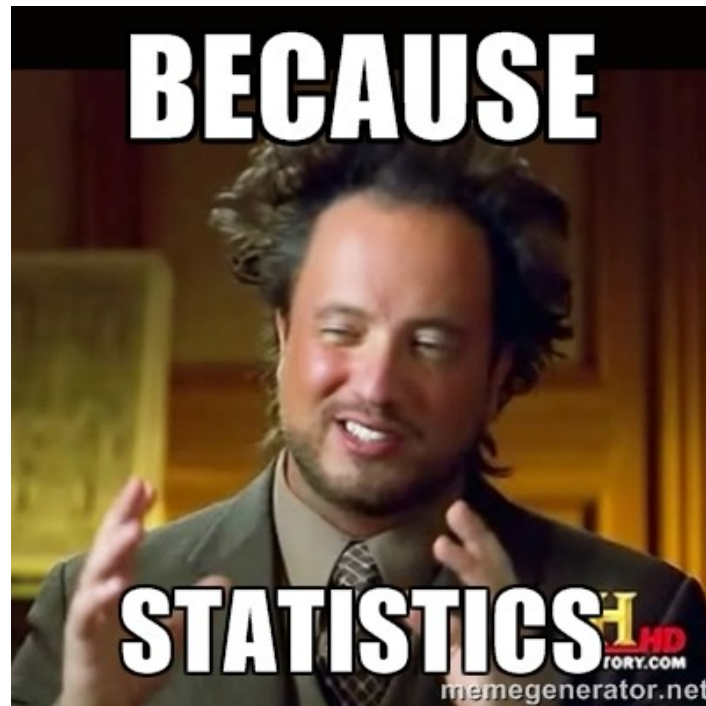


R

Der jämmerliche Versuch einer Einführung

# Scope

Soll eine kurze Einführung in R werden ohne gross in harte Programmierung einzutauchen.

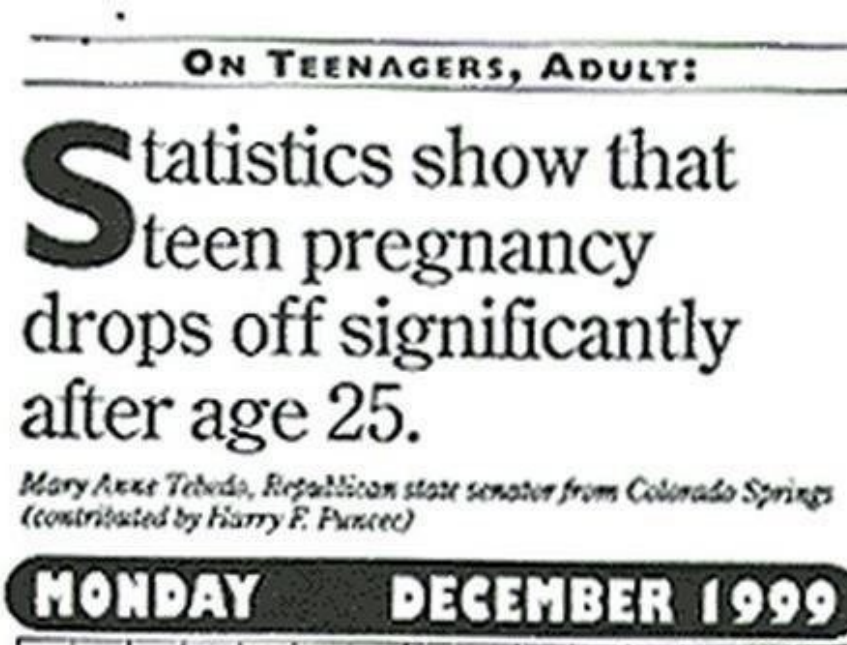


# Was ist R

R ist eine Programmiersprache um mit Daten zu hantieren.

Statistische Auswertung

Excel für Akademiker

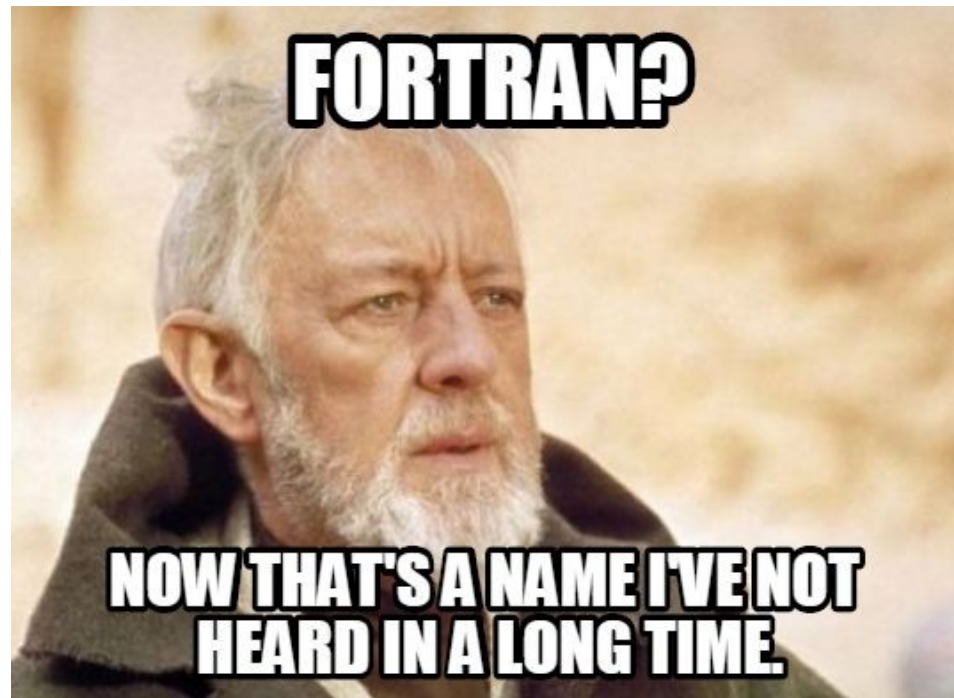


# Geschichte von R

1992 als Nachfolger von S

S war von Bell Labs und hatte nicht wirklich eine freie Lizenz

R ist in C, Fortran (heute noch) und R geschrieben.



# Bedienung

Interactive Mode oder eben als File: gewinn.R

Heute machen wir hauptsächlich Commandline Interpreter

```
noqqe@fmb ~> R
```

```
R version 3.4.3 (2017-11-30) -- "Kite-Eating Tree"  
Copyright (C) 2017 The R Foundation for Statistical Computing  
Platform: x86_64-apple-darwin17.3.0 (64-bit)
```

```
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.
```

```
>
```

```
>
```

# Wie sieht das aus?

```
#!/usr/bin/Rscript

euro <- function(x){
  y = format(x, digits=10, nsmall=2, decimal.mark=",", big.mark=".")
  paste(c(y), "EUR")
}

s = rev(read.table('~/.foo.txt'))
income = rev(s$V2)

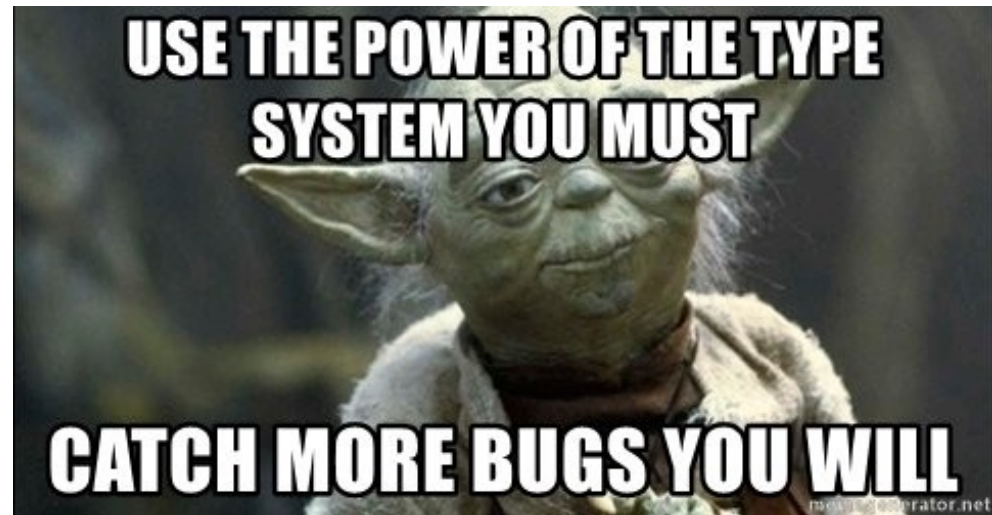
# Print some informations
euro(sum(income))
euro(mean(income))

# Graph cars using blue points overlaid by a line
plot(income, type="o", col="blue")

# Create a title with a red, bold/italic font
title(main="Gewinn", col.main="red", font.main=4)
```

# Datentypen

- Integer
- String
- Vector
- List
- Matrix
- data.frame



# Integer

```
> y  
[1] 15
```



# String

```
> y  
[1] "cccc"
```

# Vector (Basics)

Beste Sache in ganz R

```
> x <- c(1,4,6,7)
```

```
> x
```

```
[1] 1 4 6 7
```

# Vektoren (Arithmetik)

```
> x <- c(1,4,6,7)
```

```
> x > 3
```

```
[1] FALSE TRUE TRUE TRUE
```

```
> mean(x)
```

```
[1] 4.5
```

# Vektoren (mit Funktionen verwenden)

```
> sort(x)
```

```
[1] 1 4 6 7
```

```
> rev(sort(x))
```

```
[1] 7 6 4 1
```

# Vektoren (Arithmetik)

```
> 10 * x
```

```
[1] 10 40 60 70
```

```
> x + x
```

```
[1] 2 8 12 14
```

# Vektoren (Indexing)

```
> s = c("aa", "bb", "cc", "dd", "ee")  
> L = c(FALSE, TRUE, FALSE, TRUE, FALSE)  
> s[L]  
[1] "bb" "dd"
```

# Vektoren (Indexing)

```
> x ← c(1, 2, 3, 4, 5, 1, 2)
```

```
> x[x > mean(x)]
```

```
[1] 3 4 5
```

# Vektoren (Recycling)

```
> u = c(10, 20, 30)
```

```
> v = c(1, 2, 3, 4, 5, 6, 7, 8, 9)
```

```
> u + v
```

```
[1] 11 22 33 14 25 36 17 28 39
```



# List

```
> a <- c("foo", "bar", "baz")
```

```
> b <- c("alice", "bob")
```

```
> x <- list(a,b)
```

```
> x
```

```
[[1]]
```

```
[1] "foo" "bar" "baz"
```

```
[[2]]
```

```
[1] "alice" "bob"
```

# Matrix

```
> x <- matrix( c(2, 4, 3, 1, 5, 7), nrow=3, ncol=2 )
```

```
> x
```

```
      [,1] [,2]  
[1,]    2    1  
[2,]    4    5  
[3,]    3    7
```

# Data Frame

```
> d <-  
data.frame(drinks=c("Mate", "Coffee"), consumption=c(23, 42))  
> d  
  drinks consumption  
1  Mate           23  
2 Coffee          42
```

Auch etwas worin R sehr gut ist. Daten auf dem Terminal Formattieren.

# Data Frame

```
people = data.frame (  
age = c(32,34,12,41,18,23,43,22,19,24),  
height = c(177,166,165,174,156,184,191,179,182,180),  
sex = c('M','F','M','M','F','M','F','F','M','F'))
```

```
> people  
  age height sex  
1   32   177  M  
2   34   166  F  
3   12   165  M  
4   41   174  M  
5   18   156  F  
6   23   184  M  
7   43   191  F  
8   22   179  F  
9   19   182  M  
10  24   180  F
```

# Data Frame (Indexing)

```
> x <- data.frame(1:5, 6:10)
```

```
> x
```

```
  X1.5 X6.10  
1     1     6  
2     2     7  
3     3     8  
4     4     9  
5     5    10
```

```
> x[1,2]
```

```
[1] 6
```

```
> x[2,]
```

```
  X1.5 X6.10  
2     2     7
```

```
> x[,1]
```

```
[1] 1 2 3 4 5
```

# Funktionen

```
> a <- c(1,2,3)
> oddcount <- function(x) {
  k <- 0
  for (n in x) {
    if (n %% 2 == 1 ) k <- k+1
  }
  return(k)
}
```

```
> oddcount(a)
[1] 2
```

# Builtin Funktionen

```
rnorm(100) ## Random Normal Distribution
mean(x) ## Durchschnitt
sd(x) ## Standard Abweichung
seq(100) ## zähle bis 100
rep(98,4) ## wiederholt werte (repeat)
all(x > 8) ## Makro für IF Conditions, wenn ALLE dann
any(x > 8) ## wenn IRGENDEINS dann
sqrt(9) ## Wurzel ziehen
round(12.2) ## Runden auf nächsten Wert
head(x) ## einfach wie in Unix
tail(x) ## same here
subset(x,x > 5) ## Filtern innerhalb Vektoren nach Conditions
which(x,x > 5) ## Selbes, gibt aber die relativen positionen der Werte aus.
diff(x) ## berechnet die unterschiede innerhalb numerischer Vektoren
sign(x) ## "begradigt" negative und positive werte
length(x) ## länge von vektoren abfragen
sort(x)
order(x) ## sortieren mit realtiven index angaben als return
str(x) ## welche Struktur hat der Vector?
mode(x) ## was für eine Struktur liegt vor?
summary(x) ## erzähl mir alles was so geht über x?
```

# Builtin Funktionen

`print(x)` ## gib mir die print methode für den jeweiligen Datentypen

`class(x)` ## zeigt die Class an zb. "data.frame"

`apply(x,y,z)` ## funktion für jeden Wert in vecotr ausführen

`lapply(x,y,z)` ## selbes für liste

`sapply(x,y)` ## für liste, aber rückgabe als vector

`cbind(matrix,vector)` ## column an matrix binden

`rbind(matrix,vector)` ## row an matrix binden

`dim(matrix)` ## columns und rows anzahl einer matrix ausgeben

`nrow(x)` ## nur rows

`ncol(x)` ## nur columns

`attributes(x)` ## attribute einer klasse abfragen (zb. dim bei matrix)

`colnames(matrix)` ## columns namen geben (von matrix)

`rownames(matrix)` ## für rows

`names(j)` ## beschreibungen ausgeben bei listen

`unlist(j)` ## liste zu vector konvertieren

`unname(j)` ## alle namen entfernen



# apply

```
> t <- function(x) {  
  y <- x + 1  
  return(y)  
}  
  
> o <- c(1,4,56,8,12)  
> sapply(o,t)  
[1]  2  5 57  9 13
```

# lapply

```
> o <- c(1,4,56,8,12)
> p <- c(12,42,23,1337)
> q <- list(p,o)

> q
[[1]]
[1] 12 42 23 1337

[[2]]
[1] 1 4 56 8 12
```

# lapply

```
> lapply(q,mean)
[[1]]
[1] 353.5
```

```
[[2]]
[1] 16.2
```

```
> lapply(q,max)
[[1]]
[1] 1337
```

```
[[2]]
[1] 56
```

# apply

```
m<-matrix(1:100,nrow=10,ncol=10)
```

```
> m
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]
[1,]	1	11	21	31	41	51	61	71	81	91
[2,]	2	12	22	32	42	52	62	72	82	92
[3,]	3	13	23	33	43	53	63	73	83	93
[4,]	4	14	24	34	44	54	64	74	84	94
[5,]	5	15	25	35	45	55	65	75	85	95
[6,]	6	16	26	36	46	56	66	76	86	96
[7,]	7	17	27	37	47	57	67	77	87	97
[8,]	8	18	28	38	48	58	68	78	88	98
[9,]	9	19	29	39	49	59	69	79	89	99
[10,]	10	20	30	40	50	60	70	80	90	100

# apply

```
> apply(m, 1, mean)
```

```
[1] 46 47 48 49 50 51 52 53 54 55
```

```
> apply(m, 2, mean)
```

```
[1] 5.5 15.5 25.5 35.5 45.5 55.5 65.5 75.5  
85.5 95.5
```

# apply

```
> apply(m,1,function(x) x * 10)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]   10   20   30   40   50   60   70   80   90  100
[2,]  110  120  130  140  150  160  170  180  190  200
[3,]  210  220  230  240  250  260  270  280  290  300
[4,]  310  320  330  340  350  360  370  380  390  400
[5,]  410  420  430  440  450  460  470  480  490  500
[6,]  510  520  530  540  550  560  570  580  590  600
[7,]  610  620  630  640  650  660  670  680  690  700
[8,]  710  720  730  740  750  760  770  780  790  800
[9,]  810  820  830  840  850  860  870  880  890  900
[10,] 910  920  930  940  950  960  970  980  990 1000
```

# Daten einlesen

```
read.table("bla.csv",header=FALSE) ## Import von CSV Dateien
```

```
> x <- read.csv("Downloads/hurricanes.csv", header=TRUE)
```

```
> x
```

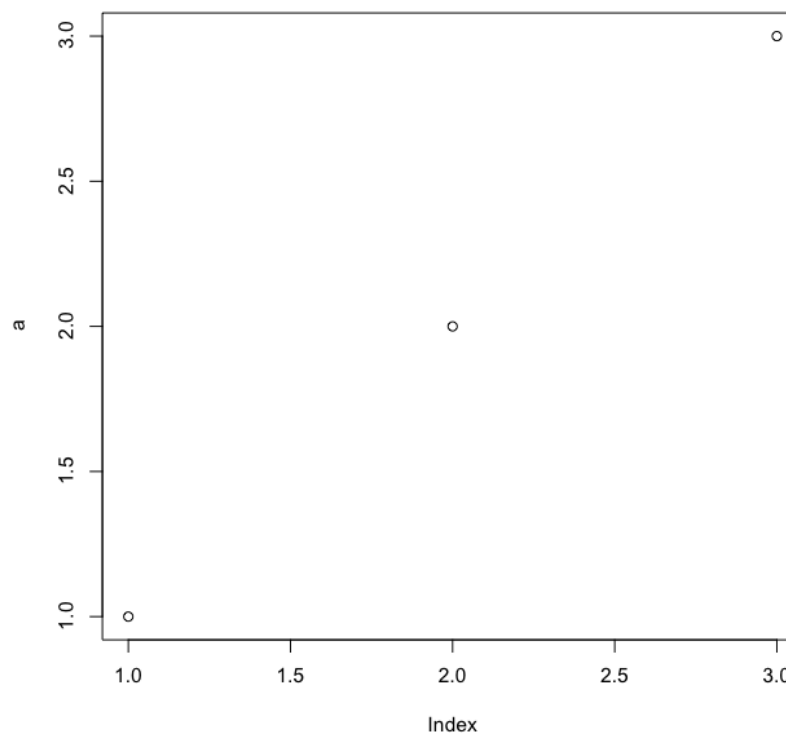
	Month	Average	X2005	X2006	X2007	X2008	X2009	X2010	X2011	X2012	X2013	X2014
1	May	0.1	0	0	1	1	0	0	0	2	0	0
2	Jun	0.5	2	1	1	0	0	1	1	2	2	0
3	Jul	0.7	5	1	1	2	0	1	3	0	2	2
4	Aug	2.3	6	3	2	4	4	4	7	8	2	2
5	Sep	3.5	6	4	7	4	2	8	5	2	5	2
6	Oct	2.0	8	0	1	3	2	5	1	5	2	3
7	Nov	0.5	3	0	0	1	1	0	1	0	1	0
8	Dec	0.0	1	0	1	0	0	0	0	0	0	0

```
scan(tf,"") ## import von textdateien
```

# Visualisierungen

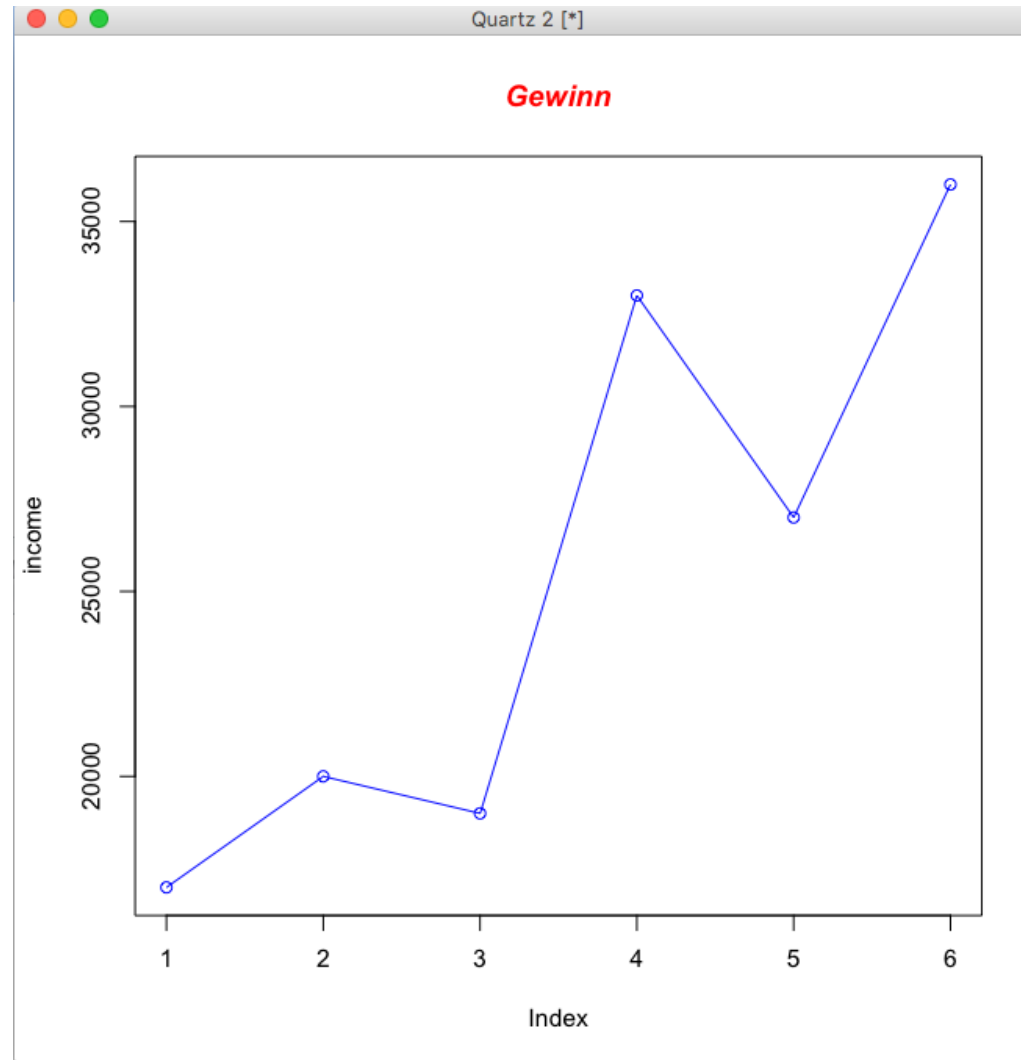
Builtin Visualisierung mittels `plot` und `lines`

```
> a <- c(1,2,3)  
> plot(a)
```



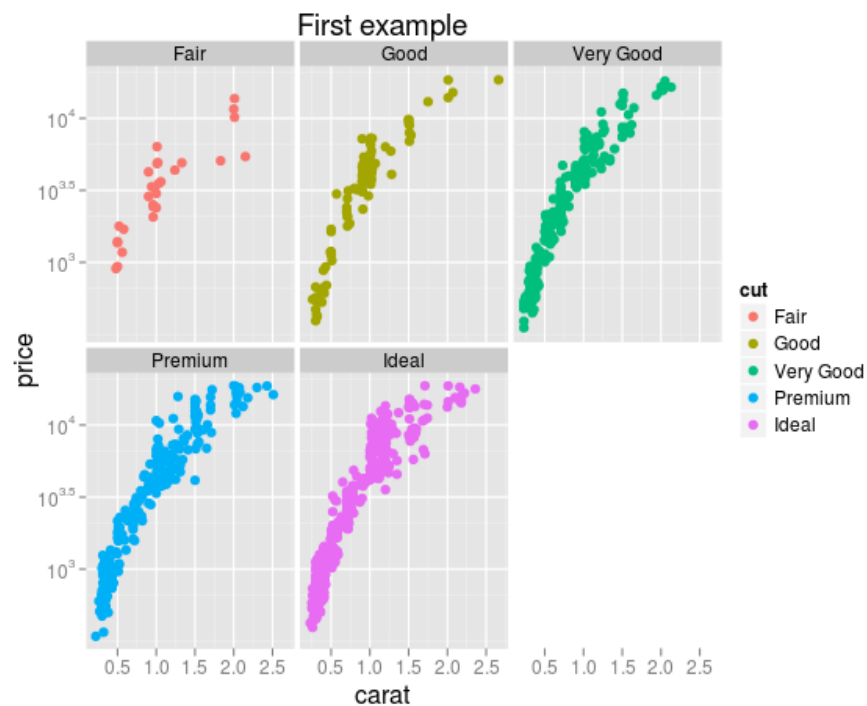


# Visualisierungen



# Visualisierungen (ggplot)

Heutzutage benutzt man aber `ggplot`, was ein externe Library ist.



# Was mach ich mit R?

Im Grunde kann man damit alles machen was man will, weil Turing Complete.



# Was macht die Industrie mit R

Marktforschern (Data Scientists) benutzt um Modelle zu bauen  
Sobald Modelle sich als valide erweisen werden sie dann von Programmierern in anderen besser skalierbareren Sprachen implementiert und auf den ganzen Datenbestand angewendet.  
Zum Beispiel mit Scala oder MapReduce/Yarn Stack von Hadoop

# R vs Kommerzielle Software



# Was kann ich mit R alles machen?

R und ich haben so eine Art Hopon-Hopoff Beziehung. Seit 2012 bastel ich immer mal wieder Phasenweise mehr und mal weniger damit herum.

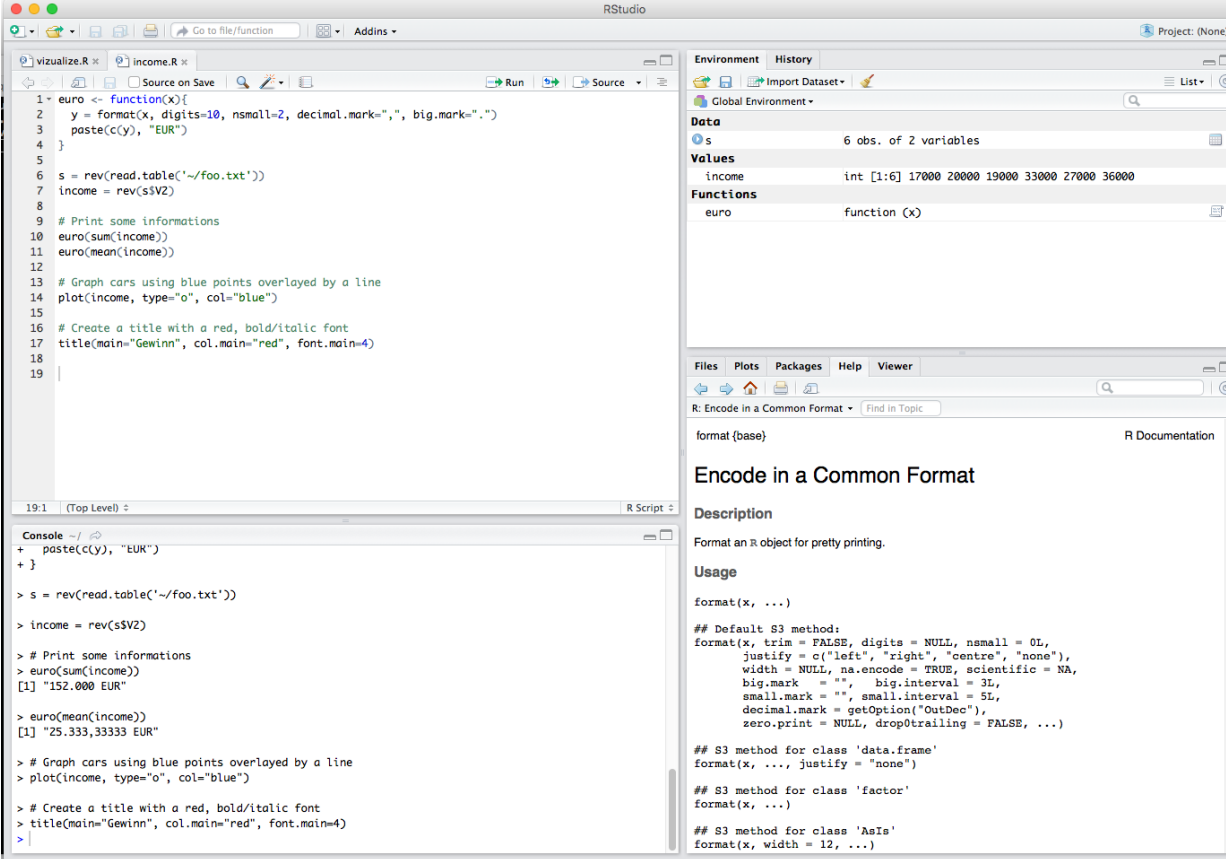
- Kontoführung
- Fahrrad Stats (Distanz, Herzrate, Geschwindigkeitsentwicklung)

# Was kann ich mit R alles machen?

## GPX Karte malen

# Wie entwickle ich R ?

## Interaktiv / Normaler Editor / R-Studio (IDE)



The screenshot displays the RStudio IDE interface. The main editor window shows R code for a function named 'euro' and a script named 'income.R'. The console window shows the execution of the code, including the output of 'euro(sum(income))' and 'euro(mean(income))'. The right-hand pane shows the Environment and History tabs, and the lower right pane shows the R Documentation for the 'format' function.

```
1 euro <- function(x){
2   y = format(x, digits=10, nsmall=2, decimal.mark=".", big.mark=".")
3   paste(c(y), "EUR")
4 }
5
6 s = rev(read.table("~/foo.txt"))
7 income = rev(s$V2)
8
9 # Print some informations
10 euro(sum(income))
11 euro(mean(income))
12
13 # Graph cars using blue points overlaid by a line
14 plot(income, type="o", col="blue")
15
16 # Create a title with a red, bold/italic font
17 title(main="Gewinn", col.main="red", font.main=4)
18
19
```

```
> paste(c(y), "EUR")
+ }
+ }
+ }

> s = rev(read.table("~/foo.txt"))

> income = rev(s$V2)

> # Print some informations
> euro(sum(income))
[1] "152.000 EUR"

> euro(mean(income))
[1] "25.333,33333 EUR"

> # Graph cars using blue points overlaid by a line
> plot(income, type="o", col="blue")

> # Create a title with a red, bold/italic font
> title(main="Gewinn", col.main="red", font.main=4)
>
```

**Environment**  
Global Environment  
Data  
s 6 obs. of 2 variables  
Values  
income int [1:6] 17000 20000 19000 33000 27000 36000  
Functions  
euro function (x)

**R Documentation**  
Encode in a Common Format  
Description  
Format an R object for pretty printing.  
Usage  
format(x, ...)  
## Default S3 method:  
format(x, trim = FALSE, digits = NULL, nsmall = 0L,  
justify = c("left", "right", "centre", "none"),  
width = NULL, na.encode = TRUE, scientific = NA,  
big.mark = "", big.interval = 3L,  
small.mark = "", small.interval = 5L,  
decimal.mark = getOption("OutDec"),  
zero.print = NULL, drop0trailing = FALSE, ...)  
## S3 method for class 'data.frame'  
format(x, ..., justify = "none")  
## S3 method for class 'factor'  
format(x, ...)  
## S3 method for class 'AsIs'  
format(x, width = 12, ...)



# Das R-Universum

- \* Grosser Frauenanteil [About us – R-Ladies Global](<https://rladies.org/about-us/>)
- \* Packages (CRAN) [The Comprehensive R Archive Network] (<https://cran.r-project.org/>)
- \* Diverse Bücher
- \* Data Scientist is der heisse Scheiss.



# Schwächen von R

RAM.

Sobald deine Daten nicht mehr in deinen RAM passen ist es vorbei.

Workaround: Wir haben riesige Maschinen bei Amazon hochgefahren, brauchen sie aber meistens nicht weil die Data Scientists eh nur mit kleinen Datenmengen modellieren.

# Ende

Danke <3

```
c(rep(NA, 16), "BATMAN!!!")
```