Schweizer Paraplegiker Forschung

Exercise: Data visualization

Workshop – Introduction to R

1 Preparing the NHANES data

a) In this exercise, we will work with the **NHANES** data which is survey data collected by the US National Center for Health Statistics. It can be used by loading the **NHANES** package and then calling the data set by its name (**NHANES**). We first want to reduce the size of the data set. Run the commands below to prepare the data accordingly:

```
### Install and load NHANES package:
install.packages("NHANES")
library(NHANES)
```

```
### Store data under new name:
d <- NHANES
### Take only data from one year:
d <- d[d$SurveyYr=="2009_10", ]
### Remove duplicated rows:
d <- d[!duplicated(d),]
### Extract wanted variables:
d <- d[, c("HHIncomeMid", "BMI", "Age", "Gender",
           "Education", "HealthGen", "PhysActive")]
```

b) Take a look at the preprocessed data to get an impression of its variables. You can find an explanation of the variables' meaning on the help page (**?NHANES**).

2 The plot function

a) Create a simple scatter plot with the prepared **NHANES** data in which you plot the age of participants on the x-axis vs. their BMI on the y-axis.

b) Extend the plot command with further arguments to change the appearance of the figure. Add a title, change the used plotting symbol. Additionally, color the points according to whether participants are physically active or not (**PhysActive** variable).

c) **Extra:** The points are now colored according to the physical activity of the people. More specifically, the points are colored according to the level numbers of the **PhysActive** factor (see slides). We can use the **legend** function to manually add a legend to the plot. In the **legend** function we can specify the legend text with the **legend** argument and the legend symbols and colors with the **pch** and **col** arguments, respectively. The commands below create the figure again and add a legend with the correct color assignment. Run the code and try to understand the individual commands inside the function call.

```
plot(BMI ~ Age, data=d, col=d$PhysActive, pch=19, cex=1.5, main="BMI vs Age")
legend('topleft',
       title = 'Physically active:',
       legend = levels(d$PhysActive),
       col = 1:nlevels(d$PhysActive),
       pch=19)
```

3 Creating a boxplot

a) Using the `NHANES` data, create a boxplot in which you plot the distribution of the income (`HHIncomeMid` variable) versus the level of education (`Education` factor). Does the figure suggest a relation between the two variables?

b) We can again use additional arguments to adapt the appearance of the figure. Try for example to change the color of the boxplots and add a title.

4 Setting graphical parameters

a) In R, we can change graphical parameters to further control R's behavior when creating figures. For example, the `mfrow` parameter determines how many figures are printed inside a plot window. Normally, it is set to a value of `c(1,1)` which means that in a plot window there will be 1 row and 1 column, resulting in only one figure in the window. To plot multiple figures next to each other we can change the value of the `mfrow` parameter. To do this, we have to use the `par` function which controls all graphical parameters (see `?par`). With the following command we can change the `mfrow` value to 1 row and 2 columns:

```
par(mfrow=c(1, 2))
```

Run the command and create again the scatter plot and the boxplot from the previous exercise. How are they now placed inside the plot window? Try out different values for the `mfrow` parameter and observe its effect.

In the end, one might want to reset `mfrow` to its original value:

```
par(mfrow=c(1, 1))
```

b) **Extra:** Another graphical parameter that is often used is the `mar` parameter. It defines the sizes of the margins around a figure. With the following code we can check its current value:

```
par()$mar

## [1] 5.1 4.1 4.1 2.1
```

As can be read in the `par` helppage (`?par`) `mar` determines the sizes of the bottom, left, top and right figure margins (in this order). This can for example be useful when we have text in a figure that is too large for the figure margins. The following command creates again the boxplot from the previous exercise but uses the `las` argument to rotate the axis labels by 90 degrees (sometimes necessary with very large labels). Run the command and look at the resulting figure.

```
plot(HHIncomeMid ~ Education, data = d, col=4, main='Income vs Education', las=2)
```

As we can see, some of the labels are cut of by the bottom margin (and they also overlay with the axis titles, but we'll ignore this here). Try to change the `mar` parameter to increase the size of the bottom margin, making all axis labels readable.

5 Plotting multiple settings

a) As we saw in the lecture, `ggplot2` is a handy R package to create complex figures. Try to use the `ggplot2` package to create again boxplots of the income vs the education level but separated into two figures, one only showing data from men and the other showing data from women.

b) **Extra:** Although a bit more effortful, we now have all the tools to create such a figure ourselves without the help of `ggplot2`. Try to recreate the figure by 1) Splitting the data into two data frames, one containing the men and one the women data and 2) use the `plot` function together with the `mfrow` argument to plot the two figures next to each other like in the ggplot version.