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.

1. Explore the load( ) function and import MyPower.RData .

load('MyPower.RData')

1. Use a function from the library foreign to import Concord1.sav and save it under the name Concord.

library(foreign)

Concord <- read.spss('Concord1.sav', to.data.frame=TRUE)

1. Use a function from the library foreign to import CPS1985.dbf and save it under the name CPS1985.

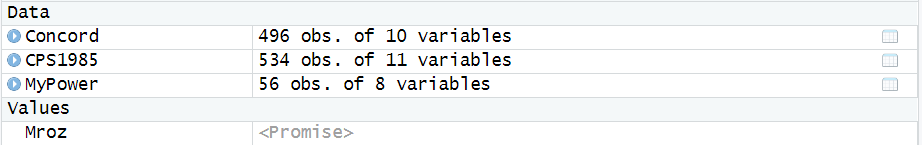
CPS1985 <- read.dbf('CPS1985.dbf')

1. 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")

1. To demonstrate that everything worked as intended show a screenshot of **Global Environment**, which displays all 4 data-frames.

Mroz is promise type when first time load (for saving the memory), once you call it in code, it would be transferred to a data frame.



# Task 2: Data-frame Basics (1.5 pts)

1. 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

1. Apply the statements   
    **MyPowerNames <- names(MyPower)(1)  
    length(MyPowerNames) (2)  
    MyPowerNames[4:6] (3)**  
   What are these statements doing? (0.2 pts)

⑴ get columns’ name and save in new variable  
⑵ get the number of columns’

⑶ get the column names for 4th to 6th variables

1. 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)

sapply(MyPower, is.factor)

SeqID Year Month MinTemp AveTemp MaxTemp kWhBill DaysBill DailykWh

FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE

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

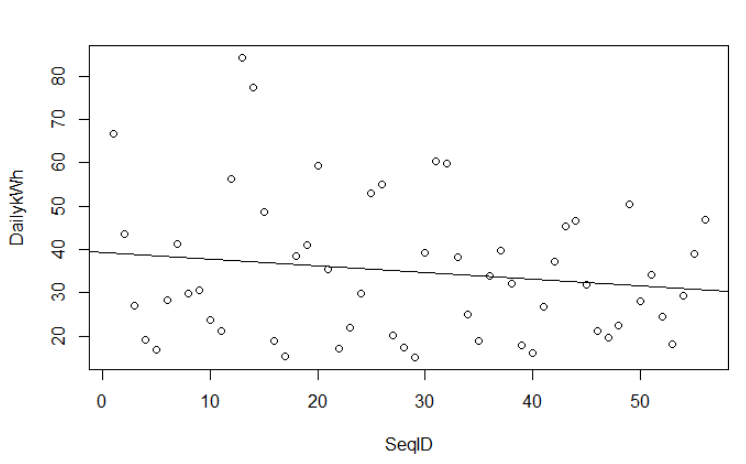
str(MyPower);

str() provides the internal structure of MyPower (shows the data type of each column)

1. 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) )**

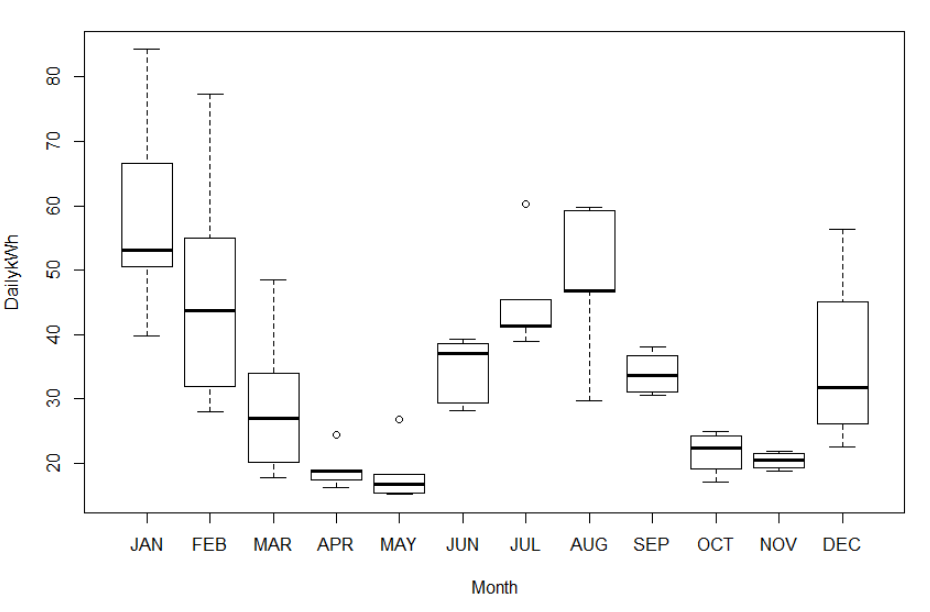
Scatterplot the Daily KWH (as y) across seqID(as x), then add the fitted line from the linear regression to it.

Since the SeqID follows the time order of the observations the power consumption over the years is decreasing.



1. 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)**

Since the electric heater runs in the winter and air conditioner in the summer,. in both seasons the power consumption increases.



1. 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)

**subset(MyPower[,4:6],MyPower$Month=="JAN")  
MyPower[MyPower$Month=="JAN",c("MinTemp","AveTemp","MaxTemp")]  
add output**

**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**

1. 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 parts of a vector, matrix, table, data frame or function.

1. 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)

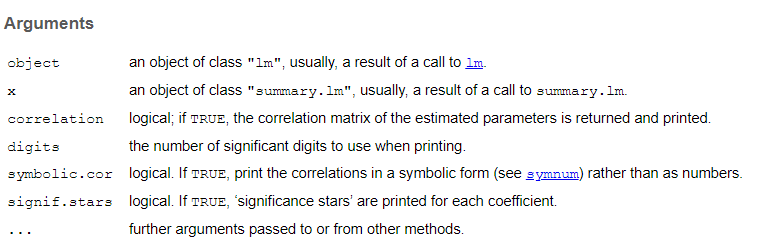
# Task 3: R Basics (1 pt)

1. Depending on the input object class type  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)

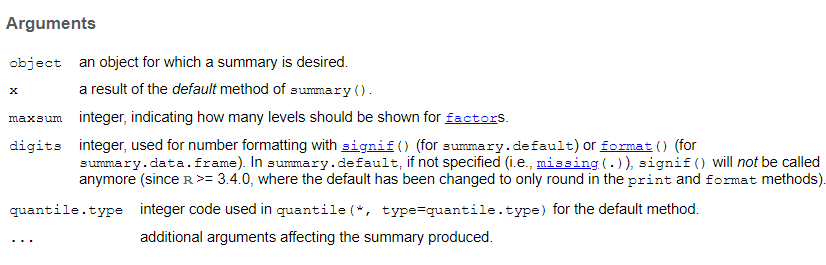
method(summary) shows all available summary methods from different packages

1. Data-frame: get the statistical information(mean, quantile, factor level counts, NA…) about each column of a data frame
2. Linear model: get the statistical information (residuals, coefficients, R-squared….) about this linear model
3. Explore the online help for  
    **?summary.lm  
    ?summary.data.frame**and discuss the optional parameters (0.2 pts)

?summary.lm



?summary.data.frame



1. 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)**  
    **x \* 2; x + 2 (1)**  
    **y <- seq(0,2, by=1); x \* y (2)**  
    **z <- rep(c(1,2),3); x \* z (3)**Note, the semicolon **;** allows to place several independent  commands into one line. Look the online help up for the functions **seq( )** and **rep( )**.

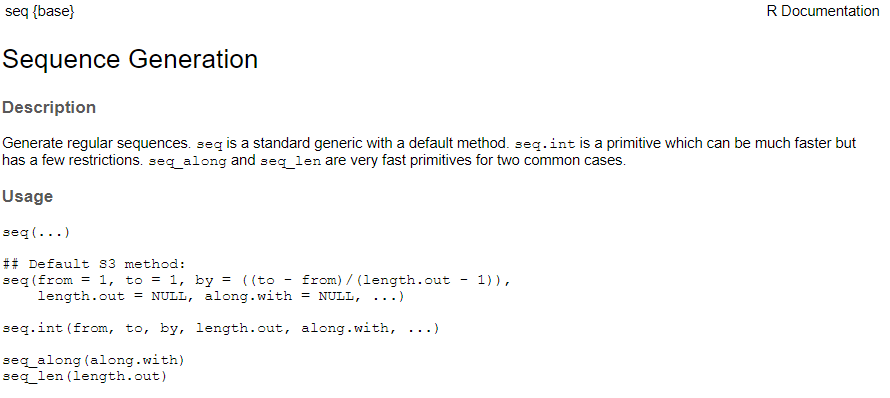
(1) Scalar multiplication: x[i] \* 2 or x[i] + 2 (i from 1 to 6)

(2) extended multiplication extends y to the same length with x by repeating, then y[i] \* x[i] (i from 1 to 6)

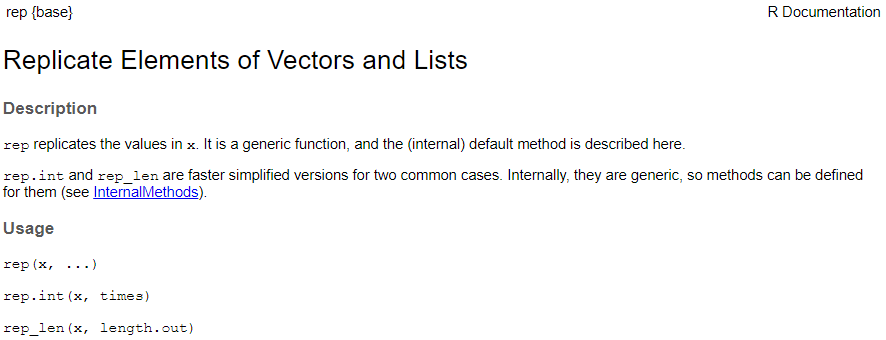
(3) element-wise multiplication: z[i] \* x[i] (i from 1 to 6)

1. 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  function **base::mean( )**. Look up the help for the functions **na.omit( )** and **mean( )**. (0.2 pts)

?seq()



?rep()



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

x[c(T,T,T,T,T,F)]

[1] 1 3 5 7 9

Gets rid of the last element (T stands for True and show the vector value at that position. F stands for False and suppresses the vector value at that position)

# 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)

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

sapply(na.omit(concord[,2:4]), mean)

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.

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

summary(Concord1)

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

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

class(concord$retire)  
[1] "factor"  
Variable retire is a factor variable.

1. 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$water79), ]$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

1. 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)

Use the class function to determine the type of each statement:  
[a]: factor,  
[b]: data.frame,  
[c]: factor,  
[d]: factor

1. 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)

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

1. 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)

above\_mean <- concord$water81 > mean(concord$water81,na.rm = TRUE)

concord$case[above\_mean]

5 6 7 9 11 13 15 16 17 18 19 23 26 29 31 41 42 43 50 52 53 56 57 63

67 68 71 74 76 80 86 87 88 89 92 93 94 97 99 101 102 103 104 107 109 115 116 117

118 120 121 124 127 128 133 134 136 138 139 144 149 152 154 155 156 158 159 160 163 164 166 169

172 180 182 183 186 194 196 198 203 204 206 207 208 213 215 216 218 219 220 224 228 229 230 232

235 237 240 243 246 247 250 252 254 267 274 276 279 281 282 283 288 289 293 294 296 297 298 300

305 307 309 313 316 318 319 321 323 324 325 326 330 331 333 334 335 338 349 350 354 355 356 357

360 361 369 371 374 375 377 378 379 381 382 383 387 391 392 395 396 408 419 422 423 426 430 431

432 433 434 436 438 439 442 444 445 446 451 454 456 458 460 464 465 466 467 468 470 471 472 473

474 476 477 478 481 484 486 488 492 493 494 495 497 498 501 502 503 504 506 507 508 509 512 513

515 516

# Task 5: Statistical Graphs (2 pts)

The following graphs are reproductions from the [R Graphics Cookbook, 2e](https://r-graphics.org/index.html). 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 educational 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)

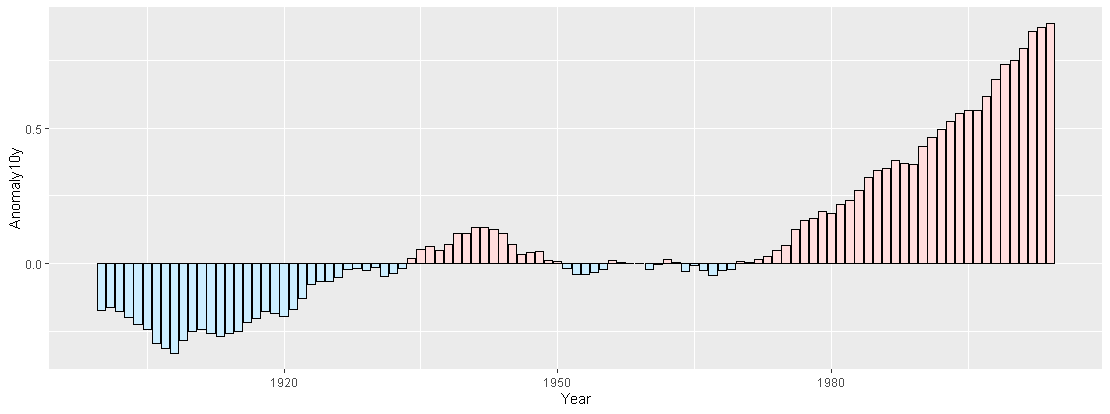
climate\_sub <- climate %>% filter(Source == "Berkeley" & Year >= 1900) %>%

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)

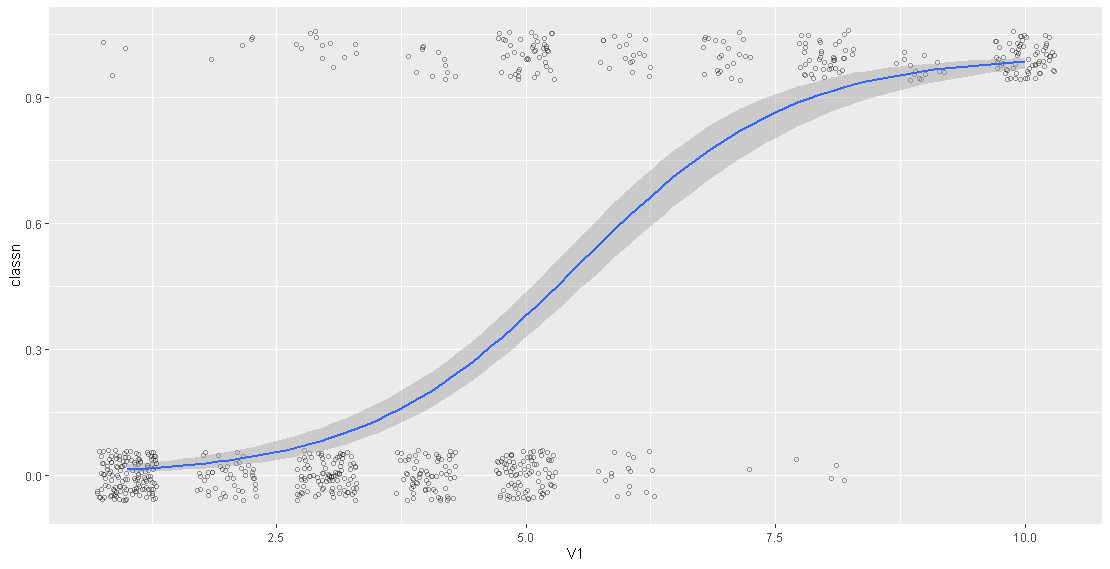
library(MASS)

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))

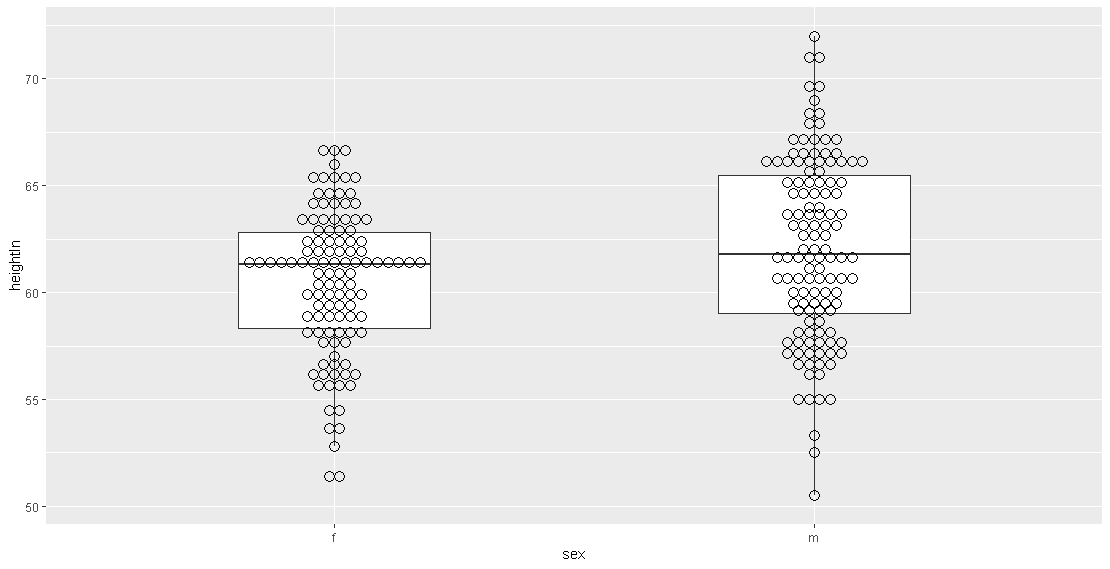


[c] Reproduce the graph below (0.5 pts)

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)



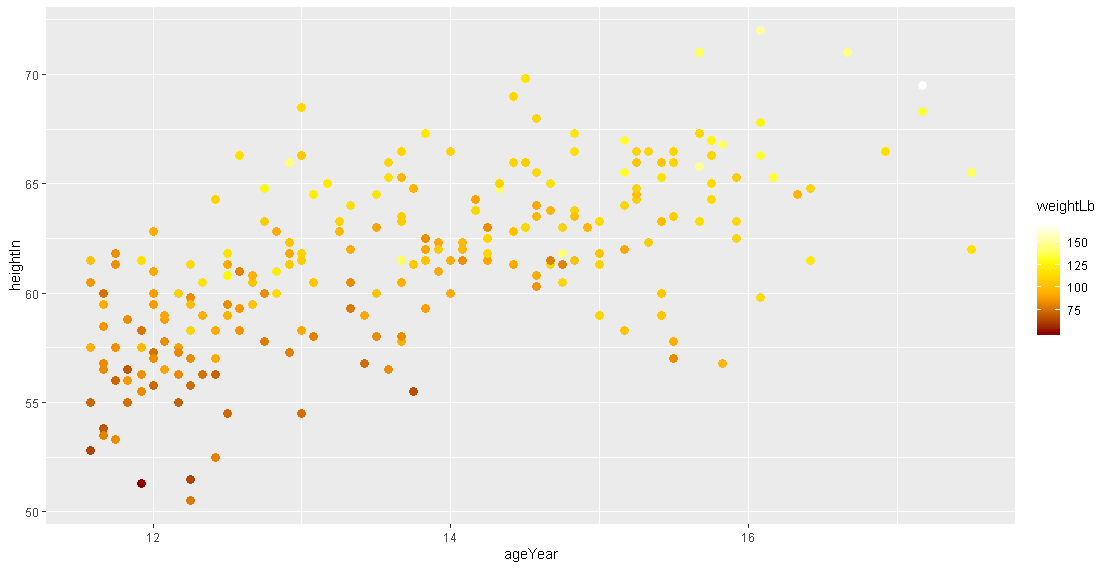
[d] Reproduce the graph below (0.5 pts)

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.

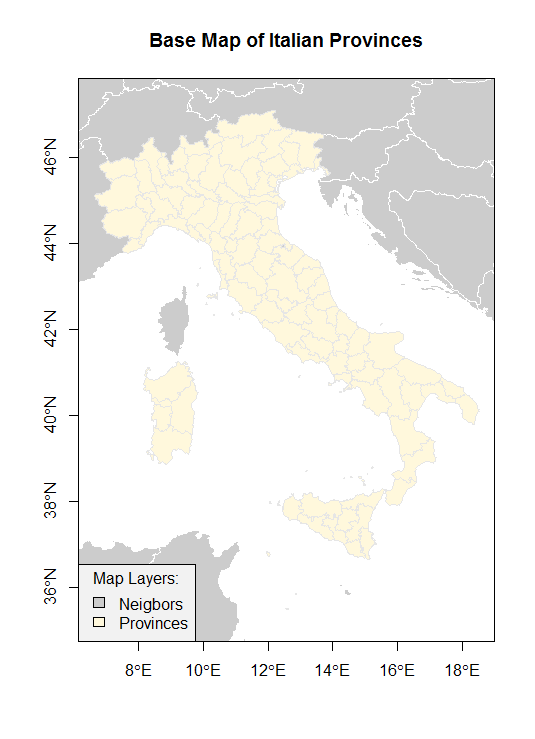


Figure 1: Base Map of the Italian Provinces

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)

library(rgdal)

library(TexMix)

#Get polygons of neighboring countries

neig.shp <- readOGR(dsn="./Italy",layer = "Neighbors", integer64 = "allow.loss")

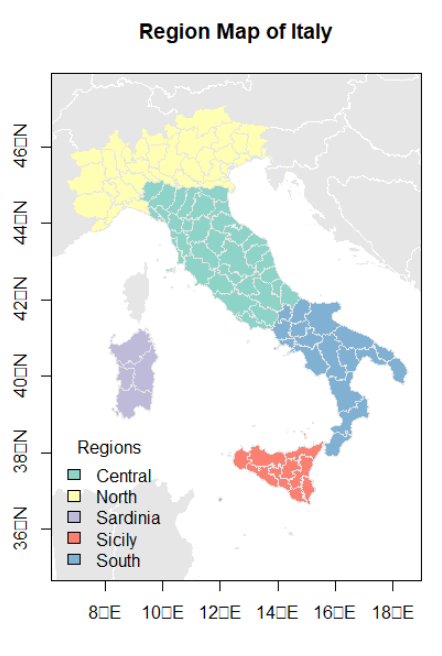
#Get polygons of Italy provinces

Italy.shp <- readOGR(dsn="./Italy",layer = "Provinces", 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(as.factor(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)

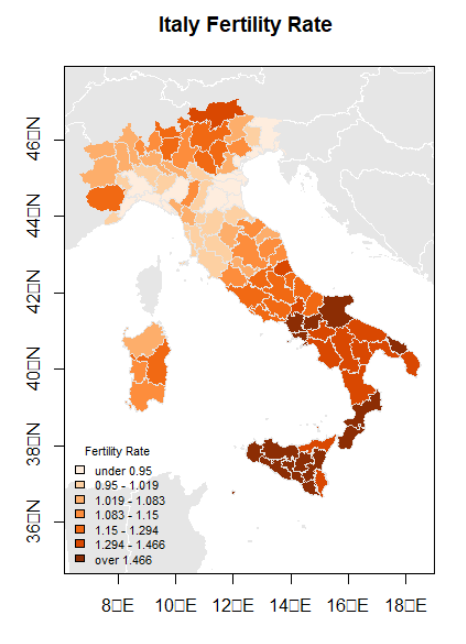
plot(neig.shp,axes=T,col=grey(0.9),border="white",

xlim=Italy.bbox[1,], ylim=Italy.bbox[2,])

# addToMap=T over-plots provinces over neighbors

mapColorRamp(Italy.shp$TOTFERTRAT,Italy.shp, breaks=7,

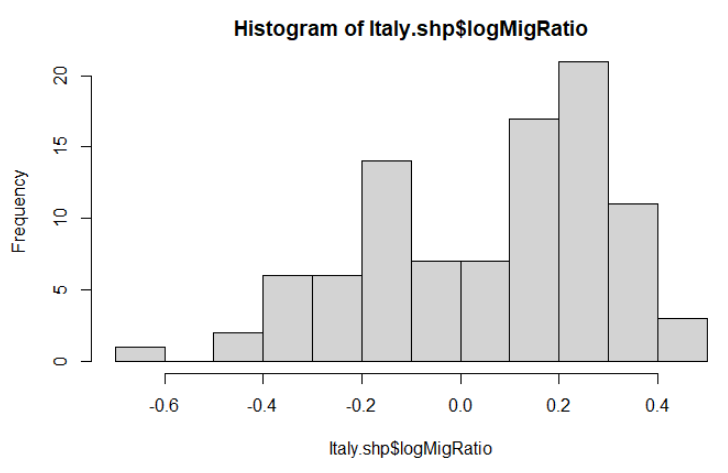
map.title="Italy Fertility Rate ", legend.title="Fertility Rate",add.to.map=T, legend.cex=0.7)



[c] Generate a bipolar map showing the gender 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)

