x,y,col="blue",xlab='Independent variable',ylab="dependent varaibale")
boxplot(x,col="grey")
boxplot(y,col="yellow")
```



(8)Functions

R has many build in functions and the user can build his own functions

8.1 Built in functions

```
#Build in function
#c function concatenation function
a=c(23,11,6,7)
a
```

```
## [1] 23 11  6  7
```

```
b=c(1,5,9,10)
#standard deviation an mean
a_sd=sd(a)
a_sd
```

```
## [1] 7.804913
```

```
a_mean=mean(a)
a_mean
```

```
## [1] 11.75
```

```
#covariance and correlation
cov_ab=cov(a,b)
cov_ab
```

```
## [1] -30.58333
```

```
cor_ab=cor(a,b)
cor_ab
```

```
## [1] -0.952707
```

8.2 Your Own Functions

```
 fun1 = function(arg1, arg2 )
   {
   w = arg1 ^ 4
   return(2*arg2 + 3*w)
   }
fun1(arg1 = 7, arg2 = 11)
```

```
## [1] 7225
```

```
fun1(4,6)
```

```
## [1] 780
```

8.3 Examples

Derivation and integration

$f(x) = x^5 + x^4$

```
#Derivation
g= D(expression(x^5+x^4),'x')
#integral
## define the integrated function
integrand <- function(x) {1/((x+1)*sqrt(x))}
## integrate the function from 0 to infinity
integrate(integrand, lower = 0, upper = Inf)
```

```
## 3.141593 with absolute error < 2.7e-05
```

```
#Integration
f=function(x) {x^2+3*x+6}
integrate(f,lower=0,upper=70)
```

```
## 122103.3 with absolute error < 1.4e-09
```

(9) Useful tools

```
example(cor)
example(plot)
example(lm)
args(plot)
args(read.csv)
apropos("test")
choose(10,3)
factorial(5)
```

(10) Reading and writing files

10.1 Inside files

```
 d = data.frame(a = c(3,4,5),
 b = c(12,43,54))
write.table(d, file="tst0.txt",row.names=FALSE)
d2 = read.table(file="tst0.txt",header=TRUE)
d2
```

```
##   a  b
## 1 3 12
## 2 4 43
## 3 5 54
```

10.2 Reading data from the Internet

"http://chart.finance.yahoo.com/table.csv?s=FTR&a=0&b=28&c=2017&d=1&e=28&f=2017&g=d&ignore=.csv"

```
library(readr)
data<- read_csv("http://chart.finance.yahoo.com/table.csv?s=FTR&a=0&b=28&c=2017&d=1&e=28&f=2017&g=d&ignc
head(data)
```

```
##         Date Open High  Low Close   Volume Adj Close
## 1 2017-02-27 3.36 3.39 3.28  3.29 31783300      3.29
## 2 2017-02-24 3.40 3.42 3.37  3.37 16963000      3.37
## 3 2017-02-23 3.38 3.40 3.33  3.39 17551700      3.39
## 4 2017-02-22 3.33 3.39 3.31  3.35 18850500      3.35
## 5 2017-02-21 3.30 3.33 3.24  3.32 26775400      3.32
## 6 2017-02-17 3.29 3.30 3.23  3.29 26284300      3.29
```

(11) Dealing with NA values

```
m=c(6,7,11,NA)
mean(m)
```

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

```
#mean(m,na.rm=T)
```

(13) Dates

```
date1=strptime( c("20100225230000","20100226000000", "20100226010000"),format="%Y%m%d%H%M")
date1
```

```
## [1] "2010-02-25 23:00:00 EST" "2010-02-26 00:00:00 EST"
## [3] "2010-02-26 01:00:00 EST"
```

(14) If statement

```
w=8
if( w < 10 )
   {
   d=2
   }else{
   d=10
   }
d
```

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

(15) For loop

```
h = seq(from=1, to=8)
s = c()
for(i in 2:10)
   {
   s[i] = h[i] * 10
   }
s
```

```
##  [1] NA 20 30 40 50 60 70 80 NA NA
```

(16) Graphics with GGPLOT2 package

One main advantage R has over other statistical softwares is the possiblity to generate high quality graphics ,the most used package to achieve this is ggplot2 . Below are examples with ggplot2

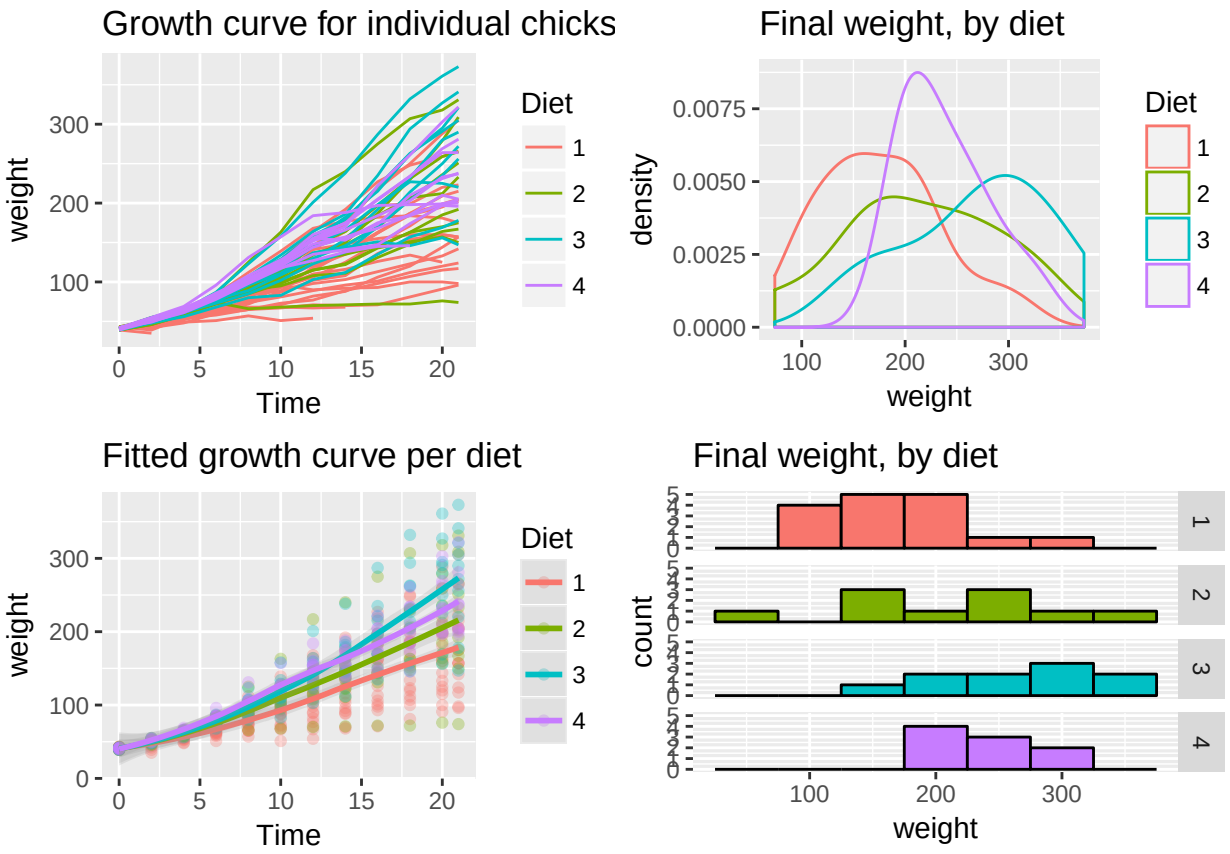
```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.3.2
```

```
# This example uses the ChickWeight dataset, which comes with ggplot2
# First plot
p1 <- ggplot(ChickWeight, aes(x=Time, y=weight, colour=Diet, group=Chick)) +
    geom_line() +
    ggtitle("Growth curve for individual chicks")

# Second plot
p2 <- ggplot(ChickWeight, aes(x=Time, y=weight, colour=Diet)) +
    geom_point(alpha=.3) +
    geom_smooth(alpha=.2, size=1) +
    ggtitle("Fitted growth curve per diet")

# Third plot
p3 <- ggplot(subset(ChickWeight, Time==21), aes(x=weight, colour=Diet)) +
    geom_density() +
    ggtitle("Final weight, by diet")

# Fourth plot
```

```r
p4 <- ggplot(subset(ChickWeight, Time==21), aes(x=weight, fill=Diet)) +
    geom_histogram(colour="black", binwidth=50) +
    facet_grid(Diet ~ .) +
    ggtitle("Final weight, by diet") +
    theme(legend.position="none")        # No legend (redundant in this graph)

multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  plots <- c(list(...), plotlist)

  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

 if (numPlots==1) {
    print(plots[[1]])

  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
                                      layout.pos.col = matchidx$col))
    }
  }
}

multiplot(p1, p2, p3, p4, cols=2)
```

```
## `geom_smooth()` using method = 'loess'
```



(17) Using R for Interactive visualization

R can be used to build interactive visualization called Shiny Apps You need to write a UI.R and Server.R codes where your functions and data are stored in . Below is an output of such visualization .

https://dieudonne.shinyapps.io/dieudonnecapstoneshinyapp/

(18) Advices

Reading Codes

Reading codes is probably the most effective way to learn the most advanced tools in R. Read the codes below ,search on the libraries used and try to comment each line!

Forecasting using R

```
library(knitr)
library(forecast)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: timeDate
```

```
## This is forecast 7.1
```

```
library(lambda.tools)
library(futile.logger)
do_forecast <- function(country, df, h=2, plot=FALSE) {
  flog.info("[%s] start", country)
  xs <- as.numeric(colnames(df))
  xs <- c(xs, max(xs) + (1:h))
  ys <- as.numeric(df[country,])
  fc<-thetaf(ys,h=h)
  if (plot) plot(fc)
  id <- sprintf("%s.%s", country, tail(xs,h))
  data.frame(Id=id, Prediction=as.numeric(fc$mean))
}
#This of code below will be different depending of the location of you data
df <- read.csv("~/Downloads/training-2.csv",row.names=1)
head(df)
```

```
##         X2000     X2001      X2002     X2003     X2004     X2005     X2006
## AUT 32.061170 37.279287 54.9843335 62.608582 71.315528 78.700298 86.190943
## BEL 99.246617 19.719073 49.7285460 53.684867 55.646251 52.515053 50.285237
## FRA 46.345897 58.113197 58.3293478 53.771285 69.918970 62.757300 64.323871
## DEU 85.301105 86.372933 99.9568401 95.105613 85.049990 68.249901 58.386503
## GRC  1.053642  1.536600  0.5279877  1.167499  1.150701  1.313203  1.854956
## ITA  4.081035  5.024211  5.8970434  7.527230  9.707260 12.638783 16.273143
##         X2007     X2008     X2009     X2010     X2011     X2012     X2013
## AUT 87.834523 94.170628 90.214187 86.535036 90.771290 94.154243 99.377402
## BEL 50.773295 48.557206 52.489125 56.234971 64.217771 62.468388 60.209594
## FRA 64.709656 71.248754 83.228358 87.577614 91.156700 94.071244 99.827232
## DEU 54.350046 56.605978 57.547879 58.581524 57.564131 59.500532 29.246463
## GRC  3.592906  4.722608  5.851027  6.699354  5.333539  4.483929  8.509164
## ITA 23.713618 28.009784 35.243334 35.949103 42.651568 46.648192 53.915155
```

```
colnames(df) <- sub('X','', colnames(df), fixed=TRUE)
out <- fold(rownames(df), function(i,acc) rbind(acc, do_forecast(i,df)), NULL)
```

```
## INFO [2017-02-28 15:20:03] [AUT] start
## INFO [2017-02-28 15:20:03] [BEL] start
## INFO [2017-02-28 15:20:03] [FRA] start
## INFO [2017-02-28 15:20:03] [DEU] start
## INFO [2017-02-28 15:20:03] [GRC] start
## INFO [2017-02-28 15:20:03] [ITA] start
## INFO [2017-02-28 15:20:03] [MLT] start
## INFO [2017-02-28 15:20:03] [ESP] start
## INFO [2017-02-28 15:20:03] [TUR] start
```

```
kable(out)
```

Id	Prediction
AUT.2014	101.720906
AUT.2015	104.064699

Id	Prediction
BEL.2014	56.698891
BEL.2015	56.791027
FRA.2014	101.605931
FRA.2015	103.490869
DEU.2014	27.229519
DEU.2015	25.209751
GRC.2014	8.779979
GRC.2015	9.051169
ITA.2014	55.910985
ITA.2015	57.907342
MLT.2014	99.456443
MLT.2015	103.307184
ESP.2014	92.024564
ESP.2015	92.541441
TUR.2014	101.905254
TUR.2015	103.890456

```
write.csv(out, 'submission.csv', row.names=FALSE, quote=FALSE)
```