Lecture 13 - Exercise Solutions

Exercise 1

```
# loading dependency
library(brolgar)

# loading data
df_long = as.data.frame(heights)
```

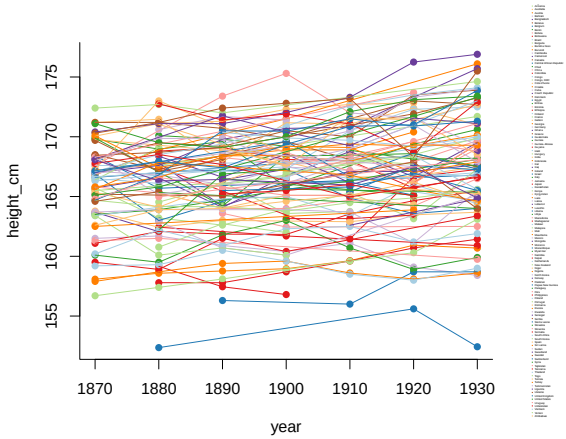
Question 1.1

```
# subsetting by year
df_long_sub <- df_long[df_long$year >= 1870 & df_long$year <= 1930, ]
```

Question 1.2

```
# loading dependency
library(ptmixed)

# making spaghetti plot
make.spaghetti(x = year,
               y = height_cm,
               id = country,
               group = country,
               data = df_long_sub,
               cex.leg = 0.175,
               legend.inset = -0.175)
```



Exercise 2

Question 2.1

We will be using `dcast` from `reshape2`. The function needs a data frame and a formula. The formula must be composed by the column (or columns) which uniquely define one row in the data frame. If the row is not defined uniquely an aggregation function is needed. The variable after `~` defines the columns where the data will appear.

```
# loading dependency
library(reshape2)

# transform from long to wide
df_wide = dcast(df_long,
                country + continent ~ year,
                value.var = 'height_cm')
```

```
# check data
head(df_wide)
```

Question 2.2

```
# checking df colnames
names(df_wide)
```

`paste0` is equivalent to `paste` when `sep = ''`.

```
# adjusting names
names(df_wide)[3:ncol(df_wide)] <- paste0('height_', names(df_wide)[3:ncol(df_wide)])
head(df_wide)
```

```
##       country continent height_1550 height_1650 height_1660 height_1670
## 1 Afghanistan      Asia          NA          NA          NA          NA
## 2     Albania    Europe          NA          NA          NA          NA
## 3     Algeria    Africa          NA          NA          NA          NA
## 4      Angola    Africa          NA          NA          NA          NA
## 5   Argentina  Americas          NA          NA          NA          NA
## 6     Armenia      Asia          NA          NA          NA          NA
##   height_1680 height_1690 height_1700 height_1710 height_1720 height_1730
## 1          NA          NA          NA          NA          NA          NA
## 2          NA          NA          NA          NA          NA          NA
## 3          NA          NA          NA          NA          NA          NA
## 4          NA          NA          NA          NA          NA          NA
## 5          NA          NA          NA          NA          NA          NA
## 6          NA          NA          NA          NA          NA          NA
##   height_1740 height_1750 height_1760 height_1770 height_1780 height_1790
## 1          NA          NA          NA          NA          NA          NA
## 2          NA          NA          NA          NA          NA          NA
## 3          NA          NA          NA          NA          NA          NA
## 4          NA          NA          NA          NA          NA       160.4
## 5          NA          NA          NA       170.3       168.2       168.0
```

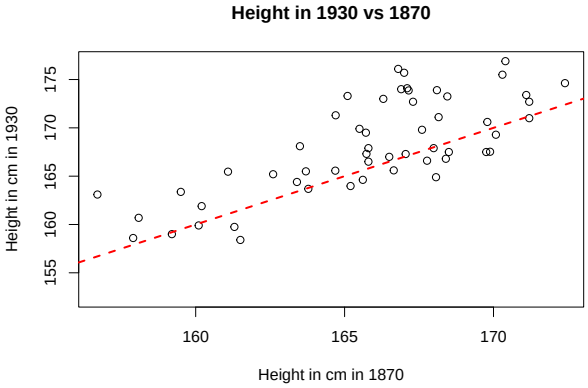
```
## 6          NA          NA          NA          NA          NA          NA
##   height_1800 height_1810 height_1820 height_1830 height_1840 height_1850
## 1          NA          NA          NA          NA          NA          NA
## 2          NA          NA          NA          NA          NA          NA
## 3          NA          NA          NA          NA          NA          NA
## 4       158.6       160.5          NA          NA          NA          NA
## 5       168.0       168.8       169.9       170.9       169.6     168.200
## 6          NA          NA          NA          NA          NA     169.324
##   height_1860 height_1870 height_1880 height_1890 height_1900 height_1910
## 1          NA       168.4     165.690          NA          NA          NA
## 2          NA          NA     170.100     169.800     169.200          NA
## 3          NA          NA          NA          NA          NA       168.80
## 4          NA          NA     168.800     169.100     168.100       168.00
## 5      167.40       167.6     167.565     167.792     167.868       168.20
## 6      167.62          NA          NA     166.200     163.580       163.75
##   height_1920 height_1930 height_1940 height_1950 height_1960 height_1970
## 1          NA       166.8          NA          NA          NA          NA
## 2          NA          NA          NA          NA          NA          NA
## 3     166.241       169.0          NA          NA          NA          NA
## 4     165.700       166.7          NA          NA          NA          NA
## 5     169.000       169.8      170.60       170.8          NA          NA
## 6          NA          NA      167.99          NA     170.601     171.749
##   height_1980 height_1990 height_2000
## 1          NA       167.1       161.4
## 2          NA          NA       167.9
## 3          NA       171.3       169.5
## 4          NA          NA          NA
## 5          NA       174.4          NA
## 6     171.653       172.4       172.2
```

Question 2.3

In order to add a line to a base `R` plot we use `abline`. To make $x = y$ we just need to make a line with slope 1. In `abline` the intercept is defined as `a` and slope as `b` arguments.

```
plot(df_wide$height_1870,
     df_wide$height_1930,
     main = 'Height in 1930 vs 1870',
     xlab = 'Height in cm in 1870',
     ylab = 'Height in cm in 1930')

abline(a = 0,
       b = 1,
       lwd = 2,
       col = 'red',
       lty = 2)
```



Question 2.4

The people in most countries seem to be taller in 1930 compared to 1870. Which was to be expected, since with time, countries get economically stronger and have improved healthcare and social security. However, there are some countries that are below the red dotted line, indicating that the people are on average smaller in those countries in 1930 than they were in 1870. We can find out for which countries this was the case. As can be seen, the countries where people were smaller in 1930 were mostly developing countries.

```
# retrieving countries whose average height was smaller in 1930 compared to 1870
df_wide[which((df_wide$height_1930 - df_wide$height_1870) < 0), 'country']
```

```
##  [1] "Afghanistan"   "Burkina Faso"  "Cambodia"      "Cote d'Ivoire"
##  [5] "Greece"        "Guinea"        "India"         "Malawi"
##  [9] "Myanmar"       "Nigeria"       "Pakistan"      "Peru"
## [13] "Sudan"         "Tanzania"      "Turkey"        "Uganda"
## [17] "Vietnam"
```

Exercise 3

Question 3.1

```
# loading data week 3
metadata <- load('metadata.RData')
population <- load('population.RData')
```

Question 3.2 & 3.3

It appears that for both datasets we have a three letter country code that could be used as a merge variable. This would be a horizontal merge, since both dataframes contain different columns for the same individuals.

```
# inspecting data
names(metadata)
names(population)

# inspecting id variable
population$Country.Code
metadata$ï..Country.Code
```

Question 3.4

```
# renaming id variable metadata
names(metadata)[1] <- 'Country.Code'

# merging datasets
pop.dat <- merge(metadata, population, by = 'Country.Code')

# Check if merge was successful
# View(pop.dat)
# names(pop.dat)
```

Question 3.5

We can get the length of the characters in `Region` using `nchar`. Then, subset where the number of characters of the region variable is not 0.

```
pop.dat.sub <- pop.dat[!(nchar(pop.dat$Region)==0),]
```

Question 3.6

With the `select` function we can select specific columns from the original data frame. This subset is grouped by the `Region` variable in the `group_by` function. This allows us to apply functions to each of the different regions. Lastly, the `top_n` function selects the largest $n = 1$ values for the variable `X2000` for each region.

```r
# loading dependency
library(dplyr)

# Retrieving statistics based on piping
pop.dat.sub |>
  select(Country.Name, Region, X2000) |>
  group_by(Region) |>
  top_n(1, X2000)
```

```
## # A tibble: 7 x 3
## # Groups:   Region [7]
##   Country.Name       Region                          X2000
##   <chr>              <chr>                           <dbl>
## 1 Brazil             Latin America & Caribbean   174790339
## 2 China              East Asia & Pacific        1262645000
## 3 Egypt, Arab Rep.   Middle East & North Africa   68831561
## 4 India              South Asia                 1056575548
## 5 Nigeria            Sub-Saharan Africa          122283853
## 6 Russian Federation Europe & Central Asia       146596869
## 7 United States      North America               282162411
```

Exercise 4

```
data(iris)
df.list = split(iris, iris$Species)
lapply(df.list, summary)
```

Question 4.1

Base `split` function may be used, however the `f` argument does not accept *data masking* so we have to define it as `iris$Species` since just `Species` will raise an error. This is against the pipe philosophy.

```
iris |> split(iris$Species) |> lapply(summary)
```

```
## $setosa
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.300   Min.   :1.000   Min.   :0.100
##  1st Qu.:4.800   1st Qu.:3.200   1st Qu.:1.400   1st Qu.:0.200
##  Median :5.000   Median :3.400   Median :1.500   Median :0.200
##  Mean   :5.006   Mean   :3.428   Mean   :1.462   Mean   :0.246
##  3rd Qu.:5.200   3rd Qu.:3.675   3rd Qu.:1.575   3rd Qu.:0.300
##  Max.   :5.800   Max.   :4.400   Max.   :1.900   Max.   :0.600
##        Species
##  setosa    :50
##  versicolor: 0
##  virginica : 0
##
##
##
##
## $versicolor
##   Sepal.Length    Sepal.Width     Petal.Length   Petal.Width          Species
##  Min.   :4.900   Min.   :2.000   Min.   :3.00   Min.   :1.000   setosa    : 0
##  1st Qu.:5.600   1st Qu.:2.525   1st Qu.:4.00   1st Qu.:1.200   versicolor:50
##  Median :5.900   Median :2.800   Median :4.35   Median :1.300   virginica : 0
##  Mean   :5.936   Mean   :2.770   Mean   :4.26   Mean   :1.326
##  3rd Qu.:6.300   3rd Qu.:3.000   3rd Qu.:4.60   3rd Qu.:1.500
##  Max.   :7.000   Max.   :3.400   Max.   :5.10   Max.   :1.800
##
## $virginica
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.900   Min.   :2.200   Min.   :4.500   Min.   :1.400
##  1st Qu.:6.225   1st Qu.:2.800   1st Qu.:5.100   1st Qu.:1.800
##  Median :6.500   Median :3.000   Median :5.550   Median :2.000
##  Mean   :6.588   Mean   :2.974   Mean   :5.552   Mean   :2.026
##  3rd Qu.:6.900   3rd Qu.:3.175   3rd Qu.:5.875   3rd Qu.:2.300
##  Max.   :7.900   Max.   :3.800   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    : 0
##  versicolor: 0
##  virginica :50
##
```

```
##
##
```

If we want neater code we need to use a function that works with *data masking*. `group_split` from `dplyr` is exactly what we need. Please see `?group_split`.

```
iris |> group_split(Species) |> lapply(summary)
```

```
## [[1]]
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.300   Min.   :1.000   Min.   :0.100
##  1st Qu.:4.800   1st Qu.:3.200   1st Qu.:1.400   1st Qu.:0.200
##  Median :5.000   Median :3.400   Median :1.500   Median :0.200
##  Mean   :5.006   Mean   :3.428   Mean   :1.462   Mean   :0.246
##  3rd Qu.:5.200   3rd Qu.:3.675   3rd Qu.:1.575   3rd Qu.:0.300
##  Max.   :5.800   Max.   :4.400   Max.   :1.900   Max.   :0.600
##        Species
##  setosa    :50
##  versicolor: 0
##  virginica : 0
##
##
##
##
## [[2]]
##   Sepal.Length    Sepal.Width     Petal.Length   Petal.Width          Species
##  Min.   :4.900   Min.   :2.000   Min.   :3.00   Min.   :1.000   setosa    : 0
##  1st Qu.:5.600   1st Qu.:2.525   1st Qu.:4.00   1st Qu.:1.200   versicolor:50
##  Median :5.900   Median :2.800   Median :4.35   Median :1.300   virginica : 0
##  Mean   :5.936   Mean   :2.770   Mean   :4.26   Mean   :1.326
##  3rd Qu.:6.300   3rd Qu.:3.000   3rd Qu.:4.60   3rd Qu.:1.500
##  Max.   :7.000   Max.   :3.400   Max.   :5.10   Max.   :1.800
##
## [[3]]
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.900   Min.   :2.200   Min.   :4.500   Min.   :1.400
##  1st Qu.:6.225   1st Qu.:2.800   1st Qu.:5.100   1st Qu.:1.800
##  Median :6.500   Median :3.000   Median :5.550   Median :2.000
##  Mean   :6.588   Mean   :2.974   Mean   :5.552   Mean   :2.026
##  3rd Qu.:6.900   3rd Qu.:3.175   3rd Qu.:5.875   3rd Qu.:2.300
##  Max.   :7.900   Max.   :3.800   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    : 0
##  versicolor: 0
##  virginica :50
##
##
##
```

4.2

```r
# %>% is imported from magrittr by default with dplyr.
iris %>% group_split(Species) %>% lapply(summary)
```

```
## [[1]]
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.300   Min.   :1.000   Min.   :0.100
##  1st Qu.:4.800   1st Qu.:3.200   1st Qu.:1.400   1st Qu.:0.200
##  Median :5.000   Median :3.400   Median :1.500   Median :0.200
##  Mean   :5.006   Mean   :3.428   Mean   :1.462   Mean   :0.246
##  3rd Qu.:5.200   3rd Qu.:3.675   3rd Qu.:1.575   3rd Qu.:0.300
##  Max.   :5.800   Max.   :4.400   Max.   :1.900   Max.   :0.600
##        Species
##  setosa    :50
##  versicolor: 0
##  virginica : 0
##
##
##
##
## [[2]]
##   Sepal.Length    Sepal.Width     Petal.Length   Petal.Width          Species
##  Min.   :4.900   Min.   :2.000   Min.   :3.00   Min.   :1.000   setosa    : 0
##  1st Qu.:5.600   1st Qu.:2.525   1st Qu.:4.00   1st Qu.:1.200   versicolor:50
##  Median :5.900   Median :2.800   Median :4.35   Median :1.300   virginica : 0
##  Mean   :5.936   Mean   :2.770   Mean   :4.26   Mean   :1.326
##  3rd Qu.:6.300   3rd Qu.:3.000   3rd Qu.:4.60   3rd Qu.:1.500
##  Max.   :7.000   Max.   :3.400   Max.   :5.10   Max.   :1.800
##
## [[3]]
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.900   Min.   :2.200   Min.   :4.500   Min.   :1.400
##  1st Qu.:6.225   1st Qu.:2.800   1st Qu.:5.100   1st Qu.:1.800
##  Median :6.500   Median :3.000   Median :5.550   Median :2.000
##  Mean   :6.588   Mean   :2.974   Mean   :5.552   Mean   :2.026
##  3rd Qu.:6.900   3rd Qu.:3.175   3rd Qu.:5.875   3rd Qu.:2.300
##  Max.   :7.900   Max.   :3.800   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    : 0
##  versicolor: 0
##  virginica :50
##
##
##
```

Exercise 5

Question 5.1

We need to load the data, replace *Not Available* by `NA` and remove %.

```
# loading data
df <- read.csv("irish_polls.csv")

# replaces all instances of "Not Available" with NA
df[df == "Not Available"] <- NA
# for each column, for each row, if % in string, replaces with ""
df[, 10:20] = apply(df[,10:20], 2, function(x) gsub("%", "", x))
df[, 10:20] = apply(df[,10:20], 2, as.numeric)
```

Question 5.2

```
# subsetting data
df[2:11, c(4,10:17,20)]
```

Question 5.3

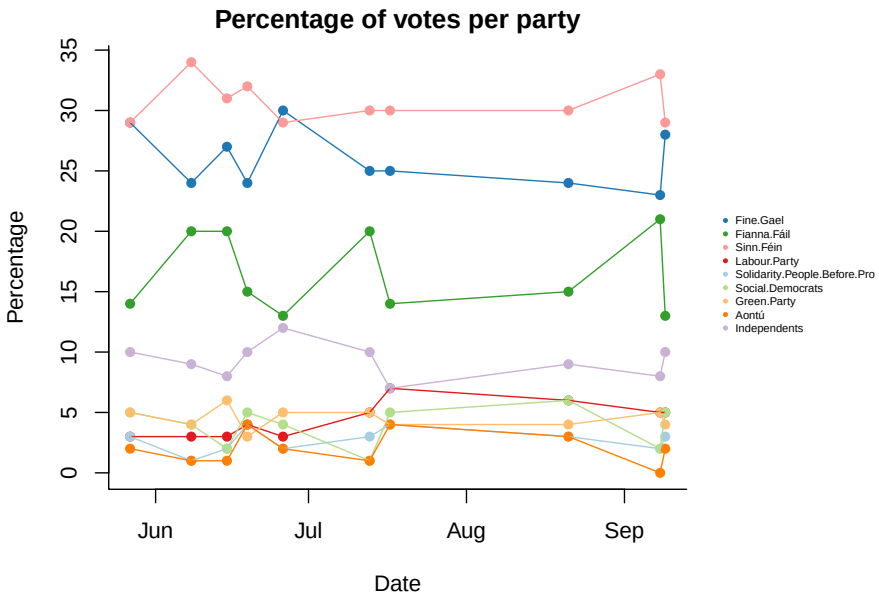
We can transform `character` class variables to `Date` class using `as.Date`.

```
# transform to long
irish_long = melt(df[2:11, c(4,10:17,20)], id.vars = 'Fieldwork.End')

# adjust dates to Date format
irish_long$Fieldwork.End <- as.Date(irish_long$Fieldwork.End, "%Y-%m-%d")
```

Question 5.4

```
# plotting
make.spaghetti(x = Fieldwork.End,
               y = value,
               id = variable,
               group = variable,
               data = irish_long,
               cex.leg = 0.5,
               legend.inset = -0.35,
               title = 'Percentage of votes per party',
               ylab = 'Percentage',
               xlab = 'Date')
```



Question 5.5

```
# calculating mean
irish_long %>%
  group_by(variable) %>%
  summarise(Mean = mean(value))
```

```
## # A tibble: 9 x 2
##   variable                          Mean
##   <fct>                            <dbl>
## 1 Fine.Gael                         25.9
## 2 Fianna.Fáil                       16.5
## 3 Sinn.Féin                         30.7
## 4 Labour.Party                       4.4
## 5 Solidarity.People.Before.Profit    2.7
## 6 Social.Democrats                   3.9
## 7 Green.Party                        4.5
## 8 Aontú                              2
## 9 Independents                       9.3
```

Question 5.6

The Sinn.FA.in party had the highest mean percentage of votes. In the spaghetti plot Sinn.FA.in was almost always the leading party, except for a brief period at the end of June, when Fine.Gael had a higher percentage.