Sample Answer Lab 01:

Handout date: Thursday, 3, 2020

Due date: Thursday, September 17, 2020 submitted as Word document to eLearning's ***SumitLab01*** link.

This lab counts 8 % toward your total grade

Task 1: Importing Data (0.5 pts)

Setup an working directory and save the files **MYPOWER.RDATA** , **CONCORD1.SAV** and **CPS1985.DBF**. into this directory.

a. Explore the load( ) function and import MyPower.RData .
```
setwd("E:\\Lectures2020\\EPPS2302\\Labs\\Lab01")
load("MyPower.RData"))
```
b. Use a function from the library foreign to import Concord1.sav and save it under the name Concord.
```
Concord <- foreign::read.spss("Concord1.sav", to.data.frame=TRUE)
```
It is important here to set the option **`to.data.frame=TRUE`**

c. Use a function from the library foreign to import CPS1985.dbf and save it under the name CPS1985.
```
CPS1985 <- foreign::read.dbf("CPS1985.dbf")
```
d. Explore the documentation of the data-frame **`Mroz`** in the library **`carData`** and link the data-frame to your session with the **`data( )`** function.
```
data(Mroz, package="carData")
```
e. To demonstrate that everything worked as intended show a screenshot of **GLOBAL ENVIRONMENT**, which displays all 4 data-frames.

The data-frame **`Mroz`** is initially just **`<Promised>`**. As soon as you call it or one of its variables with a R function its content will become fully available. The motivation behind R acting this way is to save temporarily memory space by just pointing at the data-frame in a package.

Data	
Concord	496 obs. of 10 variables
CPS1985	534 obs. of 11 variables
MyPower	56 obs. of 8 variables
Values	
Mroz	<Promise>

Task 2: Data-frame Basics (1.5 pts)

a. For the data-frame **`MyPower`** calculate the average daily power consumption by using the variables **`kWhBill`** and **`DaysBill`** and add the new variable to the data-frame **`MyPower`** with the variable name **`DailykWh`**. Show your code for this calculation. (0.2 pts)
```
MyPower$DailykWh <- MyPower$kWhBill / MyPower$DaysBill
```

b. Apply the statements

```
MyPowerNames <- names(MyPower)(1)
length(MyPowerNames) (2)
MyPowerNames[4:6] (3)
```

What are these statements doing? (0.2 pts)

Line 1: get column names and saves the string vector into a new object **`MyPowerNames`**

Line 2: reports the number number of elements, that is variables, in the vector **`MyPowerNames`**

Line 3: echoes the column names for 4th, 5th and 6th variables into the console

c. Apply the statement **`sapply(`**_`data-frame`_**`, is.factor)`** on the data-frame **`MyPower`** to evaluate the which variables are factors. What is this statement doing? Show a copy of the Console with the output of this investigation. (0.1 pts)

The statement **`sapply`** scans each variable in the data-frame and checks whether it is a factor

```
sapply(MyPower, is.factor)
SeqID     Year     Month  MinTemp  AveTemp  MaxTemp  kWhBill DaysBill DailykWh
FALSE    FALSE      TRUE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE
```

d. Apply the **`str( )`** on the data-frame **`MyPower`**. What information about the data-frame does the **`str( )`** provide to you? (0.1 pts)

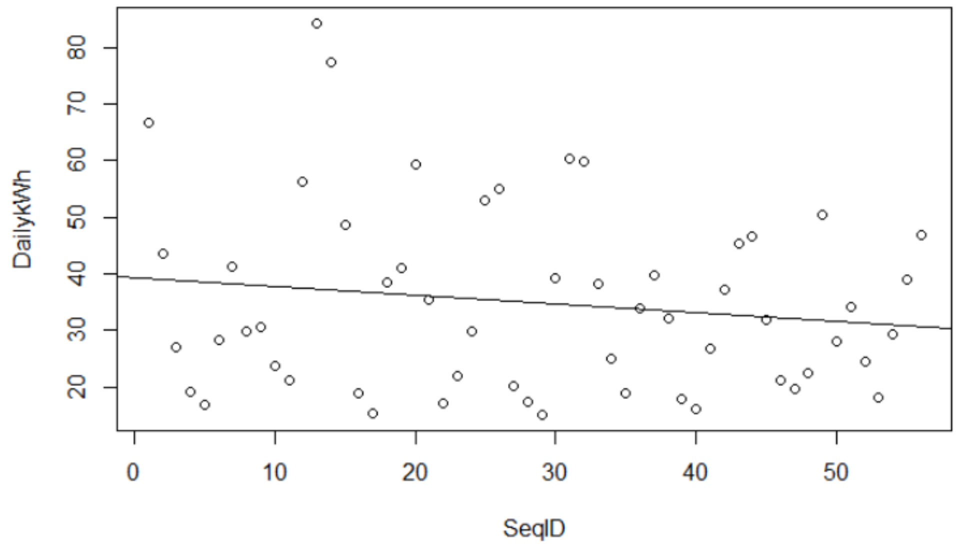
The function **`str()`** provides a look into the internal structure of the data-frame **`MyPower`** by showing the data type of each variable the first few values.)

e. What are the following statements doing? Show the plot and elaborate on the syntax of the statements. Is the power consumption over time decreasing? (0.3 pts)

```
plot(DailykWh~SeqID, data=MyPower)
abline( lm(DailykWh~SeqID, data=MyPower) )
```

Line 1: it scatterplots the daily consumption **`DailykWh`** as dependent variable on the y-axis against the independent sequence identifier **`SeqID`** on the x-axis. The symbol **~** separates the dependent from the independent variables. The sequence id matches the time order of the observations. Both variables are found in the data-frame **`MyPower`**.

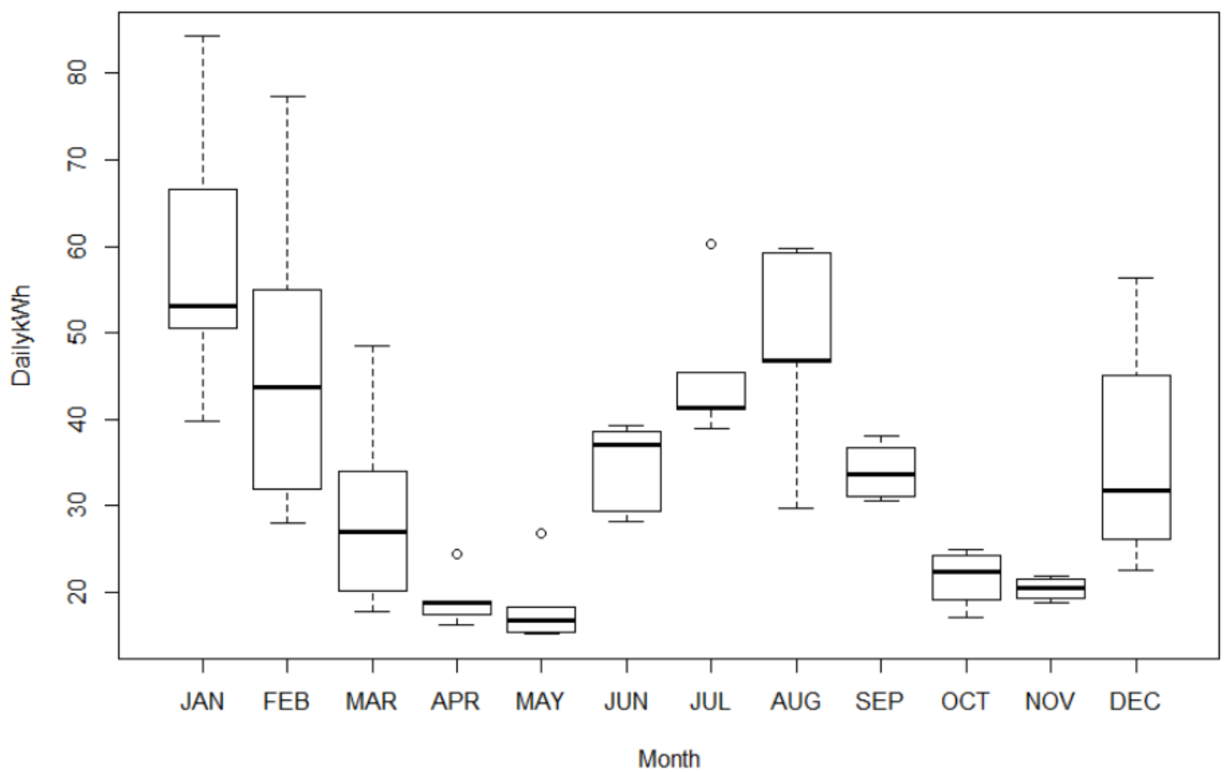
Line 2: In the **`lm( )`** function a linear regression line using both variable is calibrated. The **`abline( )`** is applied on the output of the **`lm( )`** function to add the regression line on top of the scatterpot.



f. What is the following statement doing? Show the plot and elaborate on the syntax of the statements. Why does the power consumption fluctuate over the seasons? (0.2 pts)

```
plot(DailykWh~Month, data=MyPower)
```

This command the plots side-by-side box plots of the daily power consumption **`DailykWh`** separately for each month **`Month`** . This plot uses again the function syntax **`y~x`**..



g. Use the syntax **MyPower[*rows*, *cols*]** to select all records with the three variables **c("MinTemp","AveTemp","MaxTemp")** (alternatively you could use **MyPowerNames[4:6]**) in the month of **MyPower$Month=="JAN"**. Show the code and its output. (0.2 pts)

h.
```
MyPower[MyPower$Month=="JAN",c("MinTemp","AveTemp","MaxTemp")]
   MinTemp AveTemp MaxTemp
1      35.1    48.1    61.1
13     34.5    44.3    54.0
25     31.9    42.8    53.7
37     38.9    50.4    61.9
49     39.1    49.1    59.1
```

i. Look and show at the header and the tail of the data-frame **MyPower** with the functions **head( )** and **tail( )**. (0.1 pts)

```
head(MyPower)
tail(MyPower)
```

Returns the first or last 6 records of a vector, matrix, table, data frame or function.

j. What class is the output **MetricPower** of the operation below? (0.1 pts)

```
MetricPower <- MyPower[ , c("MinTemp","AveTemp","MaxTemp","DailykWh")]
```

Data frame use **class(MetricPower)**

Task 3: R Basics (1 pt)

a. Depending on the input object class type R functions behave differently. To see the different class-specific implementations of the generic **summary( )** function try the command **methods(summary)**.
   Discuss the difference in the behavior of the **summary( )** function when applied to a **data.frame** or a **lm** object. (0.2 pts)
   The function **summary.data.frame** is defined for input objects of class **data.frame** whereas the function **summary.lm** is defined for objects of class **lm**. The generic function **summary( )** will determine its appropriate behavior according to the class of its input argument.

b. Explore the online help for
   ?summary.lm
   ?summary.data.frame
   and discuss the optional parameters (0.2 pts)
   summary.lm : process information in an class object and reports statistics about a regression model. Optional a correlation matrix among the estimated regression parameters can be requested.
   summary.data.frame : provides descriptive statistics about the matric variables and factors of a data-frame. Optional the significant digits of the descriptive statistics and shown factor levels can be requested.

c. What are the following statements below doing when processing the vector x of length 6? (0.3 pts)

```
x <- c(1,3,5,7,9,NA)
```

defines a numeric vector with 6 elements including one missing number

```
x * 2; x + 2 (1)
```

either multiplies each element of the vector by 2 or adds 2 to each element. The missing number remains a missing number. These are operations of combining elementwise a vector with a single number

```
y <- seq(0,2, by=1); x * y (2)
```

This operation combines the vector **x** of length 6 with the vector **y <- c(0,1,2)** of length 3. The elements of the shorter vector **y** are repeated to match the additional elements of the longer vector. The multiplication is executed element by element. A missing number again leads to a missing result.

```
> y <- seq(0,2, by=1); x * y
[1]  0  3 10  0  9 NA
```

```
z <- rep(c(1,2),3); x * z (3)
```

This operation generates a vector **z** of length 6 by repeating the elements **c(1,2)** three times. The multiplication with **x** is then carried out matching elements **x** with element sof **z**.

```
> z <- rep(c(1,2),3); x * z
[1]  1  6  5 14  9 NA
```

Note, the semicolon **;** allows to place several independent R commands into one line. Look the online help up for the functions **seq( )** and **rep( )**.

d. Define the function **myMean( )** and apply it on **x**

```
myMean <- function(x){
  x <- na.omit(x)
  sum(x)/length(x)
}
myMean(x)
```

Compare its output to that of the standard R function **base::mean( )**. Look up the help for the functions **na.omit( )** and **mean( )**. (0.2 pts)

The user function applies the statement **na.omit( )** and removes all **NA** elements from the input vector. In contrast, the **base::mean( )** function does not remove the NA elements and thus leads to an **NA** result. Its option **na.rm = TRUE** will first remove the **NA** elements and thus calculates the mean with valid numeric values.

e. How does the statement **x[c(T,T,T,T,T,F)]** work? (0.1 pts)

This statement only selects only those element of **x** for which the element index is **TRUE**.

```
> x[c(T,T,T,T,T,F)]
[1] 1 3 5 7 9
```

Task 4: Working with Data (1 pt)

For all steps below show your properly formatted code. You find the necessary code for the examples in Lander and the online help. Only if asked show also the output.

Import the SPSS data-file **Concord1.sav** as ***data-frame*** into the environment by using a function from the library **foreign**. Make sure to name your data-frame properly.

```
library(foreign)
concord <- read.spss('Concord1.sav',to.data.frame = TRUE)
```

a. Discuss the summary statistics for the water consumption: How did the ***average*** water consumption change from 1979 to 1981? (0.1 pts)

```
summary(concord)
water81     water80     water79
2296.214    2704.900    2974.165
```

Average water consumption reduces every year from 2,974 to 2,298 in 1979 to 1981.

b. Discuss the summary statistics: Which variable has ***missing*** observations? (0.1 pts)

```
summary(concord)
```

The summary statistics indicate that `water79` variable has 47 missing observations. `retire` is a factor and the remaining variables are metric

c. Discuss the summary statistics: Which variable is a factor? (0.1 pts)

```
class(concord$retire)
[1] "factor"
```

Variable retire is a factor variable.

d. List all ***case numbers*** (variable **case**), which have at least for one variable missing value in a variable. Show also the code. (0.2 pts)

```
concord[is.na(concord$water79),c("case")]
23 40 46 108 142 143 144 145 146 153 159 178 181 197 199 205
213 283 290 310 334 359 375 385 408 421 466 480 481 487 488 490
491 497 498 499 500 502 506 507 508 511 512 513 514 515 516
```

e. Calculate the ***average*** water consumption over the 3 years for each household and save it the new variable **meanWater** into the data-frame. Caution: also include households, which have **NA**s in the water consumption. Hint: look at the documentation of the function **rowMeans( )**. (0.2 pts)

```
meanWater <-
rowMeans(concord[c("water79","water80","water81")],na.rm=T)
```

f. Use logical statements to identify those households (variable **case**), which have above average water consumption in 1981. Show the code and the household numbers (0.2 pts)

```
concord[concord$water81 > meanWater, c("case")]
  [1]   9  10  11  19  29  33  37  38  39  41  49
 [12]  50  53  64  70  71  74  76  78  80  81  86
 [23]  87  88  91  92  95  97  98 102 104 109 117
 [34] 127 131 133 134 139 144 160 163 188 198 205
 [45] 209 219 226 232 233 254 257 283 296 297 303
 [56] 309 310 312 318 319 323 324 325 328 334 335
 [67] 338 345 346 347 348 349 350 352 359 361 364
 [78] 370 375 377 378 381 388 392 401 403 405 408
```

```
 [89] 417 421 434 443 445 454 457 459 465 476 483
[100] 500 515
```

g. Draw a random sample of 10 households without repetitions. Show the household numbers (variable **case**) and the code. Hint: look at the documentation of the function **sample( )**. (0.1 pts)

```
> sample(concord$case, 10, replace=FALSE)
 [1] 427 336 142 440 211 296  99  11  89 495
```

Task 5: Statistical Graphs (2 pts)

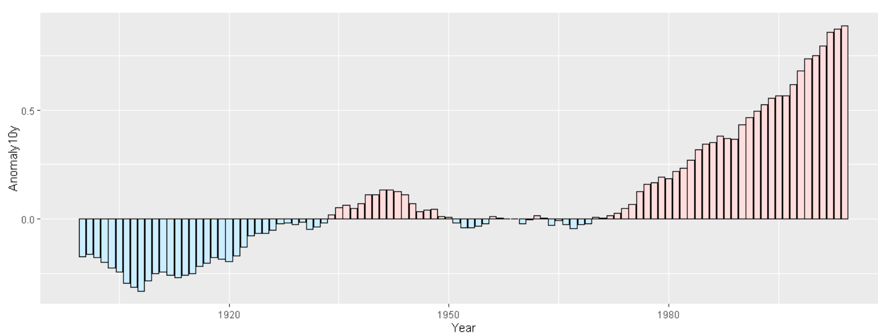
The following graphs are reproductions from the R Graphics Cookbook, 2e. One of the objectives of this task is that you visit the **R Graphics Cookbook** and explore the graphs on display there.

Study the sections associated with the graphs which are displayed below. Its sections document how these graphs were built. For e*ducational purpose* you may want to read the associated documentation of the employed functions and experiment with their options.

You job in this is to *reproduce* these graphs and *show the code*, which generated them. The data used in the **R Graphics Cookbook** are available in **library("gcookbook")**.

[a] Reproduce the graph below (0.5 pts)

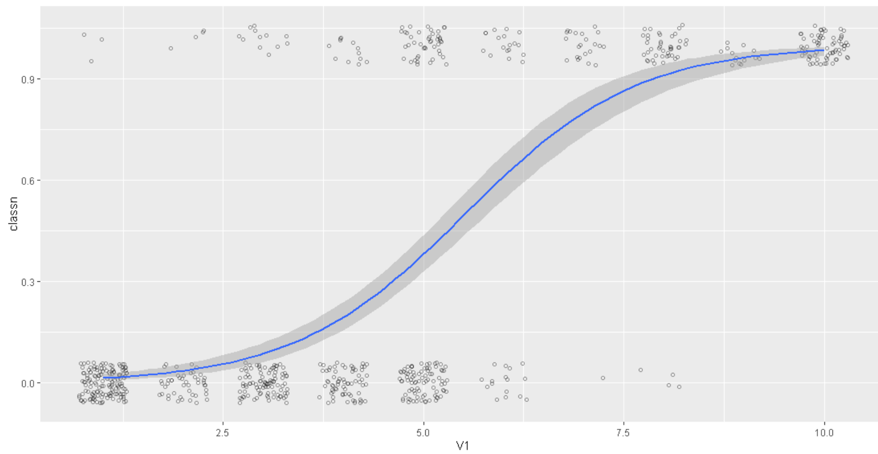
```
climate_sub <- climate %>% filter(Source == "Berkeley" & Year >= 1900) %>%
  mutate(pos=Anomaly10y >= 0)
ggplot(climate_sub, aes(x=Year, y=Anomaly10y, fill=pos)) +
  geom_col(position="identity", colour="black", size=0.25) +
  scale_fill_manual(values = c("#CCEEFF","#FFDDDD"), guide=FALSE)
```



[b] Reproduce the graph below (0.5 pts)

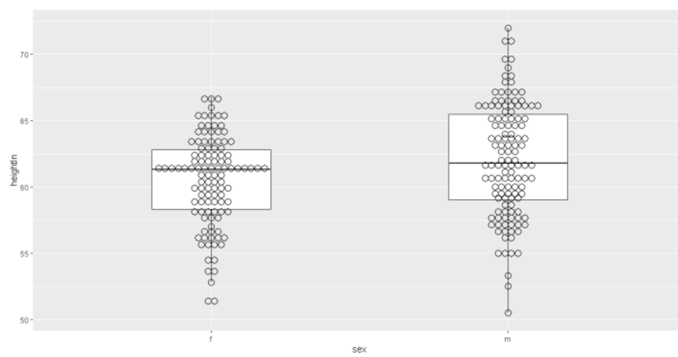
```
library(MASS)
biopsy_mod <- biopsy %>% mutate(classn = recode(class, benign=0,
malignant=1))

ggplot(biopsy_mod, aes(x=V1, y=classn))+
  geom_point(position=position_jitter(width=0.3, height=0.06), alpha=0.4,
shape=21, size=1.5)+
  stat_smooth(method=glm, method.args=list(family=binomial))
```



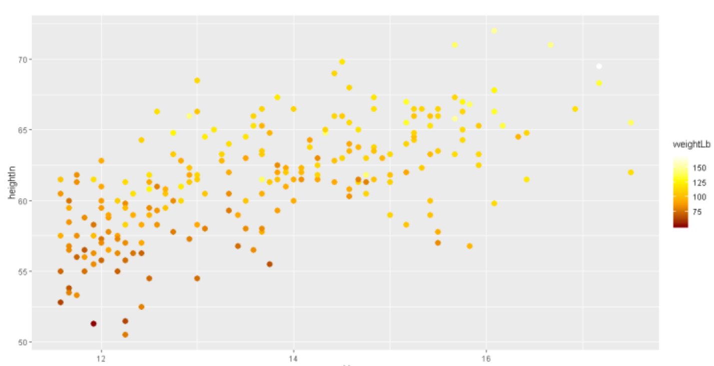
[c] Reproduce the graph below (0.5 pts)

```
ggplot(heightweight, aes(x=sex, y=heightIn)) +
  geom_boxplot(outlier.colour = NA, width= 0.4)+
  geom_dotplot(binaxis = "y", binwidth = 0.5, stackdir = "center", fill=NA)
```



[d] Reproduce the graph below (0.5 pts)

```
library(scales)
ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=weightLb))+
  geom_point(size=3)+
  scale_colour_gradientn(colours = c("darkred","orange","yellow","white"))
```



Task 6: Mapping (2 pts)

You are given tow ESRI shape files. These files are packed into the zipped file **`Italy.zip`**. The maps you will generate consist of two area layers: [a] the countries neighboring Italy (layer **`Neighbors.shp`**) and [b] added on top of it the 95 provinces of Italy (layer **`Provinces.shp`**). Make sure that the frame of the plot window is sized properly to embed Italy. The base map below shows these two layers in a properly sized window frame.

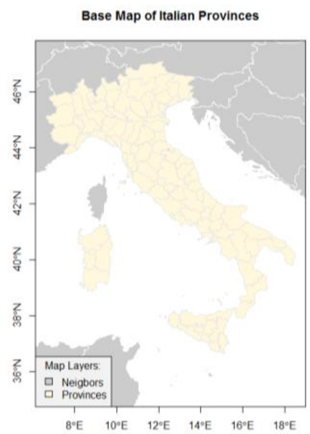


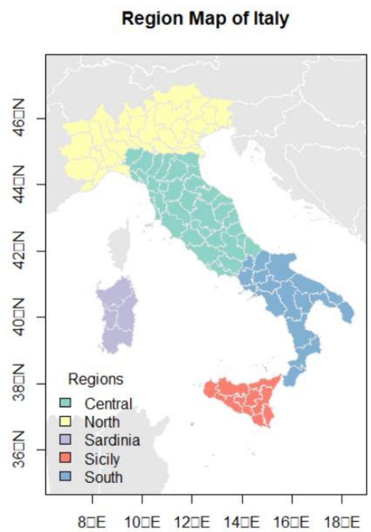
Figure 1: Base Map of the Italian Provinces

You will generate three color maps displaying different map themes. For your lab answers please show these maps in color. Each map should be properly framed, have a proper title and legend, show the neighboring countries as spatial reference frame. You can use the mapping functions in the package **`TexMix`**. Just show the code for your maps.

[a] Generate a qualitative map showing the Italian regions, which are stored in the variable **`REGION`**. Show the relevant code used to generate the map. (0.6 pts)

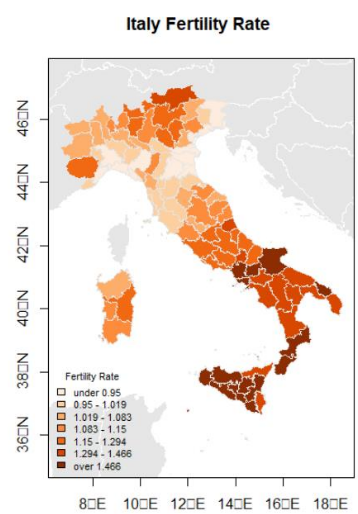
```
neig.shp <- rgdal::readOGR(dsn=getwd(), layer = "Neighbors",
                           stringsAsFactors=TRUE, integer64 = "allow.loss")

Italy.shp <- rgdal::readOGR(dsn=getwd(), layer = "Provinces",
                            stringsAsFactors=TRUE, integer64 = "allow.loss")
Italy.bbox <- bbox(Italy.shp)
plot(neig.shp,axes = T,col=grey(0.9),border = "white", xlim=Italy.bbox[1,],
     ylim=Italy.bbox[2,])
mapColorQual(Italy.shp$REGION, Italy.shp,
             map.title="Region Map of Italy",
             legend.title = "Regions",add.to.map=T)
```



[b] Generate a color ramp map showing the total fertility rate (number of births per woman) in the Italian provinces using 7 classes, which is stored in the variable **TOTFERTRAT**. Show the relevant code used to generate the map. (0.6 pts)

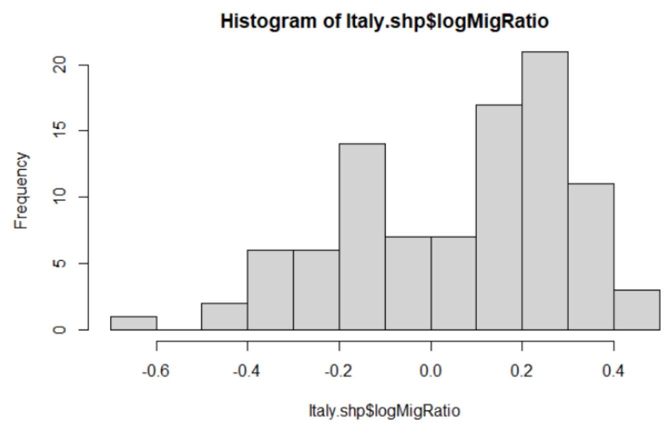
```
plot(neig.shp,axes=T,col=grey(0.9),border="white", xlim=Italy.bbox[1,],
     ylim=Italy.bbox[2,])
mapColorRamp(Italy.shp$TOTFERTRAT,Italy.shp, breaks=7,
             map.title="Italy Fertility Rate ",
             legend.title="Fertility Rate",
             add.to.map=T, legend.cex=0.7)
```



[c] Generate a bipolar map showing the log migration ratio in the 95 provinces, which can be calculated with the transformation **logMigRatio <- log(shp$INFLOW/shp$OUTFLOW)**, where **shp** refers to the name of your imported shape-file. What is the neutral break-point for the variable **logMigRatio**. Use in total 8 classes but select the appropriate number of classes for the below and

above breakpoint branches of the underlying distribution of **`logMigRatio`**. Justify your choice. Show the relevant code used to generate the map. (0.8 points)

```
Italy.shp$logMigRatio <- log(Italy.shp$INFLOW/Italy.shp$OUTFLOW)
hist(Italy.shp$logMigRatio)
```




```
sum(Italy.shp$logMigRatio <= 0)   #36
sum(Italy.shp$logMigRatio >= 0)   #59
```

The natural breaking point is 0 = log 1 when the inflow and outflow are in balance. Values greater than zero indicate population gain due to migration, whereas values below zero mean a province is losing population due to dominant outmigration.

```
plot(neig.shp,axes=T,col=grey(0.9),border="white",xlim=Italy.bbox[1,],
     ylim=Italy.bbox[2,])
mapBiPolar(Italy.shp$logMigRatio, Italy.shp, neg.breaks=3, pos.breaks=5,
           break.value=0, map.title="Italy Migration Ratio",
           legend.title="Ratio",add.to.map=T, legend.cex=0.7)
```