

## 2) Daten in Excel vorbereiten

**Ausgangspunkt:** Sie haben Daten in einer Excel-Datei, die Sie mit R analysieren wollen.

### Datenmappe in Excel öffnen:

Wir beginnen den Import von Daten in R mit einer bereits existierenden Excel-Datei. Die Aufbereitung von Daten kann mit Excel (oder LibreOffice, Open Office) am einfachsten vorgenommen werden. Dazu öffnet man als erstes die Datei, die die Daten enthält. In unserem Beispiel ist das eine Übung) aus dem letzten Semester (in stud.ip zu finden unter „Beispiel\_Uebung1\_ttest.xlsx“).

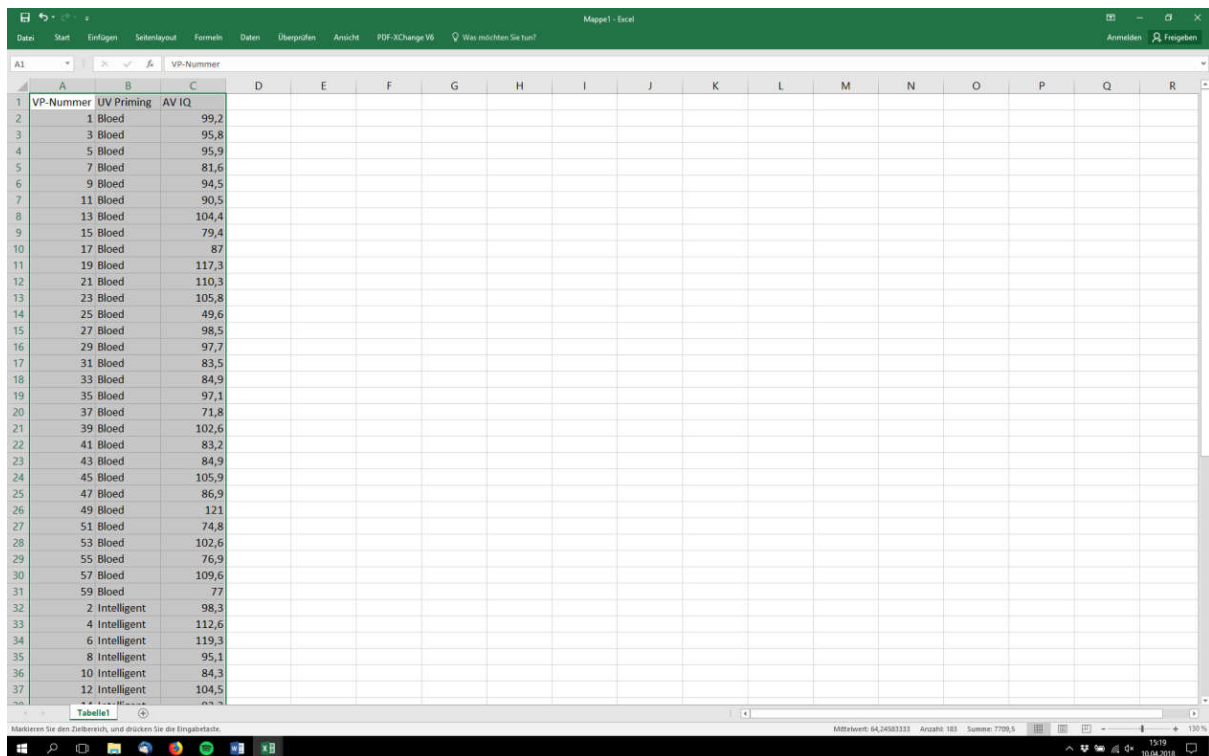
	A	B	C	D	E	F	G	H	I	J
1	VP-Nummer	UV Priming	AV IQ		Berechnung	Gruppe 1	Gruppe 2			
2	1	Bloed	99,2		T-Test	Priming =	Priming =			
3	3	Bloed	95,8		Mittelwert	=bloed	intelligent			
4	5	Bloed	95,9		Streuung	92,34	103,64333			
5	7	Bloed	81,6		n	15,08394	15,212125			
6	9	Bloed	94,5		SE Mittel	30	30			
7	11	Bloed	90,5		siB	2,753938	2,7773413			
8	13	Bloed	104,4		SE Mitteldiff	15,148168				
9	15	Bloed	79,4		t_emp	3,9112402				
10	17	Bloed	87		t_krit 1s	2,8899615				
11	19	Bloed	117,3		p	1,6715528				
12	21	Bloed	110,3		Berechnung p über	0,0027058				
13	23	Bloed	105,8		p	=ttest()				
14	25	Bloed	49,6			0,0027058				
15	27	Bloed	98,5							
16	29	Bloed	97,7							
17	31	Bloed	83,5							
18	33	Bloed	84,9							
19	35	Bloed	97,1							
20	37	Bloed	71,8							
21	39	Bloed	102,6							

**Relevante Daten in Excel auswählen:**

Unsere Beispiel-Datei enthält alle möglichen Inhalte, die wir nicht in R importieren wollen: z. B. die Überschrift, die Aufgabenstellung, die Felderbeschriftungen für die Zwischenschritte, etc. Um die relevanten Daten zu extrahieren, markieren wir diese. Wichtig ist, dass wir wirklich nur das markieren, was wir zum Rechnen brauchen! Das sind allein die Messwerte, Information über Ausprägung der UV und evtl. Versuchspersonen-Nummern. Die Spaltenbeschriftung der relevanten Werte (in unserem Beispiel *VP-Nummer*, *UV Priming* und *AV IQ* können und sollen auch markiert werden.

**Relevante Daten in eine neue Excel-Datei kopieren:**

Wenn wir die Daten ausgewählt haben, kopieren wir sie (**STRG + C**), öffnen eine neue Excel-Datei (**STRG + N**) und fügen die Daten dort ein (**STRG + V**).



VP-Nummer	UV Priming	AV IQ
1 Bloed		99,2
3 Bloed		95,8
5 Bloed		95,9
7 Bloed		81,6
9 Bloed		94,5
11 Bloed		90,5
13 Bloed		104,4
15 Bloed		79,4
17 Bloed		87
19 Bloed		117,3
21 Bloed		110,3
23 Bloed		105,8
25 Bloed		49,6
27 Bloed		98,5
29 Bloed		97,7
31 Bloed		83,5
33 Bloed		84,9
35 Bloed		97,1
37 Bloed		71,8
39 Bloed		102,6
41 Bloed		83,2
43 Bloed		84,9
45 Bloed		105,9
47 Bloed		86,9
49 Bloed		121
51 Bloed		74,8
53 Bloed		102,6
55 Bloed		76,9
57 Bloed		109,6
59 Bloed		77
2 Intelligent		98,3
4 Intelligent		112,6
6 Intelligent		119,3
8 Intelligent		95,1
10 Intelligent		84,3
12 Intelligent		104,5

**Anpassen der Überschriften:**

Um den Zugriff auf die einzelnen Spalten in R zu vereinfachen, müssen wir darauf achten, dass die Überschriften der Spalten **keine Umlaute und keine Leerzeichen** enthalten. Leerzeichen können beispielsweise durch Punkte oder Unterstriche (\_) ersetzt werden. (Wenn die Überschriften Leerzeichen enthalten, wird R beim Import automatisch Punkte an die entsprechenden Stellen setzen. Um die Beschriftung einheitlich zu gestalten und einen fehlerfreien Import sicherzustellen, bietet es sich aber an, die Überschriften schon vor dem Import anzupassen).

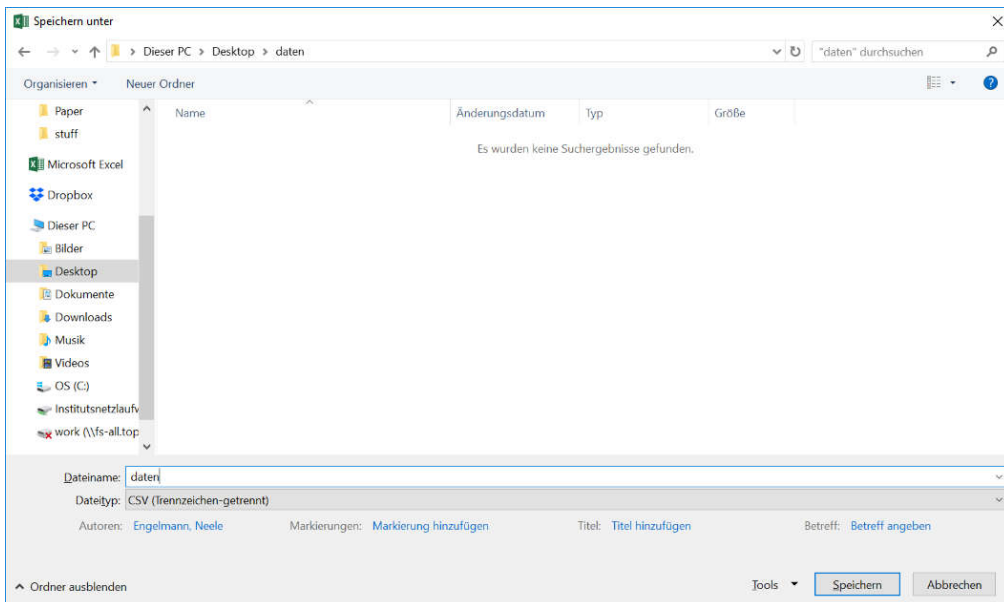
	A	B	C
1	VP-Nummer	UV_Priming	AV_IQ
2	1	Bloed	99
3	3	Bloed	96
4	5	Bloed	96

**Daten als CSV-Datei speichern:**

Die so extrahierten Daten speichern wir nun in einem Format, in dem R sie lesen kann. Dazu klicken wir in Excel unter **Datei** auf **Speichern unter**. Anschließend navigieren wir zu dem Ordner, den wir am Anfang erstellt haben.

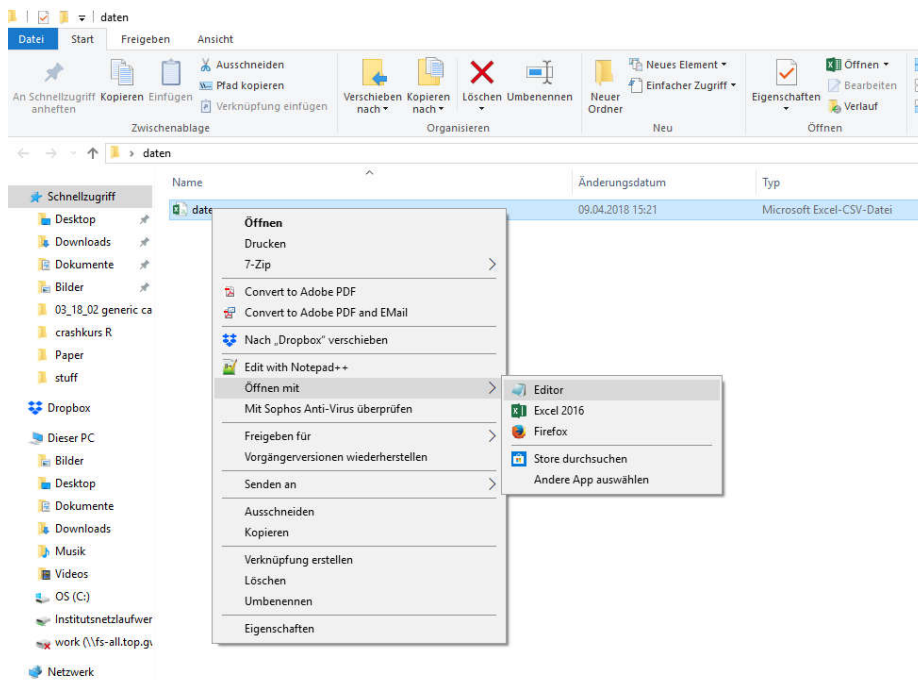
Wir wählen einen geeigneten Dateinamen (hier: "daten") und klicken dann bei **Dateityp:** auf **CSV (Trennzeichengetrennt) (\*.csv)** und klicken auf **Speichern**.

(Die daraufhin erscheinenden Warnmeldung von Excel können in diesem Fall ignoriert werden.)



**CSV-Datei prüfen:**

Um einen Eindruck davon zu bekommen, wie die erstellte CSV-Datei aussieht, öffnen wir sie mit einem beliebigen Texteditor (z.b. Editor oder Notepad). Dafür öffnen wir zuerst den Ordner, den wir am Anfang erstellt haben. Wir klicken mit der rechten Maustaste auf die CSV-Datei, wählen **Öffnen mit** und klicken auf **Editor** oder wir klicken nach dem ersten Rechtsklick direkt auf **Edit with Notepad**.



**CSV** steht für *Comma Separated Values*. Es ist ein Dateiformat für Daten in Tabellen. Es werden dafür bestimmte Zeichen benutzt, um dem Computer zu sagen, was eine Zeile und was eine Spalte ist: Spalten werden durch **Separatoren** getrennt und Zeilen normalerweise durch Zeilenumbrüche. Wenn wir Daten aus Excel in R importieren wollen, ist es wichtig zu wissen, welche Zeichen in der CSV-Datei als Separatoren verwendet wurden, denn es sind nicht immer dieselben. Außerdem werden manchmal Punkte (.) und manchmal Kommata (,) benutzt, um in Dezimalzahlen die Nachkommastellen zu markieren (1,23 vs. 1.23). Im englischen Sprachraum werden meist Punkte als **Dezimalzeichen**, im deutschen Sprachraum meist Kommata verwendet. Welche Zeichen in einer CSV-Datei als Separatoren und welche als Dezimalzeichen benutzt wurden, verrät uns ein Blick in die Datei mit Notepad:

```

C:\Users\engelmann6\Desktop\daten\daten.csv - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
daten.csv
1 VP-Nummer;UV Priming;AV IQ
2 1;Bloed;99,2
3 3;Bloed;95,8
4 5;Bloed;95,9
5 7;Bloed;81,6
6 9;Bloed;91,5
7 11;Bloed;90,5
8 13;Bloed;104,4
9 15;Bloed;79,4
10 17;Bloed;87
11 19;Bloed;117,3
12 21;Bloed;110,3
13 23;Bloed;105,8
14 25;Bloed;49,6
15 27;Bloed;98,5
16 29;Bloed;97,7
17 31;Bloed;83,5
18 33;Bloed;84,9
19 35;Bloed;97,1

```

Separator: Semikolon

Dezimalzeichen: Komma

Hier sieht man, dass in unserem Fall Semikola (;) als Separatoren verwendet werden und Kommata als Dezimalzeichen. Wir haben jetzt eine R-taugliche CSV-Datei mit unseren Daten und wissen, welche Separatoren und Dezimalzeichen verwendet werden.

## Wide format vs. Long format

Tabellen können unterschiedlich aufgebaut sein. Zur Veranschaulichung sehen wir uns den folgenden Screenshot an. Es handelt sich hier um die Messergebnisse von zehn Versuchspersonen in einem Messwiederholungsdesign. Die UV ist hier der Messzeitpunkt: jede Versuchsperson wurde einmal vor und einmal nach dem Urlaub gemessen. In der Tabelle sind die Vorher- und die Nachher-Werte in separaten Spalten eingetragen, d. h., die Daten einer Versuchsperson stehen in einer Zeile.

	A	B	C	D
1	<b>TN-Nr</b>	<b>IQ vor Urlaub</b>	<b>IQ nach Urlaub</b>	
2	1	97	99	
3	2	112	96	
4	3	118	96	
5	4	96	92	
6	5	84	96	
7	6	105	91	
8	7	93	103	
9	8	114	99	
10	9	117	86	
11	10	113	116	
12				
13				
14				
15				
16				
17				

"wide"-Format:  
Jede Zeile enthält die Daten von genau einer Versuchsperson;  
jede Spalte enthält die Daten einer erhobenen Variablen.

Im nächsten Screenshot handelt es sich um genau dieselben Daten, nur sind die unterschiedlichen Ausprägungen der UV (Messung vorher/Messung nachher) nicht mehr in zwei Spalten, sondern in einer zusammengefasst. Die Spalte mit der Überschrift **UV** enthält alle Ausprägungen der UV, die Spalte mit der Überschrift **AV: IQ** enthält alle Ausprägungen der AV. Der Unterschied zum vorigen Screenshot ist wichtig, denn dort enthielt lediglich die Spaltenüberschrift Information über die Ausprägung der UV; die Spalten selbst enthielten nur Ausprägungen der AV.

	A	B	C	D	E	F	G
1	<b>TN-Nr</b>	<b>UV</b>	<b>AV: IQ</b>				
2		1 vor Urlaub	97				
3		2 vor Urlaub	112				
4		3 vor Urlaub	118				
5		4 vor Urlaub	96				
6		5 vor Urlaub	84				
7		6 vor Urlaub	105				
8		7 vor Urlaub	93				
9		8 vor Urlaub	114				
10		9 vor Urlaub	117				
11		10 vor Urlaub	113				
12		1 nach Urlaub	99				
13		2 nach Urlaub	96				
14		3 nach Urlaub	96				
15		4 nach Urlaub	92				
16		5 nach Urlaub	96				
17		6 nach Urlaub	91				
18		7 nach Urlaub	103				
19		8 nach Urlaub	99				
20		9 nach Urlaub	86				
21		10 nach Urlaub	116				
22							

"long"-Format:  
eine Spalte für alle Ausprägungen der UV, eine Spalte für alle Werte der AV

Der kritische Unterschied ist, dass beim Wide-Format alle Daten einer Person in einer Zeile stehen; beim Long-Format ist dies nicht der Fall, hier steht in einer Zeile der Wert einer einzelnen Messung von einer einzelnen Variable (wobei Werte einer Person auf mehrere Zeilen aufgeteilt werden, sofern es mehrere Messzeitpunkte oder mehrere AVs gibt).

Das Format einer Tabelle lässt sich in R ändern (Siehe Einführung 2). Unser Beispiel aus Abschnitt 1 (Priming von Intelligenz) benötigt vorerst keine Änderung (da es genau eine UV und nur eine Messung pro Person beinhaltet, unterscheiden sich long und wide format hier nicht. Eine Zeile entspricht hier einer Versuchsperson, aber auch einer Messung).

Der Grund, warum uns das überhaupt interessiert, ist, dass R seine Datenhäppchen manchmal im einen und manchmal im anderen Format möchte (wann wir welches Format benötigen, wird in den jeweiligen Anwendungsbeispielen deutlich).

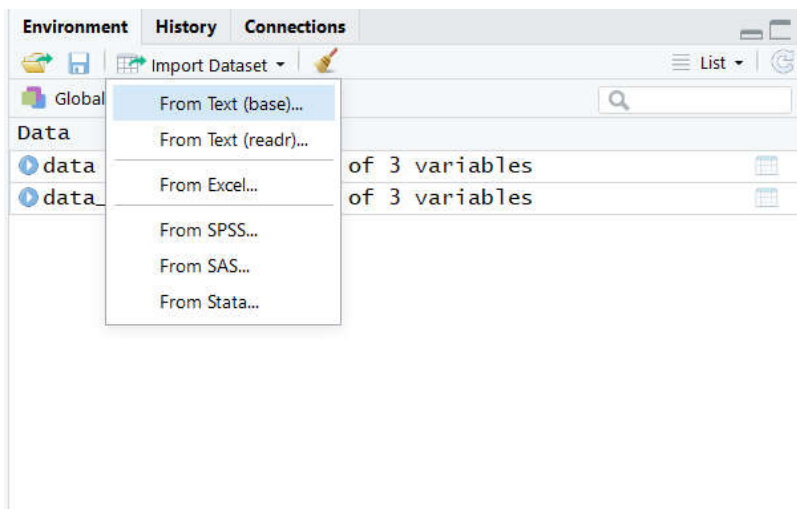
### 3) Daten in R importieren

#### Ausgangspunkt:

Die Daten wurden aufbereitet und in einer CSV-Datei gespeichert. Wir haben die CSV-Datei mit dem Editor geöffnet, um herauszufinden, welche Zeichen als Separatoren und welche als Dezimalzeichen verwendet wurden.

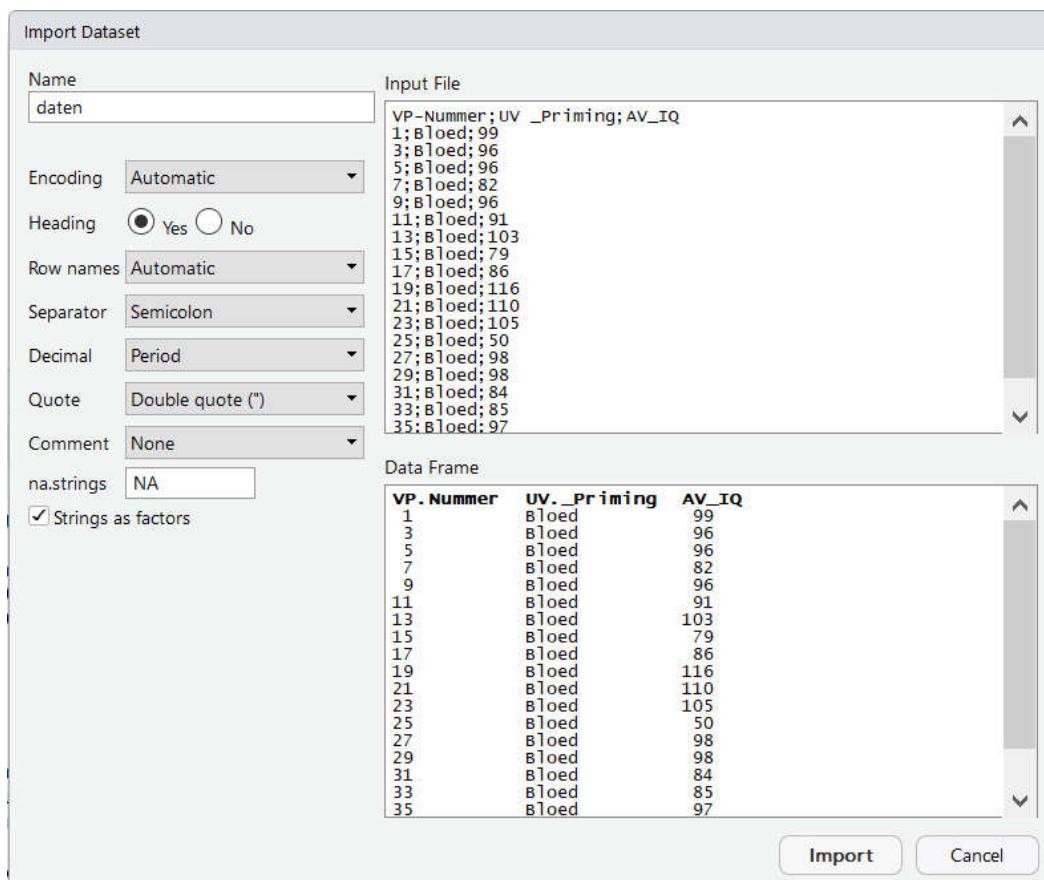
Dazu klicken wir in RStudio rechts unter *Environment* auf *Import Dataset*, klicken auf die Option *From Text (base)* und wählen den von uns vorbereiteten Datensatz aus.

(Im Drop-Down-Menü bietet das Programm auch an, Dateien direkt aus Excel zu importieren. Das funktioniert, macht aber teilweise bei späteren Arbeitsschritten Probleme. Außerdem basieren viele online verfügbare Rechenbeispiele, beispielsweise zum Lehrbuch von Andy Field, auf CSV-Dateien. Daher ist es zu diesem Zeitpunkt zu empfehlen, die Dateien als CSV zu importieren.)



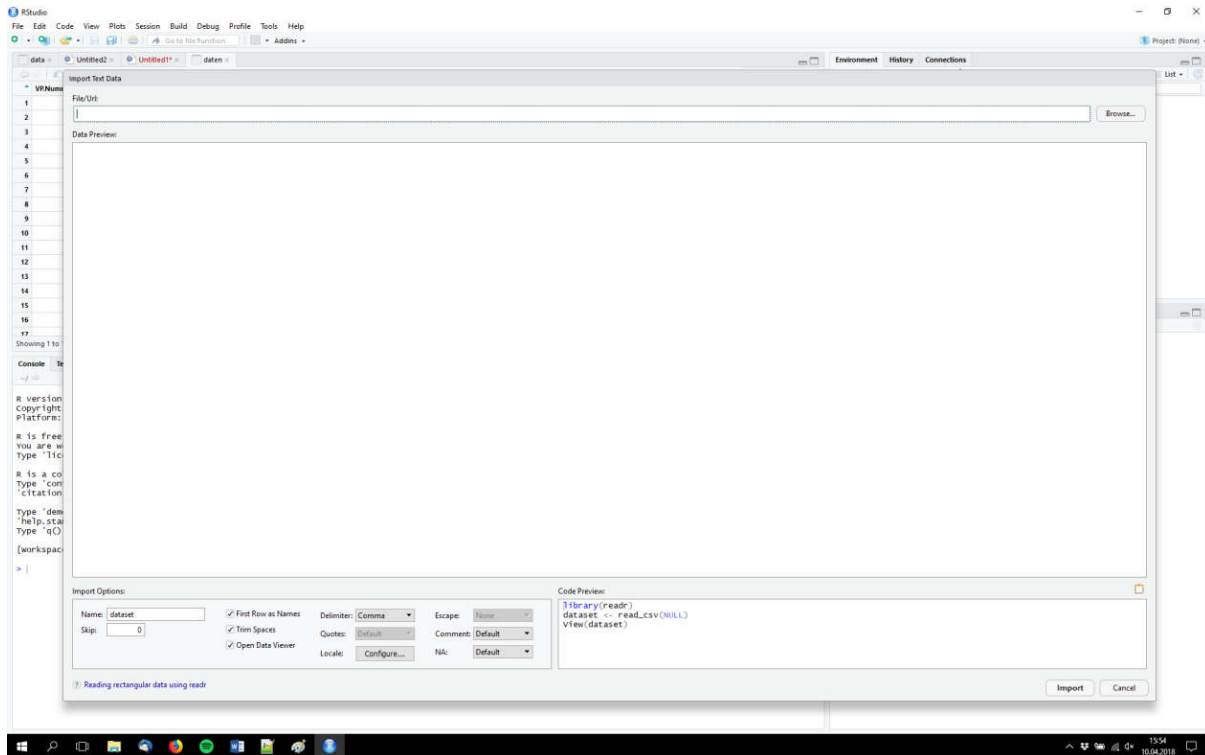


RStudio fragt uns nun ein paar Sachen. Als erstes sollten wir angeben, unter welchem Namen die Daten später in R/RStudio/RCommander ansprechbar sein sollen. Wichtig ist vor allem, dass wir ihn uns merken und später wissen, welche Daten sich hinter diesem Namen verbergen. In diesem Beispiel belasse ich den Namen bei *daten*. RStudio will jetzt außerdem wissen, wie die CSV-Datei strukturiert ist; vor allem, welche Separatoren und welche Dezimalzeichen verwendet werden. Das wissen wir jetzt, weil wir die Datei zuvor mit dem Editor angeschaut haben: in unserem Beispiel haben wir als Separatoren Semikola (Dezimalstellen gibt es in unserem Datensatz nicht, daher ist es an dieser Stelle egal, welche Option wir wählen. Bei Daten mit Dezimalstellen – der typischere Fall - ist diese Information jedoch essentiell für den korrekten Import). Die anderen Einstellungen können (in unserem Beispiel) so bleiben, wie sie voreingestellt sind.

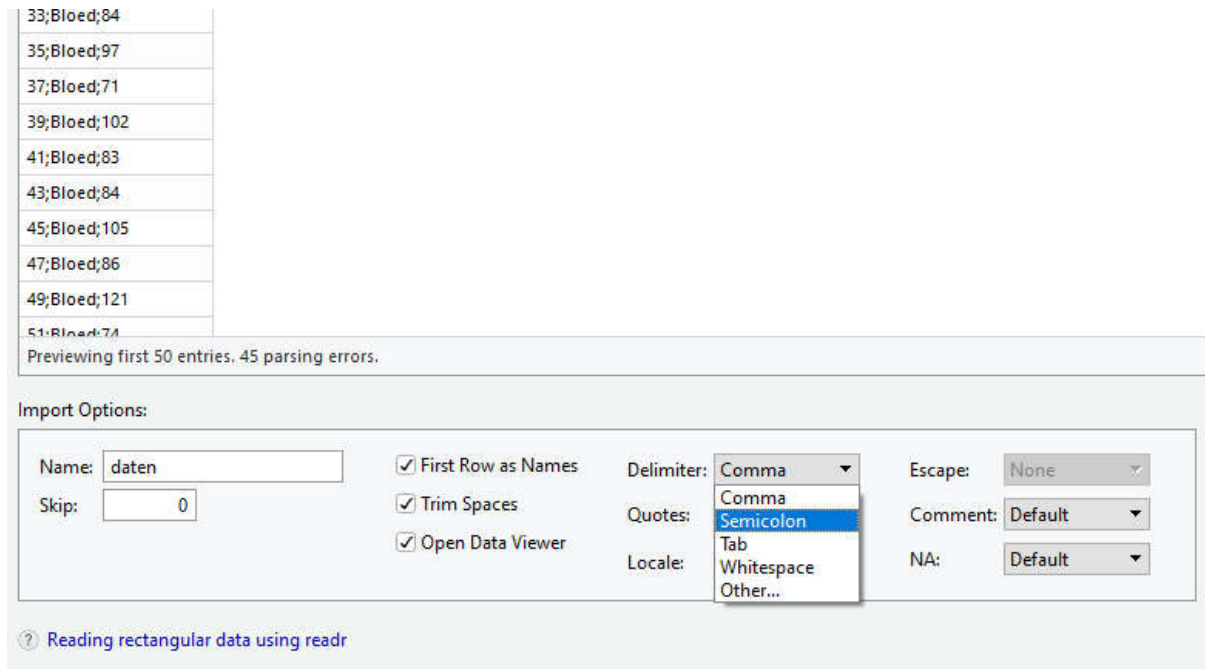


Nachdem wir auf **Import** geklickt haben, sind die Daten in RStudio. Das sehen wir daran, dass die Variable *Daten* im Fenster von RStudio rechts unter **Environment** aufgelistet wird. Hier werden alle "geladenen" Daten angezeigt, die momentan in unserer R-Sitzung herumfliegen.

Alternativ (z.B. wenn es mit der ersten Funktion Probleme gibt), können CSV-Dateien auch über die Option „from text (readr)“ importiert werden. Wird diese Funktion gewählt, öffnet sich folgendes Fenster:

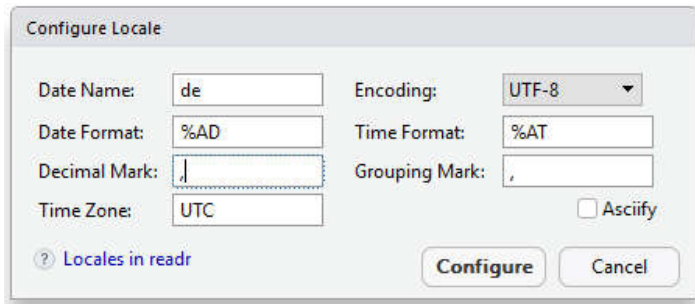


Nach einem Klick auf Browse (oben rechts) können wir unsere aufbereitete CSV-Datei auswählen und erhalten eine Vorschau. Im unteren Bereich des Fensters können wir unter „Delimiter“ das Semikolon als Separator auswählen:



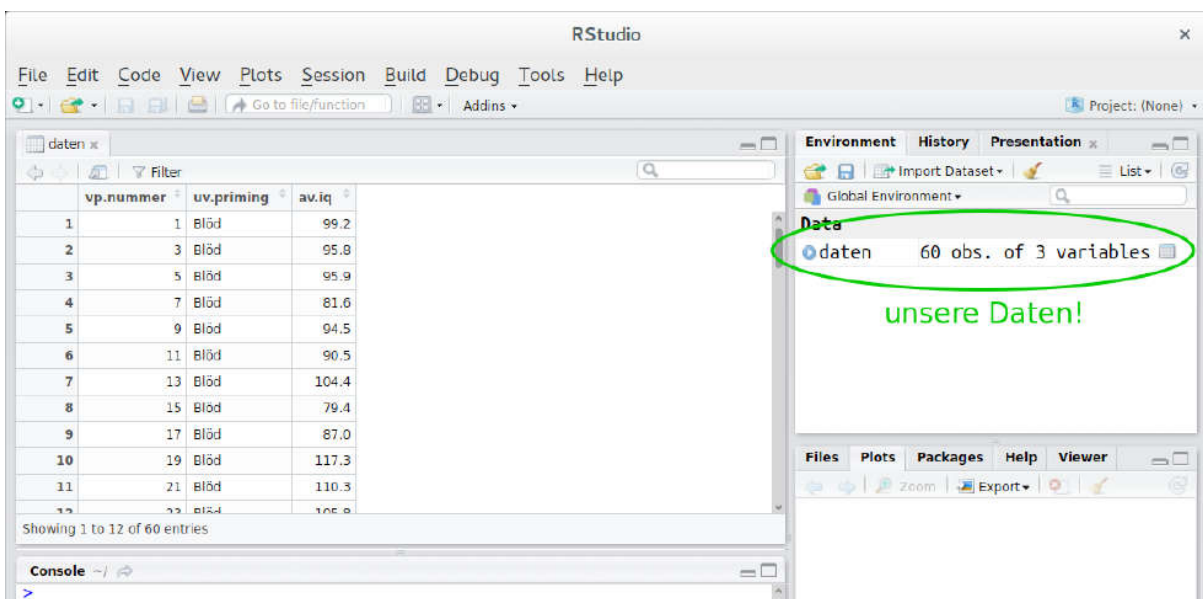
Nach einem Klick auf *Configure* lässt sich außerdem das Komma als Dezimalzeichen einstellen. Dazu einfach unter *Date Name* „de“ eintippen (um die deutsche Formatierung zu kennzeichnen) und unter *Decimal Mark* ein Komma eintippen:



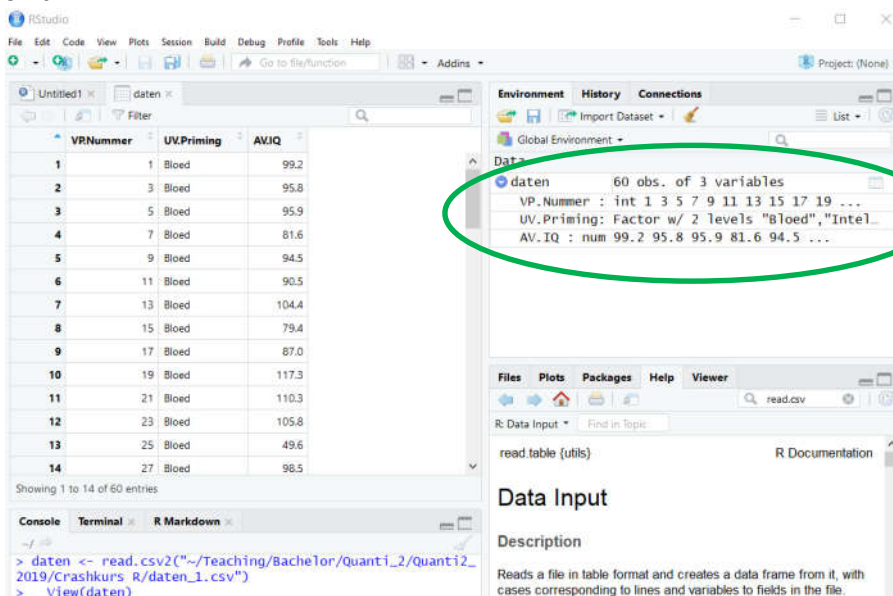


In der Vorschau sollte unsere Tabelle jetzt korrekt angezeigt werden und wir können sie importieren.

Wenn die Daten importiert sind, dann sind diese rechts oben im Environment zu sehen.



Ein Klick auf den blauen Kreis, zeigt uns welche Variablen in unserem Data Frame vorhanden sind.



R hat richtig erkannt, dass die UV ein Faktor mit 2 Ausprägungen ist und die AV eine numerische Variable.

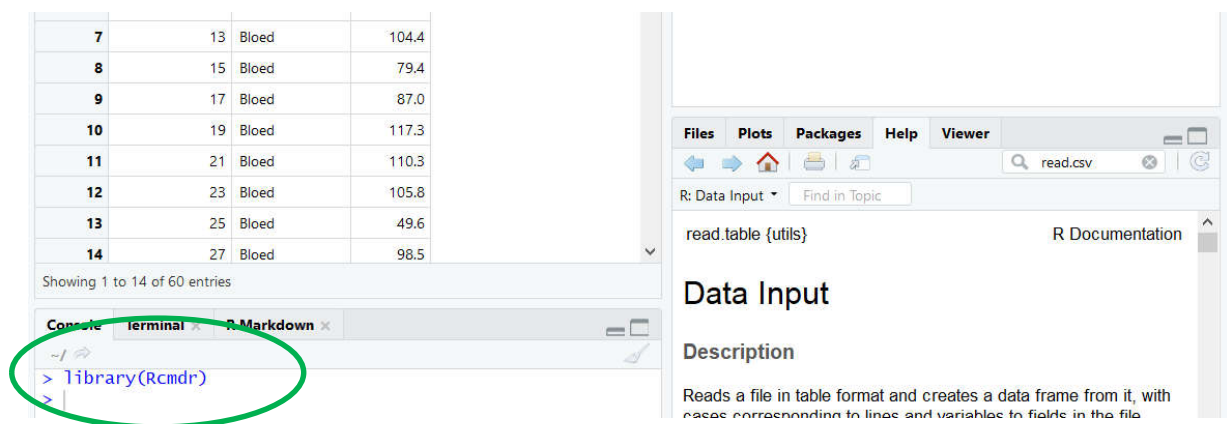
Falsch ist, dass die VP Nummer als Integer (Kardinalzahl) erkannt wurde. Die VP-Nummer dient ja nur zur Unterscheidung und ist daher ein numerisch kodierter Faktor. Das ist hier aber irrelevant.

## 4) Arbeiten mit R

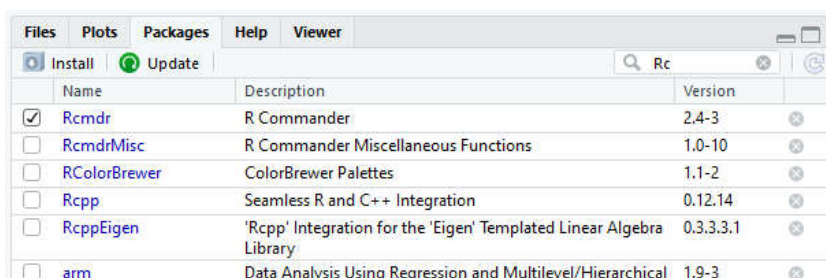
Um mit R zu arbeiten gibt es drei Möglichkeiten, die man beliebig miteinander mischen kann.

### 4.1) R-Commander

Der R-Commander ist eine graphische Benutzeroberfläche bei der man mittels Auswahlménüs angeben kann, was R für einen machen soll (z.B. Graphiken, statistische Analysen). Der R-Commander generiert ein Skript in der Sprache R, die R dann ausführen kann. Die Optionen des Commanders sind begrenzt, decken aber das meiste ab, was wir praktisch brauchen. Den Commander kann man starten, indem wir im Befehlsfenster (Console) unten in RStudio `library(Rcmdr)` eingeben und auf ENTER drücken. WICHTIG: R achtet auf Groß- und Kleinschreibung



Alternativ können wir im rechten unteren Fenster des Workspaces unter *Packages* nach „Rcmdr“ suchen und den Haken davor setzen:

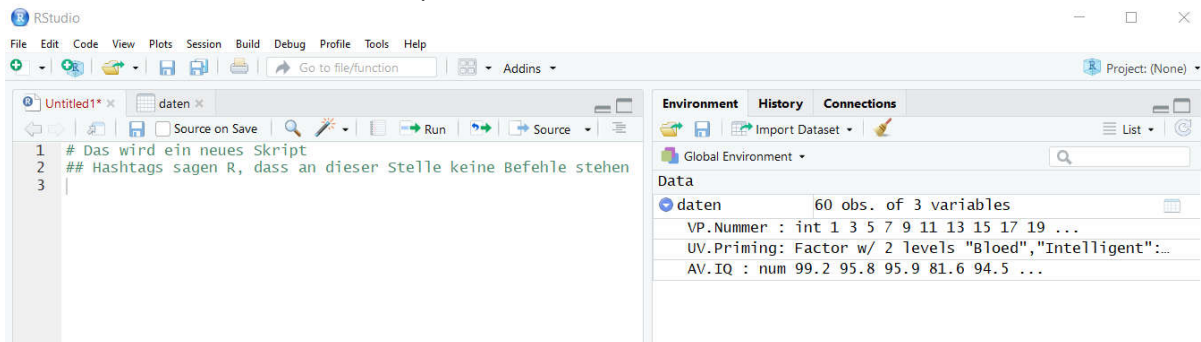


Wenn der RCommander bereits geöffnet war (und wieder geschlossen wurde) und RStudio nicht neugestartet wurde, öffnet der folgende Befehl ihn erneut:

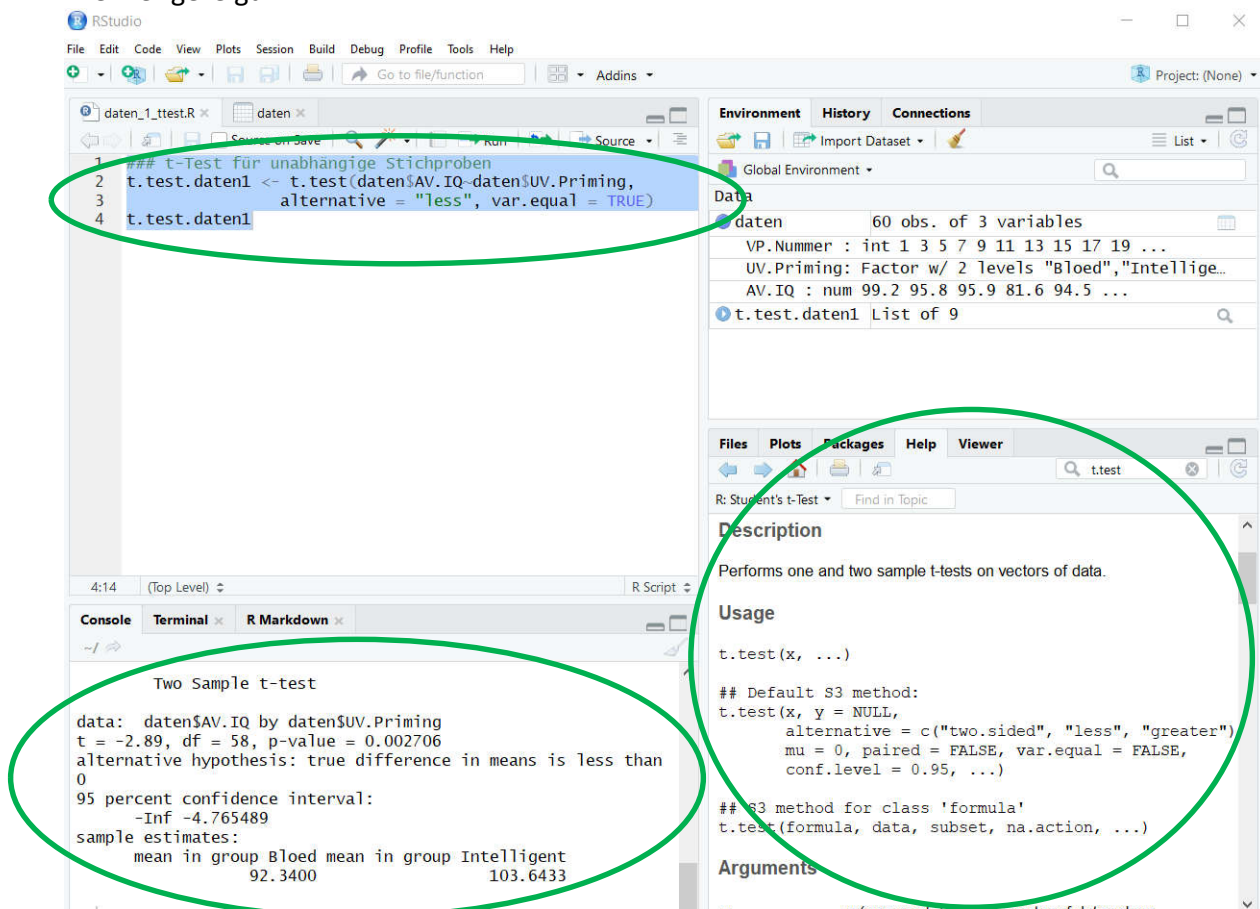
**Commander ( )**

## 4.2) Skript schreiben

Der R-Commander generiert ein Skript in R, welches dann vom Rechner für uns abgearbeitet wird. Wir können natürlich auch direkt ein Skript in R schreiben. R hat eine Befehlssprache. Kennen wir die Befehle, dann macht R alles was wir wollen. Im Internet findet man durch Googlen sehr schnell Lösungen für alle Fragen. Außerdem hilft uns R-Studio beim Schreiben eines Skripts. Im Laufe des Semesters werden wir immer mehr Befehle in R aus verschiedenen Paketen kennenlernen, die uns die Arbeit sehr erleichtern. Um ein Skript zuzuschreiben, machen wir uns in R-Studio eine neue Skriptdatei auf.



Dann können wir sehr leicht zum Beispiel einen t-Test machen. Die Hilfe rechts unten verrät uns, wie die Syntax des zugehörigen Befehls ist. Das Ergebnis der Analyse steht dann in der Konsole wie hier gezeigt.



### 4.3) R Markdown

Das Arbeiten mit einem Skript ist aus vier Gründen etwas unschön: (1) Kommentare benötigen Hashtags weil R sonst verwirrt ist, (2) die Ergebnisse stehen in der Konsole, (3) die Graphiken stehen rechts unten in einem separaten Fenster, und (4) der Export aus R ist etwas unpraktisch (siehe Abschnitt zu Ergebnisse speichern). Um diese Probleme zu lösen, gibt es eine tolle Möglichkeit: R Markdown. R Markdown erlaubt es, in eine Datei Texte, R-Befehle und R-Ergebnisse zu schreiben und diese sehr einfach nach HTML, PDF und Word zu exportieren. Die untenstehende Abbildung zeigt ein R-Markdown-File in R-Studio.

The screenshot shows the RStudio interface with an R Markdown file open. The code in the editor is as follows:

```

1 ---
2 title: "Uebung1_ttest"
3 author: "YH"
4 date: "13 März 2019"
5 output: word_document
6 ---
7 ## Pakete laden
8 In verschiedenen Paketen sind verschiedene Befehle in R hinterlegt.
9 Diese müssen vor der Verwendung aktiviert werden.
10 {r}
11 library(readr)
12
13
14 ## Daten einlesen
15 Das Einlesen der Daten kann man auch vereinfachen und
16 automatisieren. Zuerst wird definiert, wo die Daten zu finden sind
17 mit set working directory setwd() und dann werden die Daten mit
18 read.csv() geöffnet.
19 {r}
20 setwd("~/Teaching/Bachelor/Quanti_2/Quanti2_2019/Crashkurs R")
21 daten <- read.csv("daten_1.csv",header = TRUE, sep=";", dec=",")
22 View(daten)
23
24 ## t-Test
25 Für einen t-Test wird nur eine einfache Befehlszeile benötigt. Der
26 Befehl lautet t.test(). Weitere Informationen zum t.test Befehl
27 sind unter help t.test zu finden.
28 {r}
29 t.test.daten1 <- t.test(daten$AV.IQ ~ daten$UV.Priming, data =
30 daten, alternative = "less", var.equal = TRUE)
31 t.test.daten1

```

The right-hand pane shows the Environment window with the following data objects:

- Global Environment**
  - daten**: 60 obs. of 3 variables
    - VP.Nummer : int 1 3 5 7 9 11 13 15 17...
    - UV.Priming: Factor w/ 2 levels "Bloed...
    - AV.IQ : num 99.2 95.8 95.9 81.6 94.5 ...
  - t.test.da...**: List of 9
    - statistic : Named num -2.89
    - ..- attr(\*, "names")= chr "t"
    - parameter : Named num 58
    - ..- attr(\*, "names")= chr "df"

The Viewer window shows the rendered HTML output for the t-test function:

## Student's t-Test

**Description**

Performs one and two sample t-tests on vectors of data.

**Usage**

```
t.test(x, ...)
```

## Default S3 method:

Über knitr können wir das Ganze beliebig exportieren

The screenshot shows the RStudio interface with the Knit menu open. The options are:

- Knit to HTML
- Knit to PDF
- Knit to Word
- Knit with Parameters...
- Knit Directory
- Clear Knitr Cache...



In Html sieht das Ergebnis des Exports dann so aus:

The screenshot shows a web browser window with the URL `~/Teaching/Bachelor/Quanti_2/Quanti2_2019/Crashkurs R/daten_1_ttest.html`. The page content is as follows:

## Uebung1\_ttest

YH  
13 März 2019

### Pakete laden

In verschiedenen Paketen sind verschiedene Befehle in R hinterlegt. Diese müssen vor der Verwendung aktiviert werden.

```
library(readr)
```

### Daten einlesen

Das Einlesen der Daten kann man auch vereinfachen und automatisieren. Zuerst wird definiert, wo die Daten zu finden sind mit `set working directory setwd()` und dann werden die Daten mit `read.csv()` geöffnet.

```
setwd("~/Teaching/Bachelor/Quanti_2/Quanti2_2019/Crashkurs R")
daten <- read.csv("daten_1.csv", header = TRUE, sep=";", dec=".")
View(daten)
```

### t-Test

Für einen t-Test wird nur eine einfache Befehlszeile benötigt. Der Befehl lautet `t.test()`. Weitere Informationen zum `t.test` Befehl sind unter `help t.test` zu finden.

```
t.test.daten1 <- t.test(daten$AV.IQ ~ daten$UV.Priming, data = daten, alternative = "less", var.equal = TRUE)
t.test.daten1
```

```
##
## Two Sample t-test
##
## data: daten$AV.IQ by daten$UV.Priming
## t = -2.89, df = 58, p-value = 0.002706
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
## -Inf -4.765489
## sample estimates:
## mean in group Bloed mean in group Intelligent
##          92.3400          103.6433
```

Und in Word so:

## Uebung1\_ttest

13 März 2019

### Pakete laden

In verschiedenen Paketen sind verschiedene Befehle in R hinterlegt. Diese müssen vor der Verwendung aktiviert werden.

```
library(readr)
```

### Daten einlesen

Das Einlesen der Daten kann man auch vereinfachen und automatisieren. Zuerst wird definiert, wo die Daten zu finden sind mit `set working directory setwd()` und dann werden die Daten mit `read.csv()` geöffnet.

```
setwd("~/Teaching/Bachelor/Quanti_2/Quanti2_2019/Crashkurs R")
daten <- read.csv("daten_1.csv", header = TRUE, sep=";", dec=".")
View(daten)
```

### t-Test

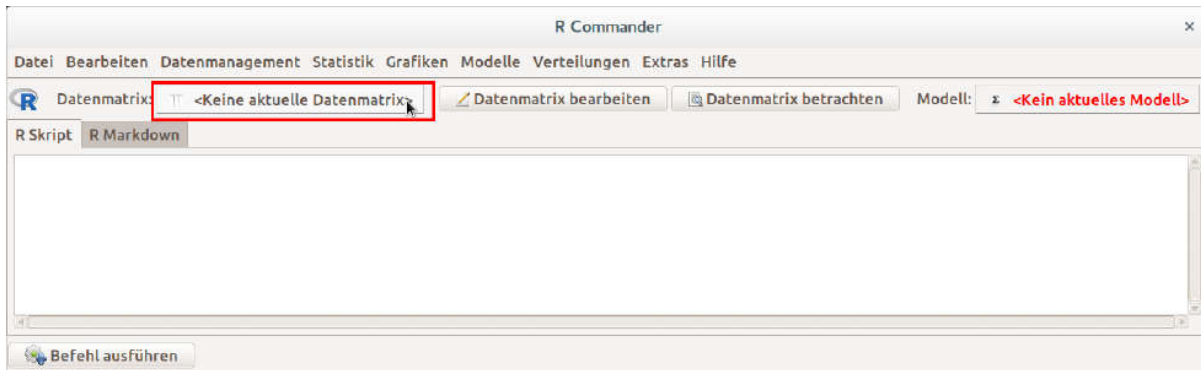
Für einen t-Test wird nur eine einfache Befehlszeile benötigt. Der Befehl lautet `t.test()`. Weitere Informationen zum `t.test` Befehl sind unter `help t.test` zu finden.

```
t.test.daten1 <- t.test(daten$AV.IQ ~ daten$UV.Priming, data = daten,
alternative = "less", var.equal = TRUE)
t.test.daten1
```

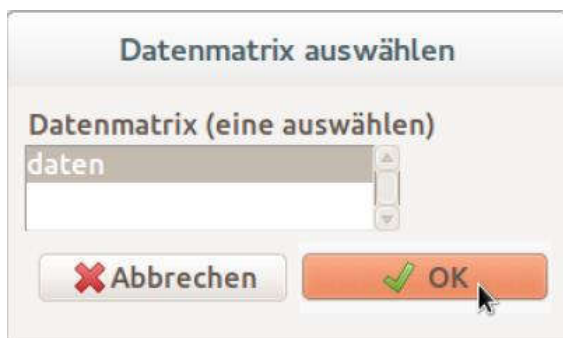
```
##
## Two Sample t-test
##
## data: daten$AV.IQ by daten$UV.Priming
## t = -2.89, df = 58, p-value = 0.002706
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
## -Inf -4.765489
## sample estimates:
## mean in group Bloed mean in group Intelligent
##          92.3400          103.6433
```

## 5) Deskriptive Statistik

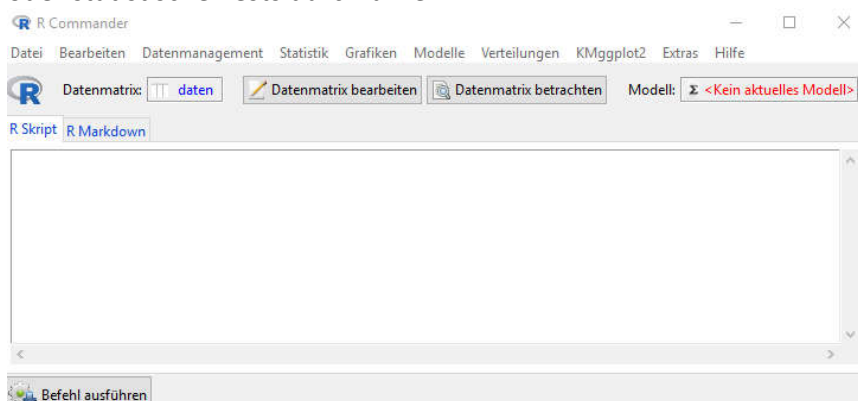
Nach dem Öffnen des R-Commanders über die Eingabe von `library(Rcmdr)` in der Konsole, müssen wir dem R-Commander mitteilen, mit welche Daten wir arbeiten wollen. Dazu klicken wir auf das Feld rechts neben *Datenmatrix*:



Im darauffolgenden Fenster wählen wir die Daten, die wir soeben importiert haben (hier erscheint der Name, den wir ihnen in RStudio gegeben haben) und klicken auf **OK**.

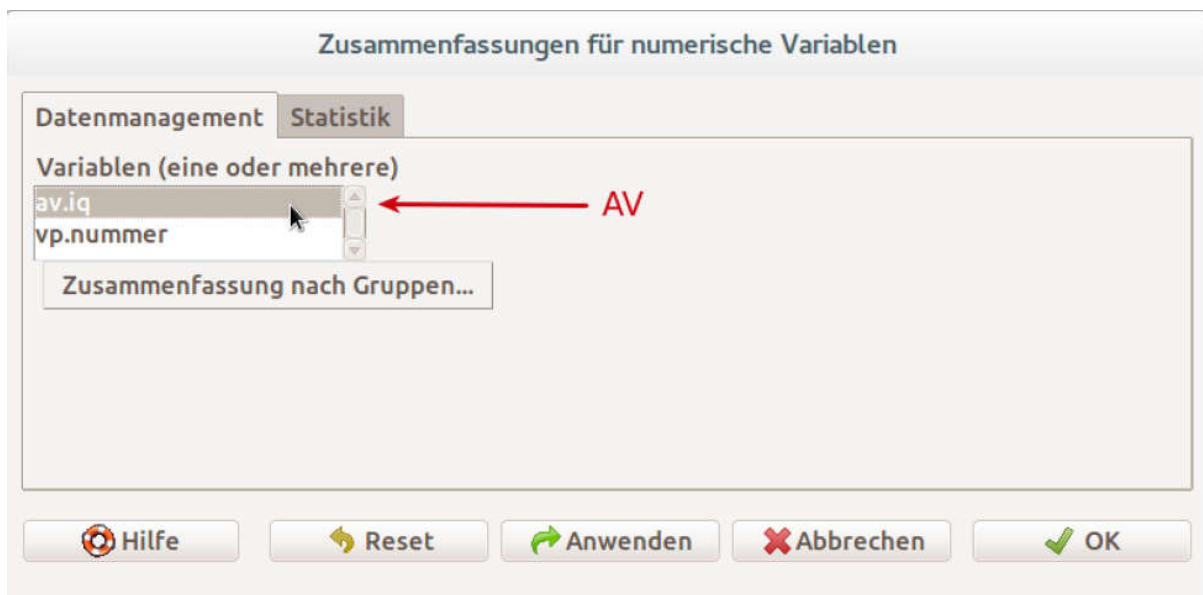


Die Daten sind jetzt in R-Commander verfügbar und wir können damit z. B. Grafiken erstellen oder statistische Tests durchführen.





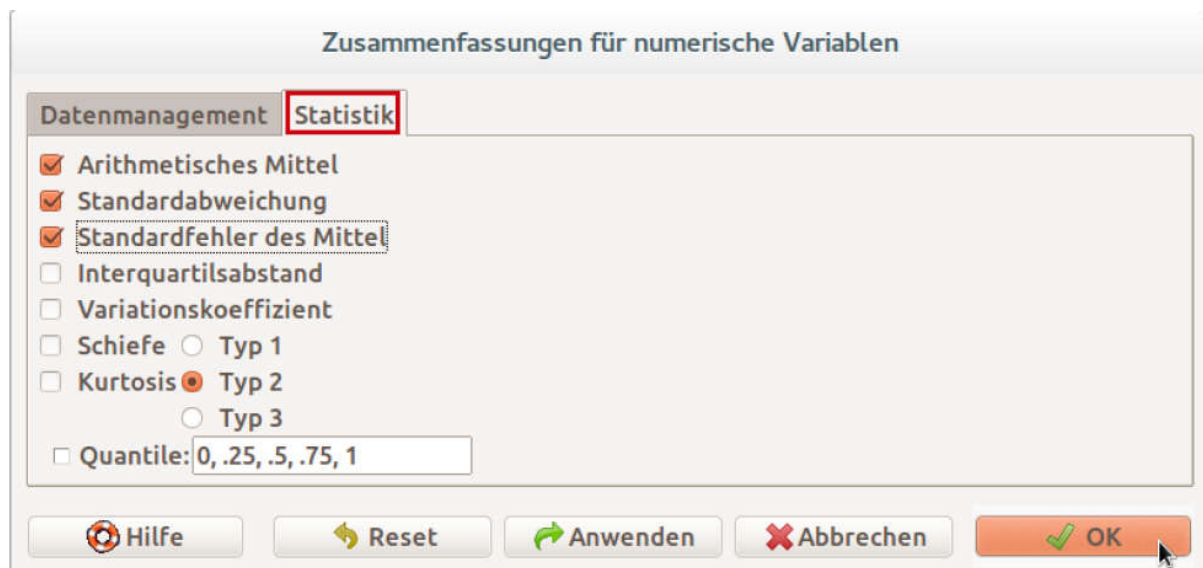
Die Optionen zur Berechnung deskriptiver Statistiken sind im Commander unter Statistik / Deskriptive Statistik zu finden.



Wir haben bei Gruppendesigns die Möglichkeit, die deskriptiven Statistiken für jede Gruppe einzeln anzuzeigen. Dazu klicken wir auf *Zusammenfassung nach Gruppen* und wählen die UV aus.



Hier wählen wir aus, welche deskriptiven Statistiken wir uns anzeigen lassen wollen.



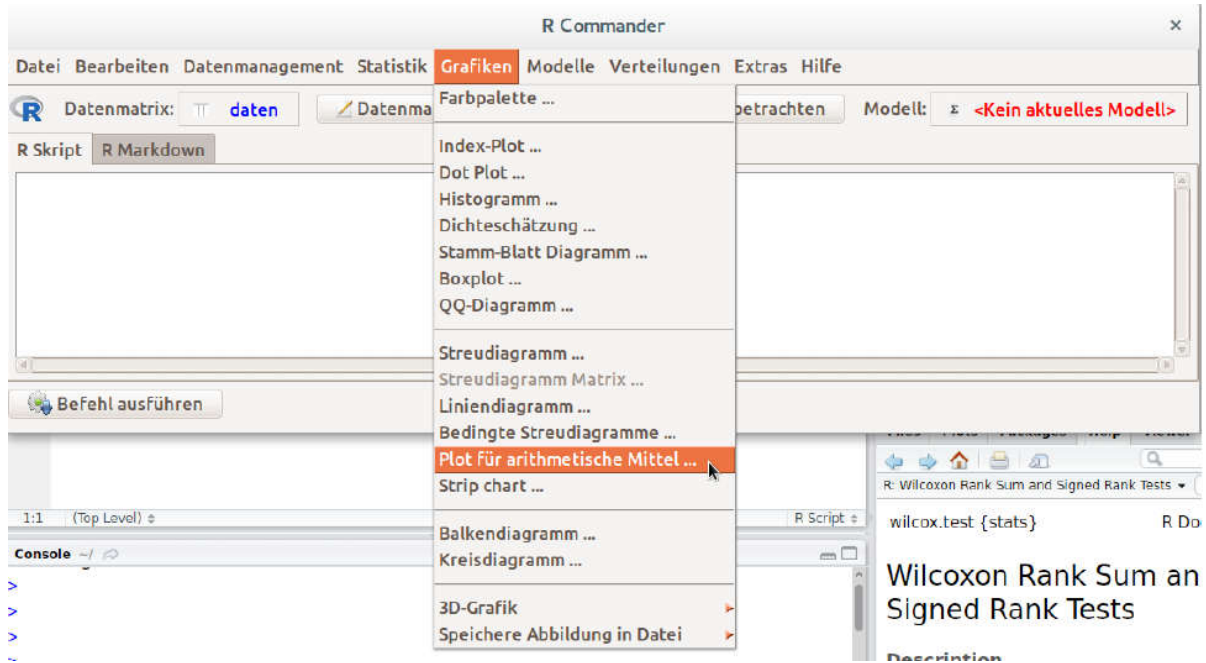
Die deskriptiven Statistiken werden in der Console von RStudio ausgegeben:

```
Rcmdr> numSummary(daten[, "AV.IQ", drop=FALSE], groups=daten$uv.Priming,
Rcmdr+   statistics=c("mean", "sd", "IQR", "quantiles"), quantiles=c(0,.25,.5,.75,1))
      mean      sd      IQR      0%      25%      50%      75%      100% AV.IQ:n
Bloed    92.3400 15.08394 19.325 49.6 83.275 95.15 102.60 121.0      30
Intelligent 103.6433 15.21213 18.925 71.2 94.725 103.30 113.65 144.3      30
> |
```

## 6) Mittelwert-Graphik

**Ausgangspunkt:** Die Daten liegen aufbereitet und im *Wide*-Format vor, die CSV-Datei wurde in RStudio geladen und die entsprechende Datenmatrix wurde in RCommander ausgewählt.

Wir müssen die Mittelwerte nicht von Hand ausrechnen lassen, um sie als Grafik anzuzeigen. Das macht R automatisch.



Hier wählen wir UV (*Gruppierungsvariablen*) und AV:



Unter dem Reiter *Optionen* haben wir die Möglichkeit, Fehlerbalken anzeigen zu lassen. Wir können Standardfehler (des Mittelwerts), Standardabweichungen und Konfidenzintervalle als Fehlerbalken anzeigen lassen. In der rechten Hälfte können wir eigene Achsenbeschriftungen festlegen.

Grafik für arithmetische Mittelwerte

Datenmanagement Optionen

Fehlerbalken

- Standardfehler
- Standardabweichungen
- Konfidenzintervalle**
- Keine Fehlerbalken

Konfidenzintervallniveau: 0.95

Grafikbeschriftung

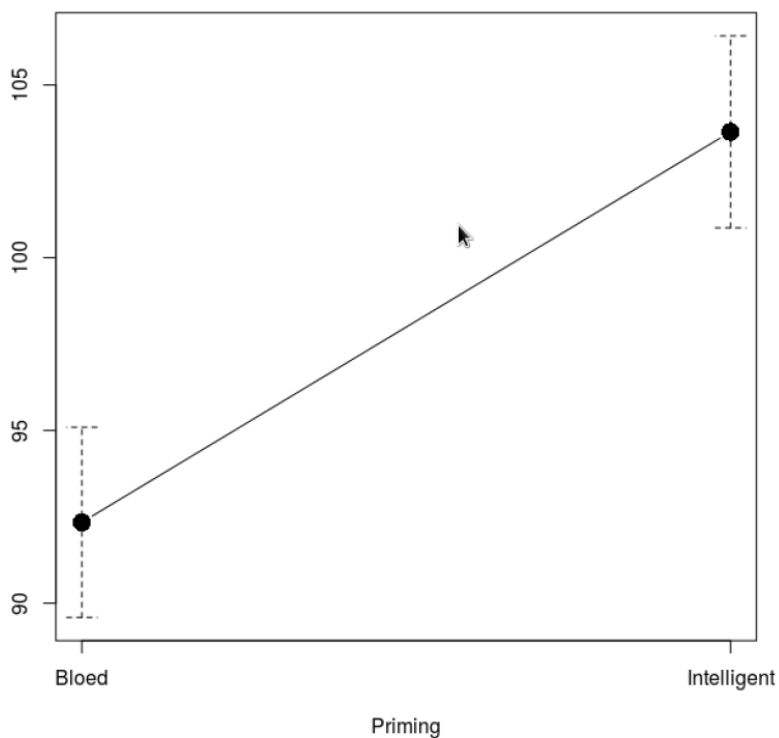
Label der X-Achse Priming

Label der Y-Achse IQ

Titel der Grafik Fluss von Priming auf den IQ

Hilfe Reset Anwenden Abbrechen OK

Einfluss von Priming auf den IQ



R-Commander bietet keine Möglichkeit, die Achsenskalierung zu verändern. Wir können das von Hand machen, indem wir den Befehl, mit dem R die Grafik erstellt, verändern. Der Befehl befindet sich im R Skript-Feld des R-Commanders und beginnt mit "with(":

R Commander

Datei Bearbeiten Datenmanagement Statistik Grafiken Modelle Verteilungen Extras Hilfe

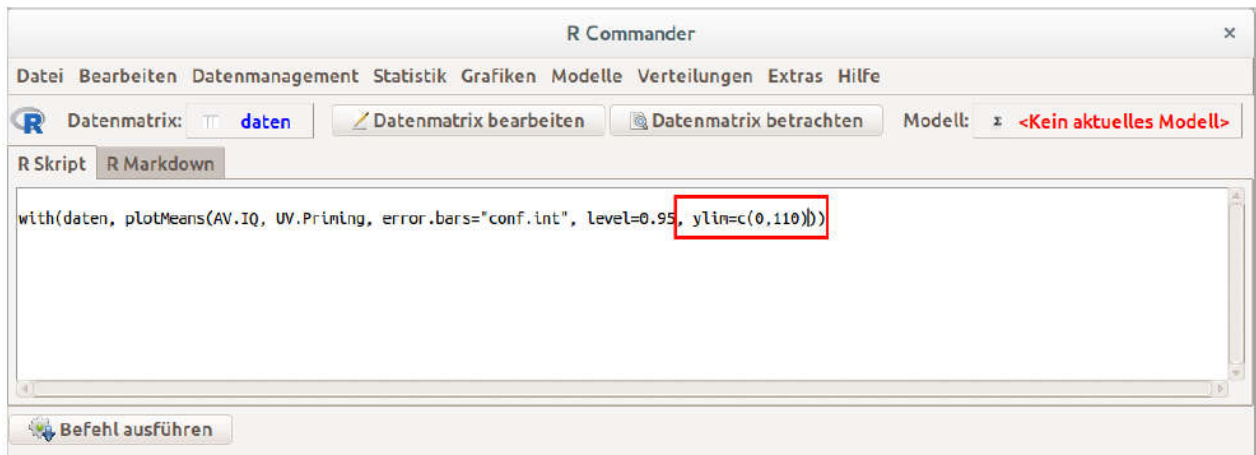
Datenmatrix: **daten** Datenmatrix bearbeiten Datenmatrix betrachten Modell: **<Kein aktuelles Modell>**

R Skript R Markdown

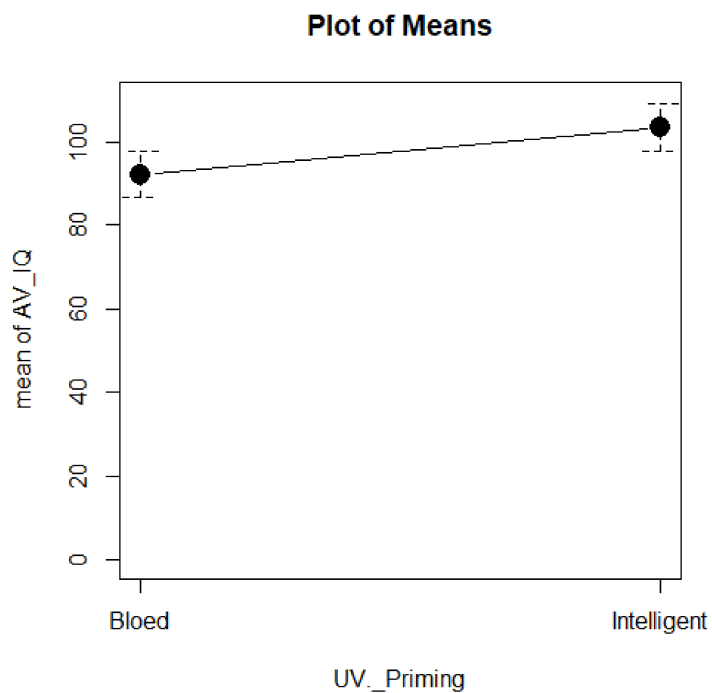
```
with(daten, plotMeans(AV.IQ, UV.Priming, error.bars="conf.int", level=0.95))
```

Befehl ausführen

Wir können die Achsenskalierung ändern, indem wir dem darin enthaltenen "plotMeans"Befehl Ein weiteres Argument übergeben. Das tun wir, indem wir vor ")))" folgendes dazuschreiben: "ylim=c(0,110)". Das bedeutet, dass die yAchse von 0 bis 110 gehen soll.



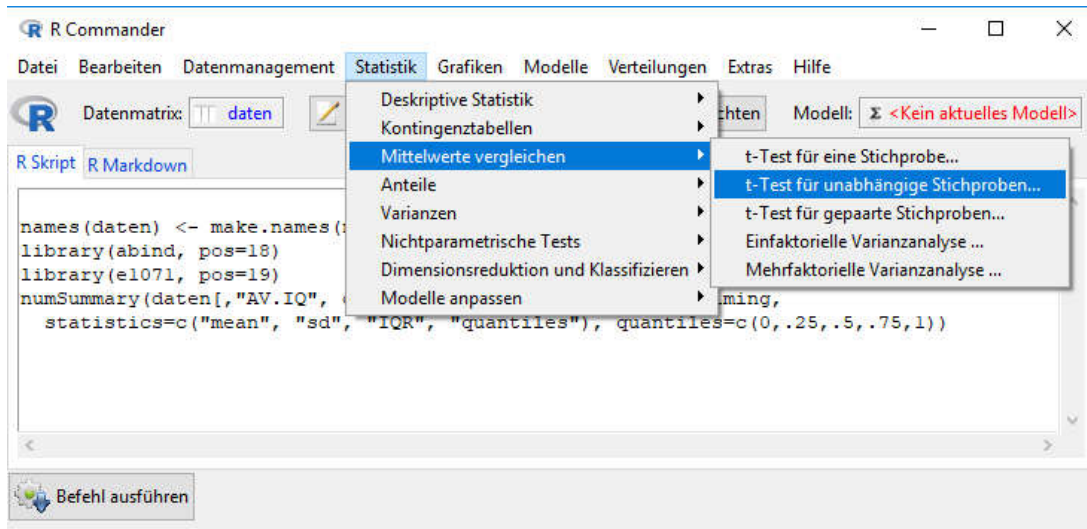
Nach einem Klick auf Befehl ausführen erscheint eine Graphik mit der gewünschten Achsenskalierung:



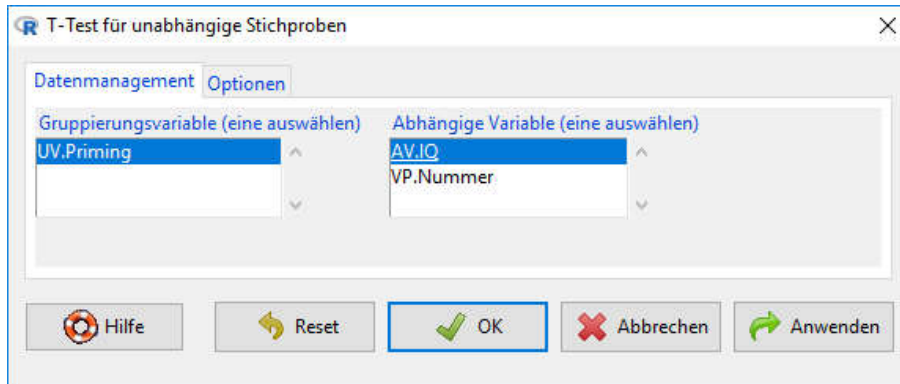
## 7) t-Test

**Ausgangspunkt:** Wir haben eine CSV-Datei in RStudio und RCommander importiert und wollen einen Mittelwertsunterschied auf Signifikanz prüfen (Beispiel: Priming von Intelligenz).

Wir wählen unter *Statistik* und *Mittelwerte vergleichen* den passenden Test aus (in diesem Fall: t-Test für unabhängige Stichproben):

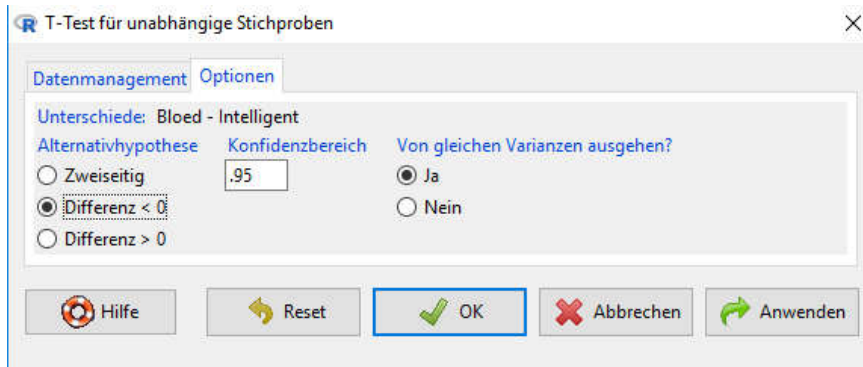


Im nächsten Schritt wählen wir die korrekte UV und AV aus:



Unter dem zweiten Reiter, „Optionen“, machen wir ergänzende Angaben zur Richtung des Tests und zum Alpha-Niveau. Bei einer ungerichteten Hypothese wählen wir „zweiseitig“, bei einer gerichteten Hypothese müssen wir zusätzlich angeben, in welche Richtung der Mittelwertsunterschied erwartet wird. Dabei ist zu beachten, dass R stets den alphabetisch früheren Faktor als ersten und den alphabetisch späteren als zweiten Mittelwert verwendet. In unserem Fall müssen wir also entscheiden, ob die Differenz zwischen den Gruppen „Bloed“ und „Intelligent“ größer oder kleiner als 0 sein wird (da unsere inhaltliche Hypothese eine Wirkung des Primings vorhersagt, erwarten wir, dass der Mittelwert für Intelligent höher ist und die Differenz somit negativ werden sollte). Alpha geben wir indirekt über „Konfidenzbereich“ an. Der Konfidenzbereich ist  $1 - \text{Alpha}$ . Bei „von gleichen Varianzen ausgehen?“ wählen wir „Ja“.





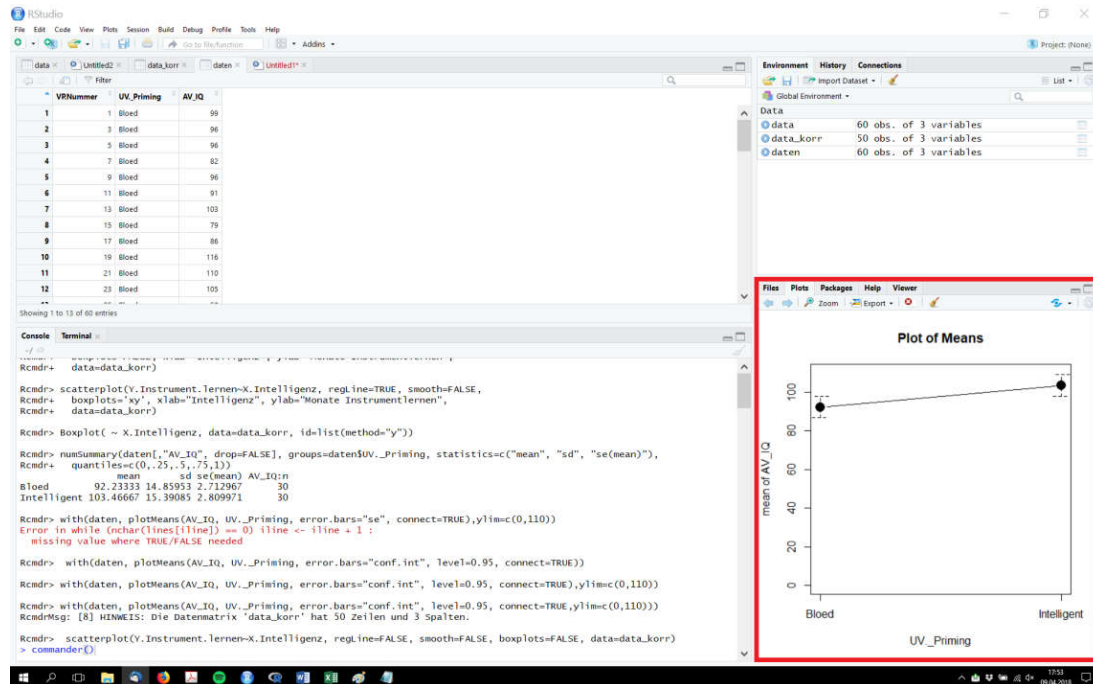
Das Ergebnis des Signifikanztests wird in der Console ausgegeben: Empirischer t-Wert, Freiheitsgrade, p-wert, 95%-Konfidenzintervall der Mittelwertsdifferenz und die Gruppenmittelwerte.

```
Two sample t-test
data: AV.IQ by UV.Priming
t = -2.89, df = 58, p-value = 0.002706
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
 -Inf -4.765489
sample estimates:
 mean in group Bloed mean in group Intelligent
      92.3400          103.6433
```

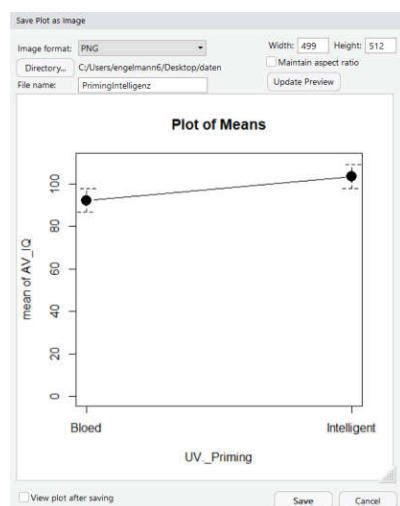
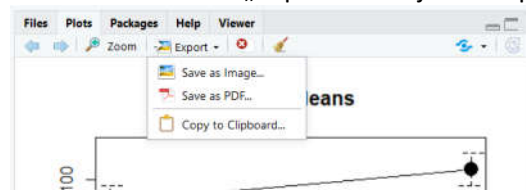
## 8) Ergebnisse und Graphiken speichern

Die Ergebnisse des t-Tests lassen sich am einfachsten speichern, wenn man diese mit der Maus markiert und mittels Strg+c in die Zwischenablage kopiert und dann mit Strg+v in ein Word-Dokument einfügt.

Graphiken werden im rechten unteren Fenster des R-workspace unter dem Reiter „Plots“ angezeigt (mit den Pfeilen kann zwischen verschiedenen im Laufe einer Sitzung angelegten Graphiken gewechselt werden).



Mit einem Klick auf „Export“ kann jede Graphik extern gespeichert werden (als PDF oder Bild):



Im nun erscheinenden Dialogfenster kann Dateiname und Speicherort für die Graphik gewählt werden