

# CorrelCon: Introduction to R

Mirka Henninger

## 1 Let's get started...

First, we read in the data. Of course, also other data format 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	0	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	0	1	1	0	1	0	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		no	daily
691336	1	1	1	1	1	0	1	female	23		no	daily
691506	0	1	1	1	1	1	1	male	21		yes	1-3/month
691527	1	0	0	1	1	1	1	male	23		yes	1/week
691849	1	1	1	1	0	0	0	female	27		no	1-3/month
691972	1	1	0	1	1	0	1	female	22		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 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 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 nice plots. Here for the `gender` (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$gender)
```

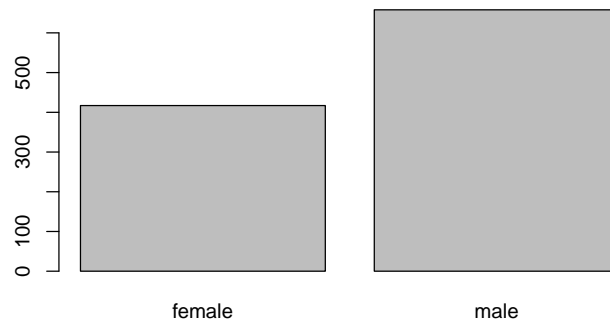
```
female  male
   417   658
```

```
> table(dat$gender)
```

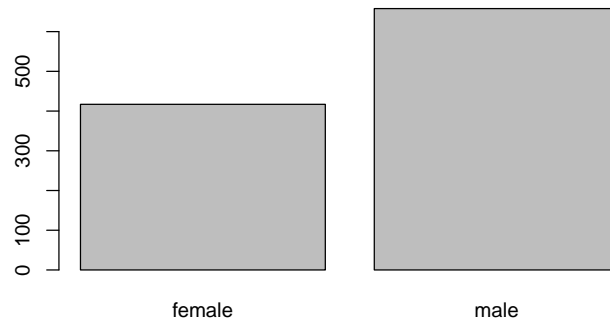
```
female  male
   417   658
```

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!

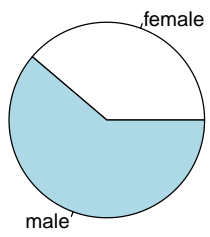
```
> plot(dat$gender)
```



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



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



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.

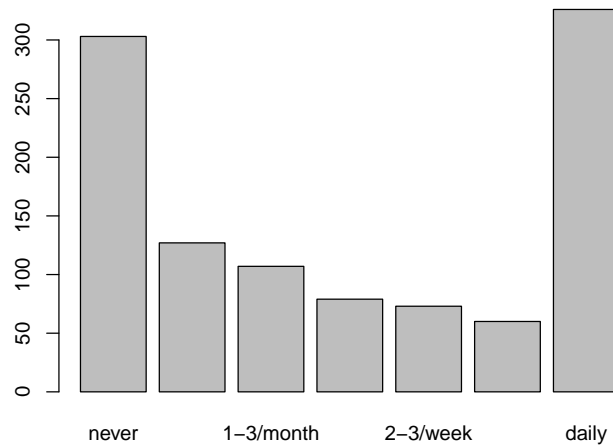
```
> summary(dat$spon)
```

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)
```

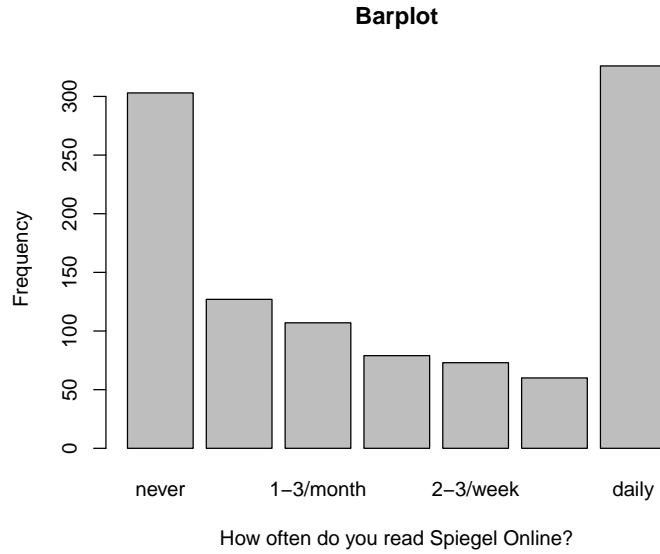
never	<1/month	1-3/month	1/week	2-3/week	4-5/week	daily
303	127	107	79	73	60	326

```
> plot(dat$spon)
```

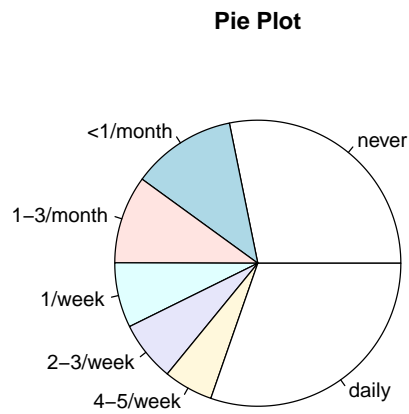


We can also add titles or axis labels

```
> 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.

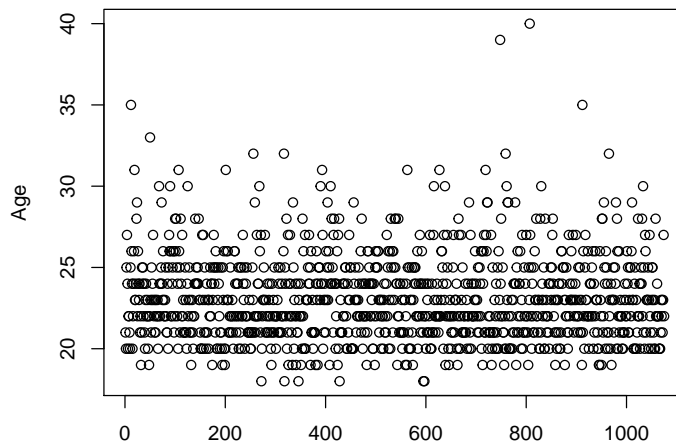
```
> table(dat$age)
```

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)
```

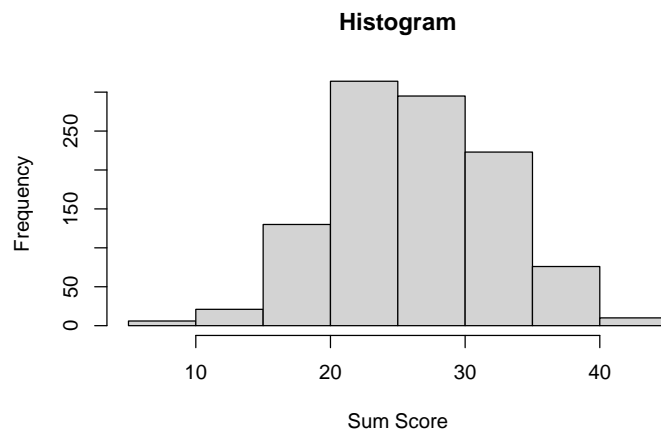
Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
18.0	21.0	23.0	23.1	25.0	40.0

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





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



### 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:

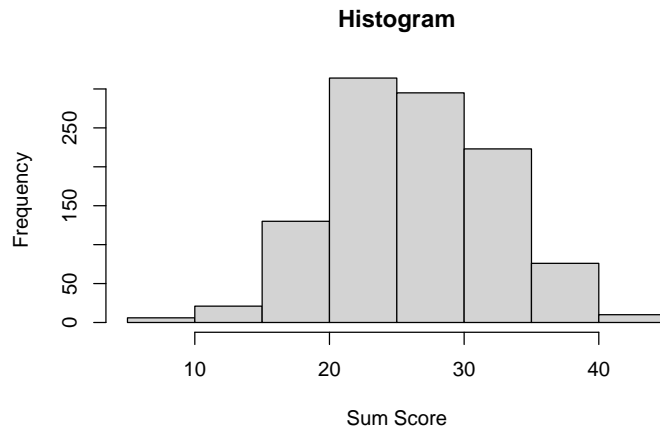
```
> # subset data
> testItems <- dat[, 1:45]
> # alternativ:
> testItems <- dat[, paste("X", 1:45, sep = "")]
> # alternativ:
> testItems <- dat[, grep("X", colnames(dat))]
```

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)
```

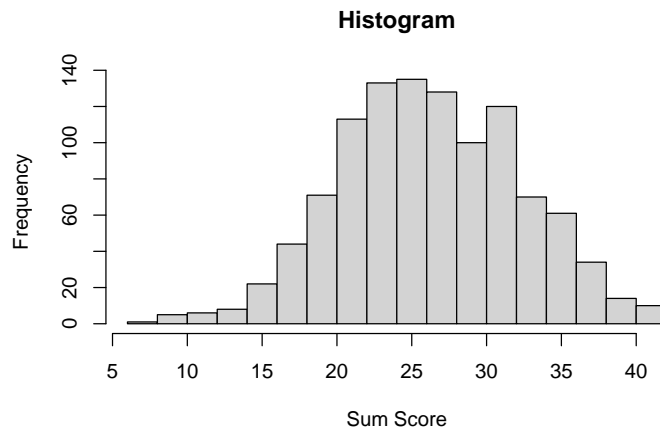
Now again we can look at a histogram,

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



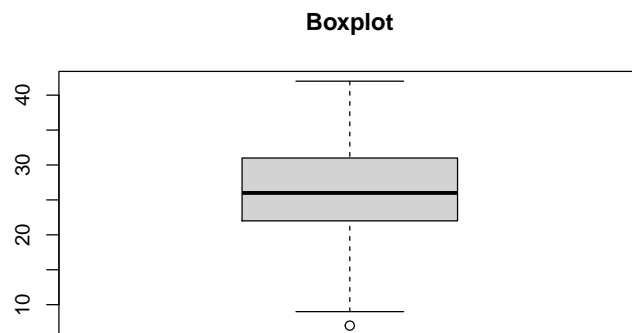
a histogram with more bins,

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



or a boxplot of the new variable.

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



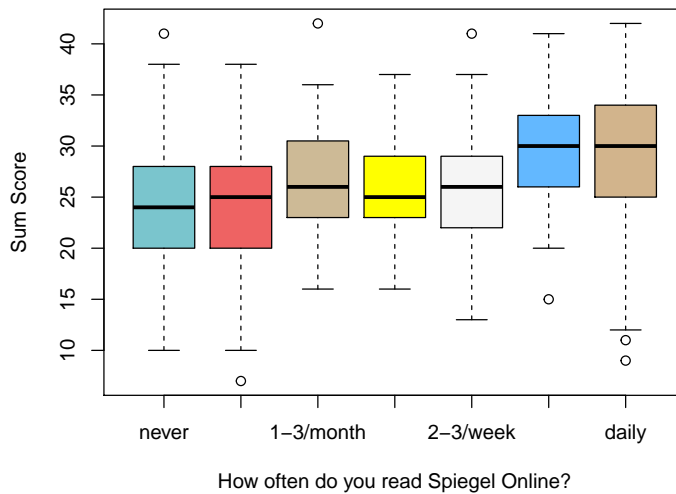
## 4 Bivariate Plots

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"

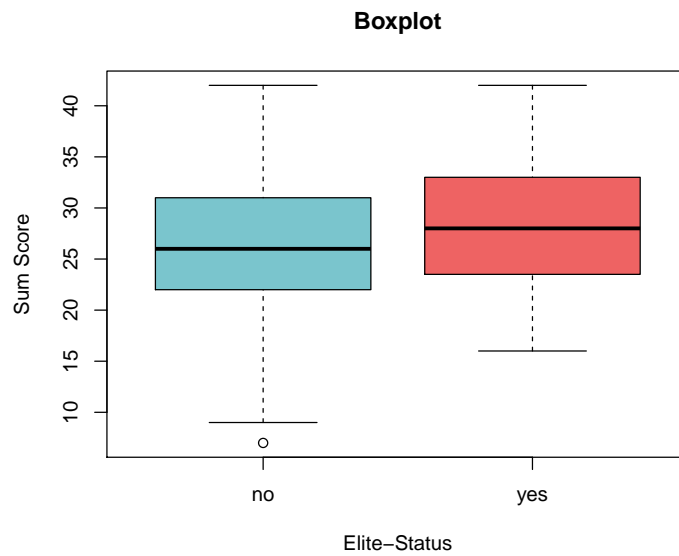
> # boxplot for each group of spon readers
> boxplot(sumscore ~ spon, data = dat,
+         col = c("cadetblue3", "indianred2", "wheat3",
+                 "yellow1", "whitesmoke", "steelblue1", "tan"),
+         ylab = "Sum Score",
+         xlab = "How often do you read Spiegel Online?")
```



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

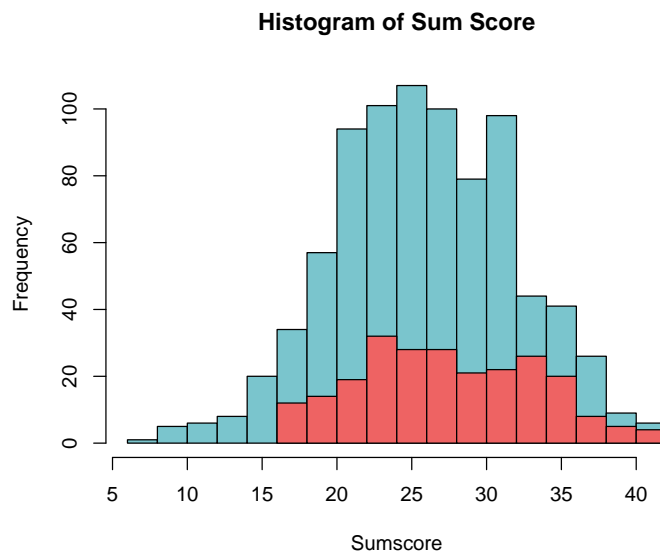
Now, let us look at boxplots for students from elite and non-elite universities

```
> boxplot(sumscore ~ elite, data = dat,  
+         col = c("cadetblue3", "indianred2"),  
+         ylab = "Sum Score",  
+         xlab = "Elite-Status", main = "Boxplot",  
+         names = c("no", "yes"))
```



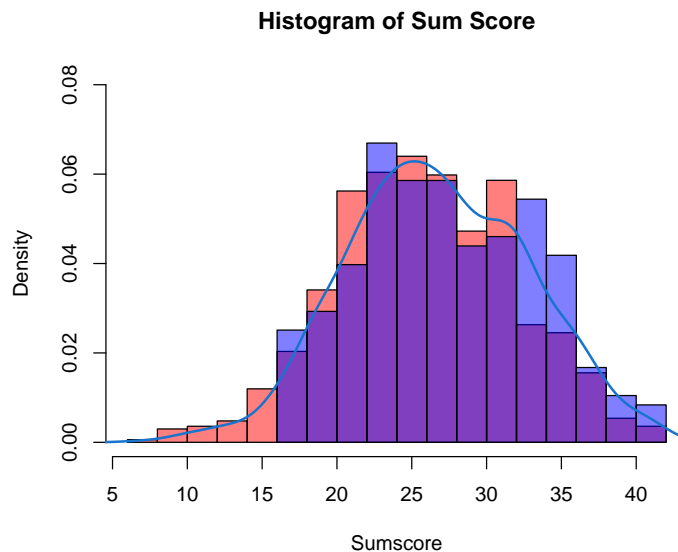
When we want to look at a histogram of the two groups, we first have to make two subsets of the data (for elite and non-elite students) and save them separately. When we make the histograms, we have to adjust the number of **breaks**, as there is a lot more non-elite students in the dataset, and they have a wider range on the dependent variable.

```
> # first create two separate datasets for elite and non_elite
> # and save them in new objects
> dat_non_elite <- dat[dat$elite == "no",]
> dat_elite <- dat[dat$elite == "yes",]
> # create histogram for elite students
> hist(dat_non_elite$sumscore,
+      col = "cadetblue3", breaks = 20,
+      xlab = "Sumscore", main = "Histogram of Sum Score")
> # create histogram for non-elite
> hist(dat_elite$sumscore, add = TRUE,
+      col = "indianred2", breaks = 10)
```



Here, we use rgb codes for colors. The last number (0.5) indicates the alpha (= transparency) of the color. This way, we can better see, where the two distributions overlap. In addition, we use `freq = FALSE`, so can get an idea of the proportion that has a certain sumscore in each of the two groups. Then, we also must adjust the range of the y-axis through the `ylim` option. Just because we can, we add a density line of the joint distribution.

```
> # create histogram for elites
> hist(dat_non_elite$sumscore, col = rgb(1,0,0,0.5),
+      breaks = 20, freq = FALSE,
+      xlab = "Sumscore", main = "Histogram of Sum Score",
+      ylim = c(0,.08))
> # add histogram for non_elites
> hist(dat_elite$sumscore, add = TRUE,
+      col = rgb(0,0,1,0.5),
+      breaks = 10, freq = FALSE)
> # add density line
> lines(density(dat$sumscore),
+      col="dodgerblue3",
+      lwd=2)
```





## 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

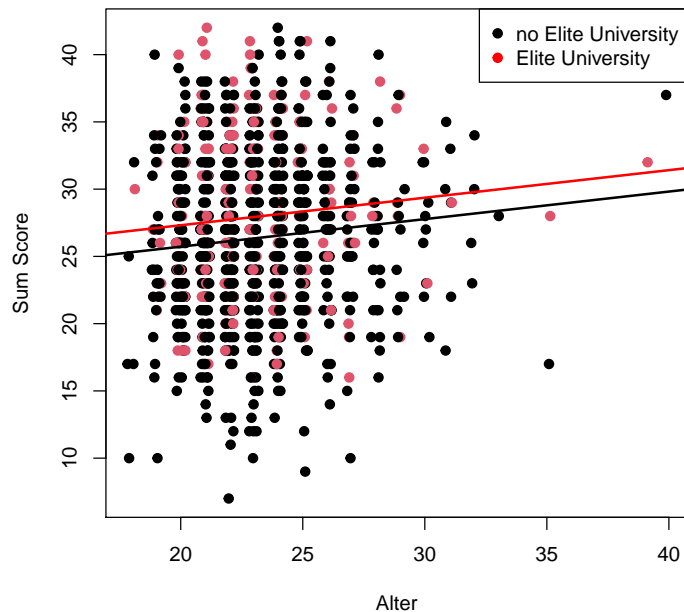
We want to visualize the results of the regression. So we create a scatterplot that plot `age` against `sumscore` and the points are colored by `elite`.

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)
```



When we want to save the 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
```