# CorrelCon: Introduction to R

Mirka Henninger

## 1 Let's get started...

First, we read in the data from an `rda` file (typical for R). Of course, also other data format (e.g., Excel sheets, text files) can be read in. In RStudio, you can also use the `Import Dataset` button in the Environment window. It guides you through, and you can later paste the code into your script.

```
> load("student_pisa.rda")
```

We get a first overview.

```
> # first six rows
> head(dat)
```

```
     X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X16 X17 X18 X19 X20 X21
743   1  0  0  1  1  1  1  0  0   1   1   1   0   0   0   1   0   1   0   0   1
801   0  1  0  1  1  1  0  0  0   1   0   1   1   0   1   0   0   1   0   1   0
3747  0  0  0  1  0  1  0  0  1   1   1   1   0   0   1   1   0   0   0   0   0
4273  0  0  1  1  1  0  1  0  1   0   0   1   0   0   1   1   1   0   0   0   1
4366  1  1  1  1  1  1  1  1  1   1   1   1   1   0   1   1   1   1   0   1   0
5177  1  0  0  1  0  1  1  1  0   1   0   1   0   0   1   1   0   1   0   1   1
     X22 X23 X24 X25 X26 X27 X28 X29 X30 X31 X32 X33 X34 X35 X36 X37 X38 X39
743    1   1   1   0   0   0   0   0   0   1   1   1   1   1   1   1   1   0
801    1   1   1   0   1   1   0   0   1   0   1   1   0   0   1   1   1   0
3747   0   0   1   1   0   0   1   1   1   0   0   1   1   1   1   1   1   0
4273   1   0   1   1   0   0   1   1   0   0   0   1   1   1   1   0   1   1
4366   1   1   1   1   1   0   0   1   1   1   1   0   1   1   1   1   1   1
5177   0   1   0   0   0   1   1   0   1   0   1   1   1   1   1   0   1   1
     X40 X41 X42 X43 X44 X45 gender age semester elite      spon
743    1   0   1   1   1   1 female  21        3    no 1-3/month
801    1   1   1   1   1   1   male  20        1    no  4-5/week
3747   1   0   1   1   1   1 female  25        9    no 1-3/month
4273   1   1   1   0   1   1   male  27       10    no     never
4366   1   0   0   1   1   1   male  24        8    no    1/week
5177   1   1   1   0   1   1   male  20        1   yes 1-3/month
```

```
> # last six rows
> tail(dat)
```

```
       X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X16 X17 X18 X19 X20
690775  0  1  0  1  1  1  1  1  0   0   0   0   1   0   0   0   1   1   0   0
691336  1  1  0  1  1  1  1  0  0   1   0   1   0   0   1   0   1   1   0   1
691506  0  1  1  1  1  0  1  1  1   0   0   1   0   0   1   1   0   0   1   0
691527  0  1  0  1  0  1  1  0  0   1   1   1   0   0   1   1   1   0   1   1
691849  0  0  0  1  1  1  1  0  0   0   0   0   0   0   1   0   1   0   0   0
691972  1  1  0  1  1  1  1  1  0   1   1   1   0   0   0   1   0   1   0   0
       X21 X22 X23 X24 X25 X26 X27 X28 X29 X30 X31 X32 X33 X34 X35 X36 X37 X38
690775   0   1   1   1   1   0   1   0   0   1   1   0   1   0   1   0   1   0
691336   1   1   0   0   0   1   1   0   0   1   1   1   1   1   1   1   1   1
691506   1   1   1   1   1   1   1   1   0   1   1   1   0   1   1   1   1   1
691527   1   1   1   1   1   1   1   0   0   0   1   1   1   0   1   1   1   1
691849   1   1   1   1   0   0   1   1   1   0   0   0   1   1   1   1   0   1
691972   0   1   0   1   0   1   1   1   0   0   0   1   1   1   1   1   1   1
       X39 X40 X41 X42 X43 X44 X45 gender age semester elite      spon
690775   0   0   1   1   1   1   1   male  22        2    no     daily
691336   1   1   1   1   1   0   1 female  23        9    no     daily
691506   0   1   1   1   1   1   1   male  21        2   yes 1-3/month
691527   1   0   0   1   1   1   1   male  23        6   yes    1/week
691849   1   1   1   1   0   0   0 female  27        9    no 1-3/month
691972   1   1   0   1   1   0   1 female  22        3    no  2-3/week
```

```
> # overview of the variables
> colnames(dat)
```

```
 [1] "X1"       "X2"       "X3"       "X4"       "X5"       "X6"
 [7] "X7"       "X8"       "X9"       "X10"      "X11"      "X12"
[13] "X13"      "X14"      "X15"      "X16"      "X17"      "X18"
[19] "X19"      "X20"      "X21"      "X22"      "X23"      "X24"
[25] "X25"      "X26"      "X27"      "X28"      "X29"      "X30"
[31] "X31"      "X32"      "X33"      "X34"      "X35"      "X36"
[37] "X37"      "X38"      "X39"      "X40"      "X41"      "X42"
[43] "X43"      "X44"      "X45"      "gender"   "age"      "semester"
[49] "elite"    "spon"
```

```
> str(dat)
```

```
'data.frame':        1075 obs. of  50 variables:
 $ X1      : num  1 0 0 0 1 1 0 1 1 0 ...
 $ X2      : num  0 1 0 0 1 0 1 1 0 0 ...
 $ X3      : num  0 0 0 1 1 0 0 1 0 0 ...
 $ X4      : num  1 1 1 1 1 1 1 0 1 1 ...
 $ X5      : num  1 1 0 1 1 0 1 1 0 1 ...
 $ X6      : num  1 1 1 0 1 1 1 1 0 0 ...
 $ X7      : num  1 0 0 1 1 1 0 1 1 0 ...
 $ X8      : num  0 0 0 0 1 1 0 0 0 1 ...
 $ X9      : num  0 0 1 1 1 0 0 0 1 0 ...
```

```
$ X10     : num  1 1 1 0 1 1 1 1 1 1 ...
$ X11     : num  1 0 1 0 1 0 0 0 0 1 ...
$ X12     : num  1 1 1 1 1 1 1 1 1 1 ...
$ X13     : num  0 1 0 0 1 0 1 0 0 0 ...
$ X14     : num  0 0 0 0 0 0 1 0 0 0 ...
$ X15     : num  0 1 1 1 1 1 1 0 1 0 ...
$ X16     : num  1 0 1 1 1 1 1 1 1 0 ...
$ X17     : num  0 0 0 1 1 0 0 0 1 0 ...
$ X18     : num  1 1 0 0 1 1 0 0 0 0 ...
$ X19     : num  0 0 0 0 0 0 0 0 0 0 ...
$ X20     : num  0 1 0 0 1 1 1 0 1 1 ...
$ X21     : num  1 0 0 1 0 1 1 0 0 0 ...
$ X22     : num  1 1 0 1 1 0 0 1 1 1 ...
$ X23     : num  1 1 0 0 1 1 0 0 1 1 ...
$ X24     : num  1 1 1 1 1 0 1 0 1 0 ...
$ X25     : num  0 0 1 1 1 0 1 0 0 0 ...
$ X26     : num  0 1 0 0 1 0 0 0 1 0 ...
$ X27     : num  0 1 0 0 0 1 1 1 1 0 ...
$ X28     : num  0 0 1 1 0 1 1 1 0 1 ...
$ X29     : num  0 0 1 1 1 0 0 1 0 1 ...
$ X30     : num  0 1 1 0 1 1 1 0 1 0 ...
$ X31     : num  1 0 0 0 1 0 0 1 1 1 ...
$ X32     : num  1 1 0 0 1 1 0 0 1 1 ...
$ X33     : num  1 1 1 1 0 1 1 1 1 1 ...
$ X34     : num  1 0 1 1 1 1 1 0 1 1 ...
$ X35     : num  1 0 1 1 1 1 1 1 1 1 ...
$ X36     : num  1 1 1 1 1 1 1 0 1 1 ...
$ X37     : num  1 1 1 0 1 0 1 1 1 1 ...
$ X38     : num  1 1 1 1 1 1 1 1 1 0 ...
$ X39     : num  0 0 0 1 1 1 0 0 0 0 ...
$ X40     : num  1 1 1 1 1 1 1 1 1 1 ...
$ X41     : num  0 1 0 1 0 1 0 0 1 0 ...
$ X42     : num  1 1 1 1 0 1 1 1 1 1 ...
$ X43     : num  1 1 1 0 1 0 0 1 1 1 ...
$ X44     : num  1 1 1 1 1 1 0 0 0 0 ...
$ X45     : num  1 1 1 1 1 1 1 1 1 1 ...
$ gender  : Factor w/ 2 levels "female","male": 1 2 1 2 2 2 2 2 2 1 ...
$ age     : num  21 20 25 27 24 20 22 22 21 20 ...
$ semester: Ord.factor w/ 11 levels "1"<"2"<"3"<"4"<..: 3 1 9 10 8 1 2 2 1 1 ...
$ elite   : Factor w/ 2 levels "no","yes": 1 1 1 1 1 2 1 1 1 1 ...
$ spon    : Ord.factor w/ 7 levels "never"<"<1/month"<..: 3 6 3 1 4 3 1 4 4 3 ...
```

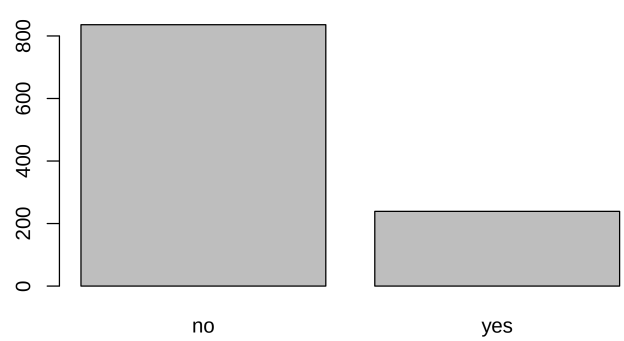
## 2 Get some summary and plots

R provides some basic built-in function that return summmary or visualizations. Here for the `elite` (coded as a factor). The `summary` function is a generic function. It checks the data type and then returns what it thinks is the best output (here summary is a table). Alternatively, the `table` function works fine as well.

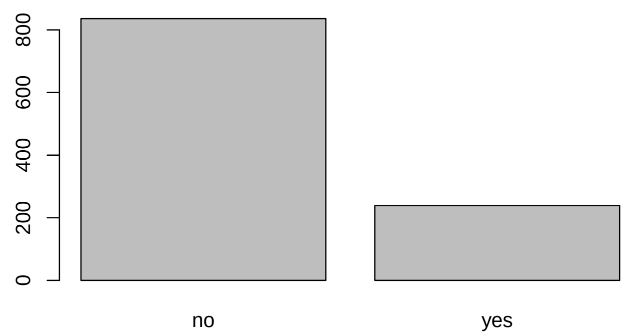
```
> summary(dat$elite)

 no yes
836 239

> table(dat$elite)

 no yes
836 239
```

When we call `plot` for a factor, we get a barplot (or we use `barplot` directly). But we can also easily get a `pie` chart!

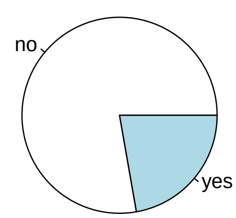
```
> plot(dat$elite)
```



```
> # alternative
> barplot(table(dat$elite))
```

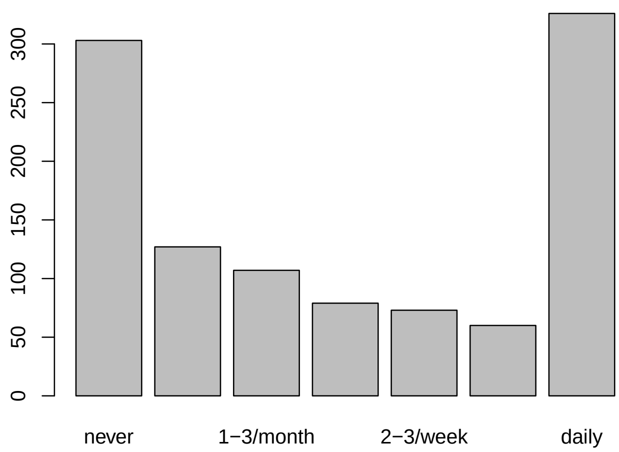



```
> pie(table(dat$elite))
```



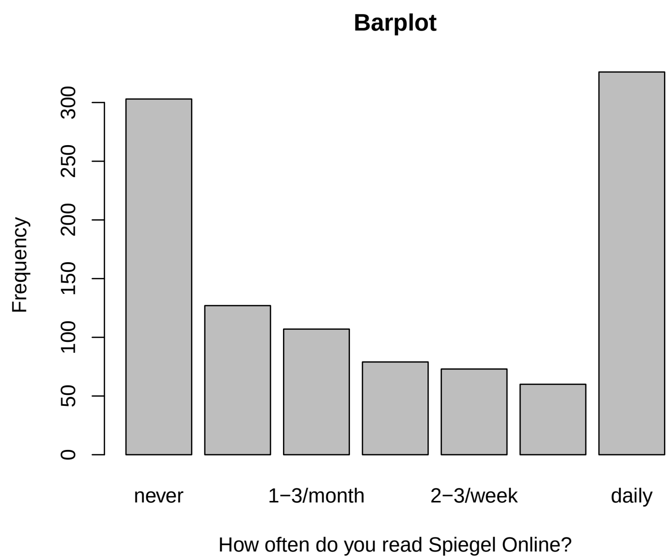
The same applies to the variable `spon` (coded as an ordered factor). As `spon` is ordered, the order of the values in the plot is not alphabetically, but ordered as the factor order.

```
> summary(dat$spon)

    never  <1/month 1-3/month    1/week  2-3/week  4-5/week     daily
      303       127       107        79        73        60       326

> table(dat$spon)

    never  <1/month 1-3/month    1/week  2-3/week  4-5/week     daily
      303       127       107        79        73        60       326

> plot(dat$spon)
```

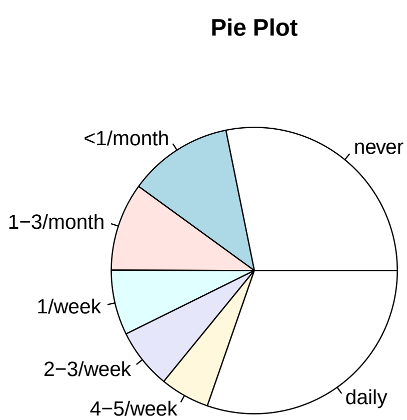


We can also add titles or axis labels

```
> barplot(table(dat$spon),
+         xlab = "How often do you read Spiegel Online?", ylab = "Frequency",
+         main = "Barplot")
```

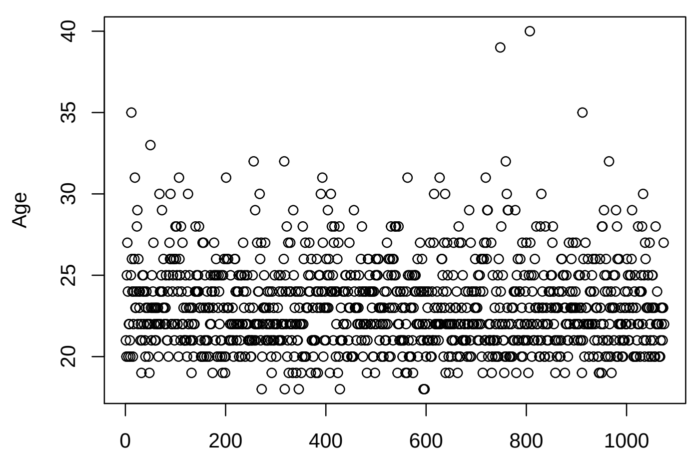



```
> pie(table(dat$spon), main = "Pie Plot")
```

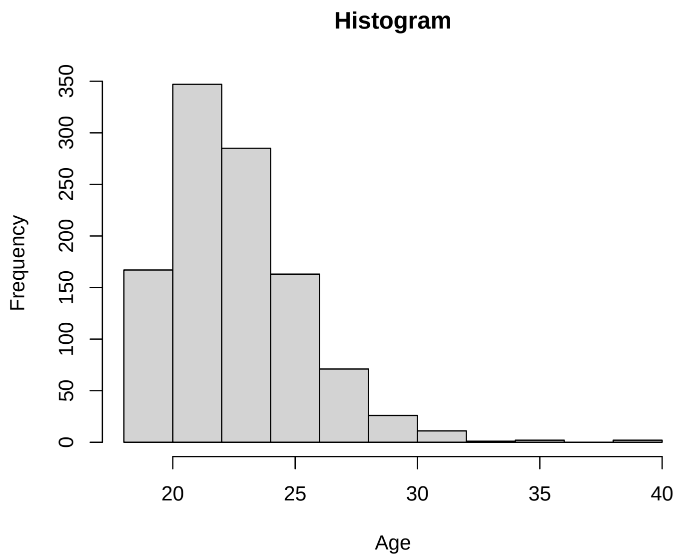


For `age` (continuous variable), we get different plots. But also here, *R* tries to find the best solution for us, when we just use `summary` or `plot`. But maybe a scatterplot is not what we want, so ask for a `histogram` instead.

```
> table(dat$age)

 18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  35  39  40
  6  37 124 159 188 149 136 102  61  43  28  15  11   7   4   1   2   1   1

> summary(dat$age)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   18.0    21.0    23.0    23.1    25.0    40.0

> plot(dat$age,
+      xlab = "", ylab = "Age")
```



```
> hist(dat$age,
+       xlab = "Age",
+       main = "Histogram")
```



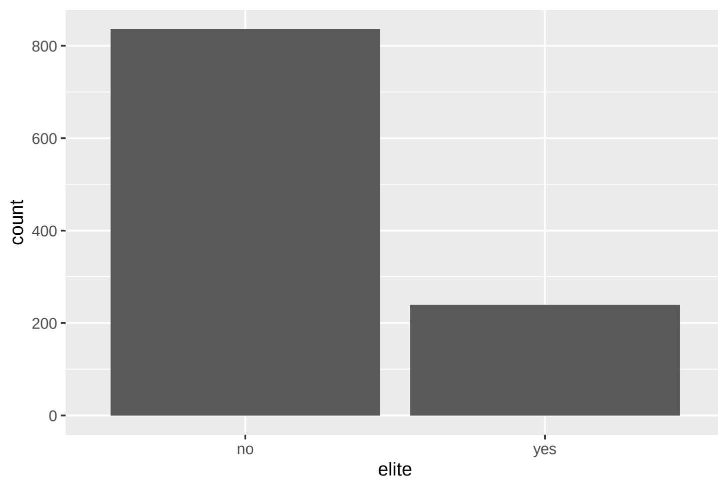
## 2.1 In another universe

The plot functions that we have used until now are `R`'s built-in functions. Alternatively, we can use functions from packages from the `tidyverse` by Hadley Wickham. He has written many functions that some people find easier to use than `R`'s built-in functions.

```
> # install.packages("ggplot2")
> library(ggplot2)
```

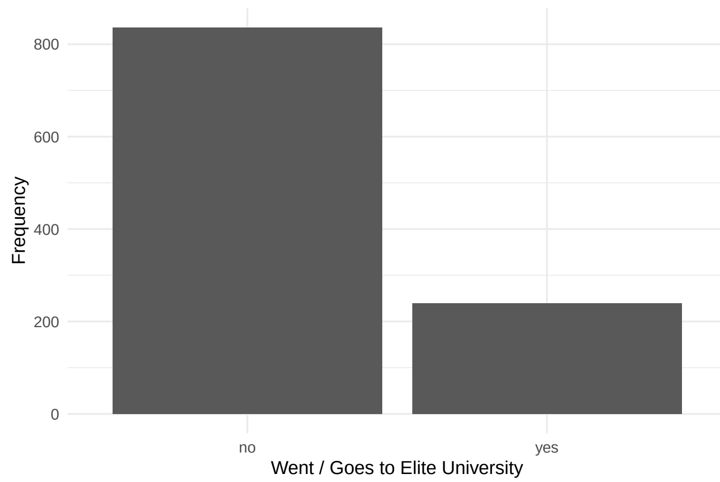
The plotting mechanism for `ggplot` works very differently from `base R`. We first define what will be on the x-axis and y-axis, and then add what kind of plot we want.

```
> ggplot(dat, aes(x = elite)) +
+   geom_bar()
```



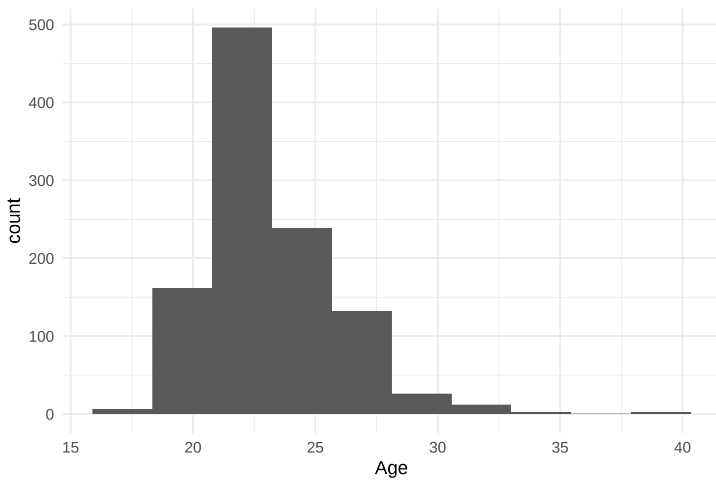
We can add more options, such as axis labels and another background theme

```
> ggplot(dat, aes(x = elite)) +
+   geom_bar() +
+   xlab("Went / Goes to Elite University") +
+   ylab("Frequency") +
+   theme_minimal()
```



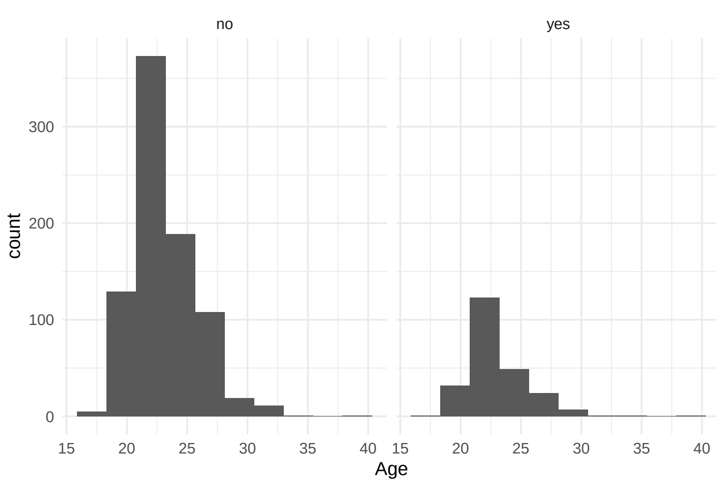
Similarly for the continuous variable `age`, we can ask for a histogram

```
> ggplot(dat, aes(x = age)) +
+   geom_histogram(bins = 10) + # with "bins", we can set the number of bins
+   xlab("Age") +
+   theme_minimal()
```




With `ggplot`, you can very easily create faceted plots. There are very helpful for data exploration. Here, we explore whether the distribution of `Age` is different between people that went/go to a elite university:

```
> ggplot(dat, aes(x = age)) +
+   geom_histogram(bins = 10) +
+   facet_wrap(~elite) +
+   xlab("Age") +
+   theme_minimal()
```



Some people like the `base R` way of plotting more, some the `ggplot` way of plotting. I use both, with a tendency to `base R` because I think it is more flexible. Some people find `ggplot` easier to use or to get used to. It is up to you to decide which one you choose, both are great and provide much more options than we could cover in this workshop! Maybe you find these cheatsheets helpful when you work with visualizations in the future:

- base R: http://publish.illinois.edu/johnrgallagher/files/2015/10/BaseGraphicsCheatsheet.pdf
- ggplot: https://statsandr.com/blog/files/ggplot2-cheatsheet.pdf

# 3 Subsetting & computing new variables

To compute a sum score of items correct, we have to extract the test items. There is several options to do so:

```
> # subset data
> testItems <- dat[, 1:45]
> # alternativ:
> testItems <- dat[, paste("X", 1:45, sep = "")]
> # alternativ:
> testItems <- dat[, grep("X", colnames(dat))]
```

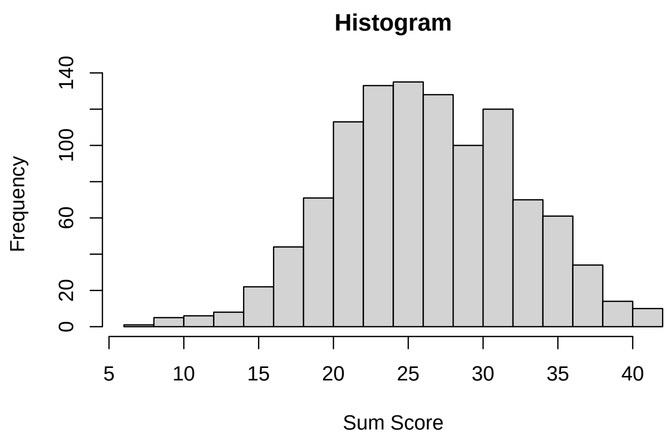
Then we can compute a sumscore. We use the `rowSums` function, that computes a sum row-wise (there is also `colSums`):

```
> dat$sumscore <- rowSums(testItems)
> head(dat)
```

```
     X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 X13 X14 X15 X16 X17 X18 X19 X20 X21
743   1  0  0  1  1  1  1  0  0   1   1   1   0   0   0   1   0   1   0   0   1
801   0  1  0  1  1  1  0  0  0   1   0   1   1   0   1   0   0   1   0   1   0
3747  0  0  0  1  0  1  0  0  1   1   1   1   0   0   1   1   0   0   0   0   0
4273  0  0  1  1  1  0  1  0  1   0   0   1   0   0   1   1   1   0   0   0   1
4366  1  1  1  1  1  1  1  1  1   1   1   1   1   0   1   1   1   1   0   1   0
5177  1  0  0  1  0  1  1  1  0   1   0   1   0   0   1   1   0   1   0   1   1
     X22 X23 X24 X25 X26 X27 X28 X29 X30 X31 X32 X33 X34 X35 X36 X37 X38 X39
743    1   1   1   0   0   0   0   0   0   1   1   1   1   1   1   1   1   0
801    1   1   1   0   1   1   0   0   1   0   1   1   0   0   1   1   1   0
3747   0   0   1   1   0   0   1   1   1   0   0   1   1   1   1   1   1   0
4273   1   0   1   1   0   0   1   1   0   0   0   1   1   1   1   0   1   1
4366   1   1   1   1   1   0   0   1   1   1   1   0   1   1   1   1   1   1
5177   0   1   0   0   0   1   1   0   1   0   1   1   1   1   1   0   1   1
     X40 X41 X42 X43 X44 X45 gender age semester elite      spon sumscore
743    1   0   1   1   1   1 female  21        3    no 1-3/month       27
801    1   1   1   1   1   1   male  20        1    no  4-5/week       27
3747   1   0   1   1   1   1 female  25        9    no 1-3/month       24
4273   1   1   1   0   1   1   male  27       10    no     never       26
4366   1   0   0   1   1   1   male  24        8    no    1/week       37
5177   1   1   1   0   1   1   male  20        1   yes 1-3/month       28
```

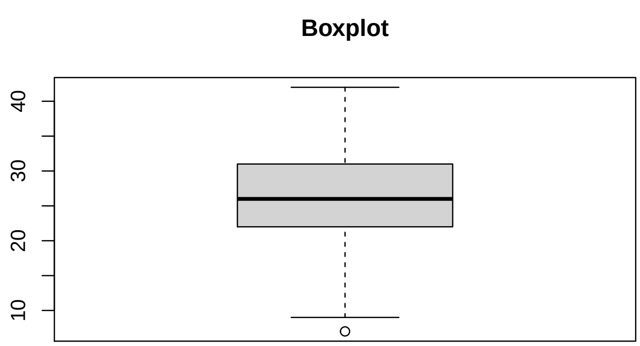
Now again we can explore the sum score by looking at a histogram

```
> hist(dat$sumscore,
+       xlab = "Sum Score",
+       main = "Histogram",
+       breaks = 20)
```



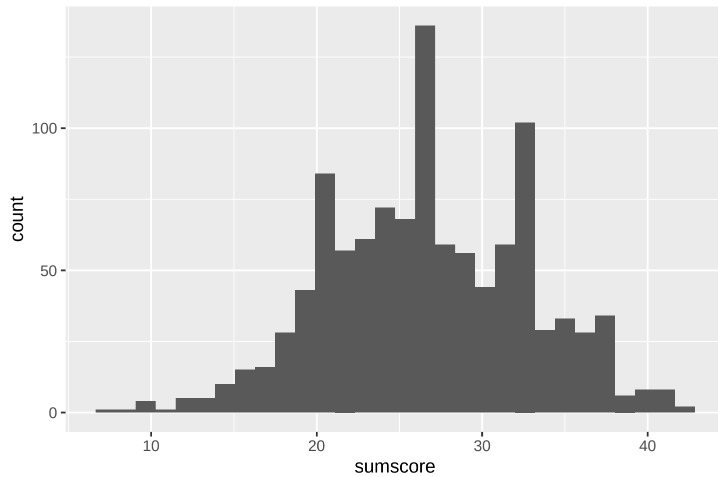

or a boxplot of the new variable.

```
> boxplot(dat$sumscore,
+          main = "Boxplot")
```

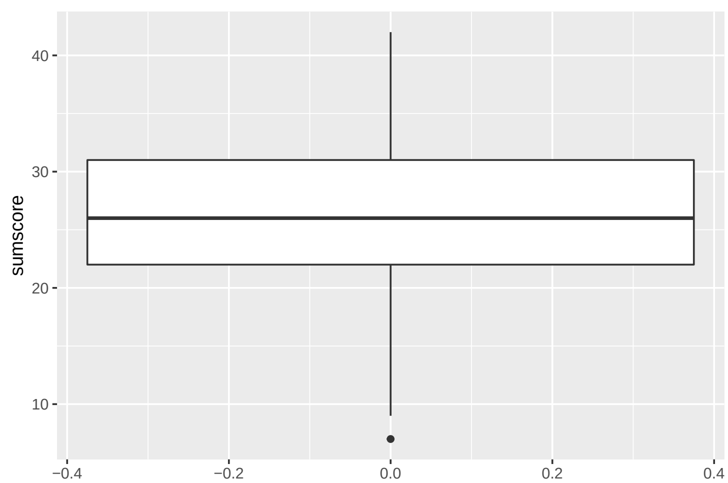


Or we use `ggplot` for plotting:

```
> ggplot(dat, aes(x = sumscore)) +
+   geom_histogram()
```



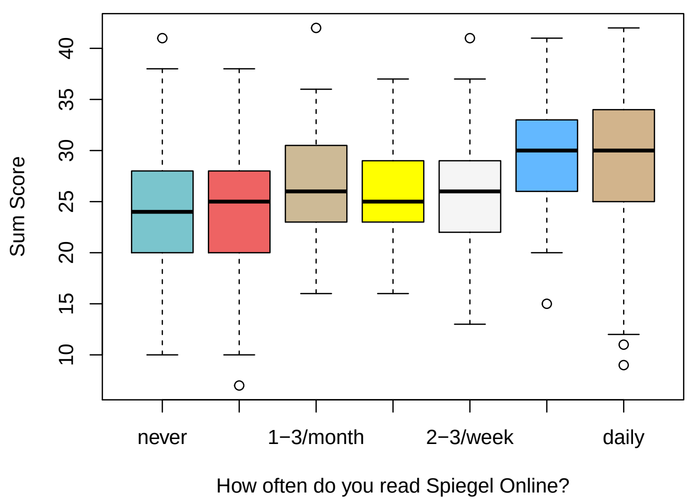

```
> ggplot(dat, aes(y = sumscore)) +
+   geom_boxplot()
```



## 4 Bivariate Plots

Now, we want to look at bivariate plots to get more insights into the data. For example, boxplots of the sumscore for each group of Spiegel Online readers (`spon`). With the `colors` function, we can get the names of many nice colors. There are also color palettes, or you use HEX or rgb codes.

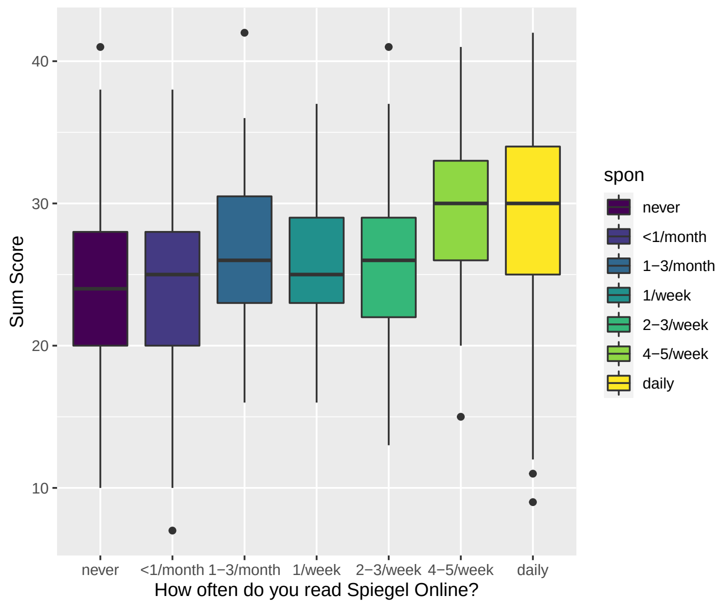
```
> head(colors())

[1] "white"          "aliceblue"      "antiquewhite"   "antiquewhite1"
[5] "antiquewhite2"  "antiquewhite3"

> # boxplot for each group of spon readers
> boxplot(sumscore ~ spon, data = dat,
+         col = c("cadetblue3", "indianred2", "wheat3",
+                 "yellow1", "whitesmoke", "steelblue1", "tan"),
+         ylab = "Sum Score",
+         xlab = "How often do you read Spiegel Online?")
```




From a visual inspection, it seems as if regular readers of Spiegel Online had better results in the test.

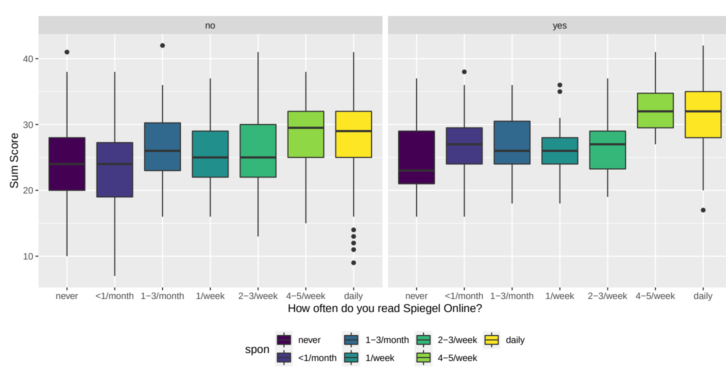
And a ggplot graphic (it uses pretty standard palettes for the fill colors):

```
> ggplot(dat, aes(x = spon, y = sumscore, fill = spon)) +
+   geom_boxplot() +
+   ylab("Sum Score") +
+   xlab("How often do you read Spiegel Online?")
```




As I said above, it is so easy to use facetting with ggplot:

```
> ggplot(dat, aes(x = spon, y = sumscore, fill = spon)) +
+   geom_boxplot() +
+   facet_wrap(~elite) +
+   theme(legend.position = "bottom") +
+   ylab("Sum Score") +
+   xlab("How often do you read Spiegel Online?")
```



# 5 Linear regression model

We want to fit a linear regression model, using the variables `age` and `elite` as predictor variables, and the `sumscore` as a dependent variable. Of course, we could add more predictors, or also interaction effects (using `*` or `:`). The `summary` function gives us the best output for the regression object.

```
> # fit model and save it in the object "lmod"
> lmod <- lm(sumscore ~ age + elite, data = dat)
> summary(lmod)

Call:
lm(formula = sumscore ~ age + elite, data = dat)

Residuals:
     Min       1Q   Median       3Q      Max 
-19.1326  -4.1326  -0.1326   4.4577  15.4577 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 21.62635    1.56811  13.791  < 2e-16 ***
age          0.20483    0.06722   3.047 0.002368 ** 
eliteyes     1.59303    0.43929   3.626 0.000301 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1 

Residual standard error: 5.988 on 1072 degrees of freedom
Multiple R-squared:  0.02017,	Adjusted R-squared:  0.01834 
F-statistic: 11.04 on 2 and 1072 DF,  p-value: 1.804e-05
```

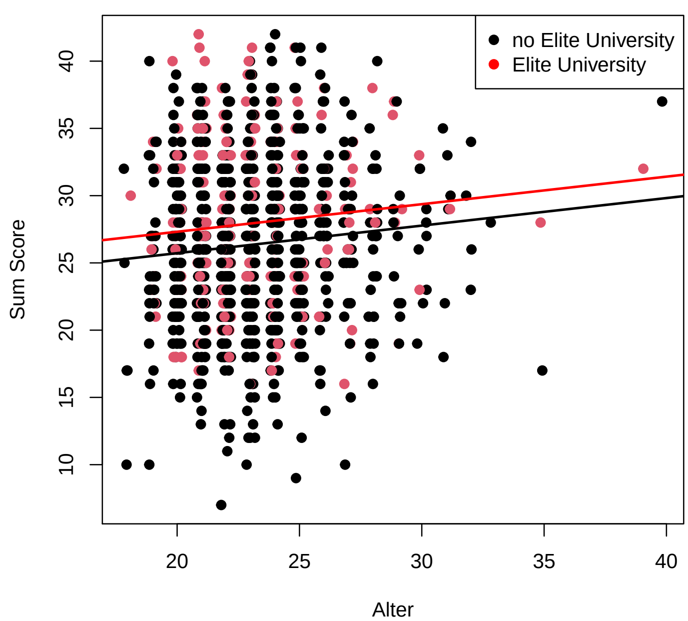
We want to visualize the results of the regression. So we create a scatterplot that plot `age` against `sumscore` and the points are colored by `elite`.

First, we extract the regression weights from the regression object.

```
> # extract regression coefficients
> intercept <- coefficients(lmod)[1]
> age <- coefficients(lmod)[2]
> eliteyes <- coefficients(lmod)[3]
```

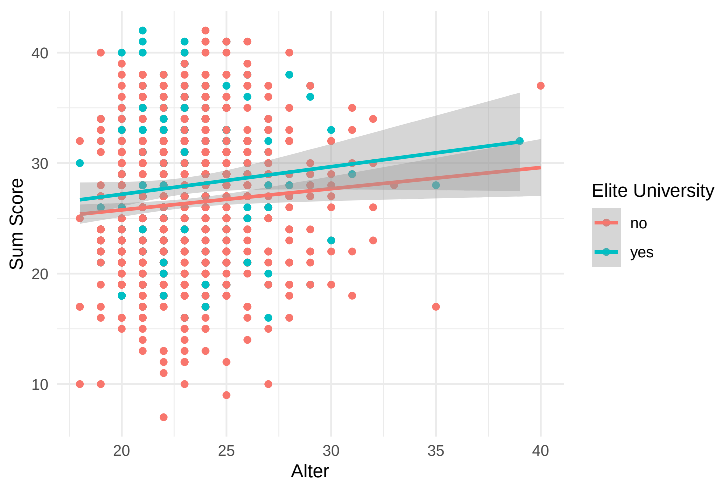
Then, we can create a scatterplot. As age is an integer variable, the scatterplot looks strange and the points are on top of each other. Using `jitter`, we can randomly jitter the points so that we can see them all. We add a `legend` to the plot, so we know what the colors of the points mean. Then, we can add the regression lines, one for students from an elite university, one from non-elite universities.

```
> # create plot
> plot(jitter(dat$age), dat$sumscore,
+      col = dat$elite,
+      pch = 19,
+      xlab = "Alter", ylab = "Sum Score")
> # add legend
> legend("topright", # Position of the legend
+        col = c("black", "red"),
+        legend = c("no Elite University", "Elite University"),
+        pch = 19)
> # add regression lines
> abline(a = intercept, b = age,
+        lwd = 2) # line width
> abline(a = intercept + eliteyes, b = age,
+        col = "red",
+        lwd = 2)
```



Of course we can create a similar plot using ggplot:

```
> # create plot
> ggplot(dat, aes(x = age, y = sumscore,
+                  # unintuitively the argument is no named color,
+                  # not fill as for the boxplot above
+                  color = elite)) +
+   geom_point() +
+   # this how ggplot automatically adds the regression lines
+   geom_smooth(method = "lm",
+                formula = y ~ x) +
+   xlab("Alter") +
+   ylab("Sum Score") +
+   theme_minimal() +
+   # also quite unintuitively: to change legend title, the syntax has
+   # nothing to do with "legend":
+   labs(color = "Elite University")
```




With `ggplot`, I also search for tutorial (e.g., https://www.datanovia.com/en/blog/ggplot-legend-title-position-and-labels/ or https://statisticsglobe.com/add-regression-line-to-ggplot2-plot-in-r) to adjust labels, legends, add regression lines, because in the `tidyverse`, the is a syntax for everything, but often (I find it) not easy to guess.

When we want to save the `R` `base` plot in an external file, we must open a device before we draw the plot, and close it afterwards. Here, we export a `png`, but of course other formats are possible. If you want to know which, click on `png` and press `F1`. alternatively run `?png` in the Console.

```
> # open device
> png("myRegressionPlot.png", height = 500, width = 500)
> # create plot
> plot(jitter(dat$age), dat$sumscore,
+      col = dat$elite,
+      pch = 19,
+      xlab = "Alter", ylab = "Sum Score")
> # add legend
> legend("topright", # Position of the legend
+        col = c("black", "red"),
+        legend = c("no Elite University", "Elite University"),
+        pch = 19)
> # add regression lines
> abline(a = intercept, b = age,
+        lwd = 2) # line width
> abline(a = intercept + eliteyes, b = age,
+        col = "red",
+        lwd = 2)
> # close device
> dev.off()
```

```
null device
          1
```

For `ggplot`, we first save the plot as an `R` object, and than save that object:

```
> # save plot:
> regressionPlot <-
+   ggplot(dat, aes(x = age, y = sumscore, color = elite)) +
+   geom_point() +
+   geom_smooth(method = "lm", formula = y ~ x) +
+   xlab("Alter") +
+   ylab("Sum Score") +
+   theme_minimal() +
+   labs(color = "Elite University")
> ggsave(filename = "myRegressionggPlot.png", regressionPlot)
```

Of course, also `pdf` files are possible using `filename = "myRegressionggPlot.pdf"`.