# Quick & diRty

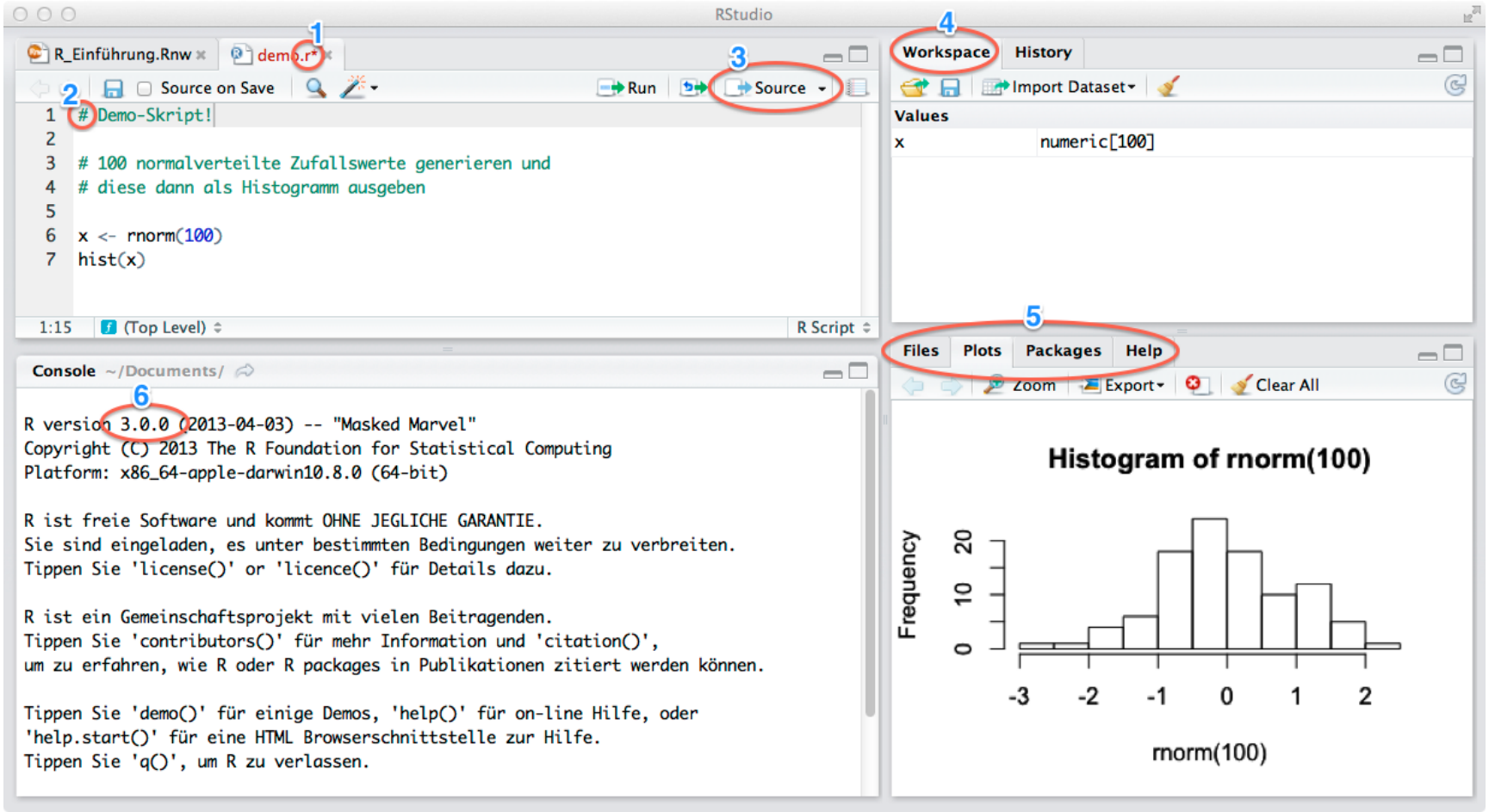
## Eine pragmatische Einführung in R
## (Seminar Forschungsevaluation, Universität Wuppertal)

Stephan Holtmeier

kibit GmbH, stephan@holtmeier.de

31. Mai 2013

# Unsere Entwicklungsumgebung: RStudio



# Erste Schritte mit RStudio

## Voraussetzungen

**R** (>v2.11) und **RStudio** (> v0.97) müssen installiert sein

1. Ein R-Skript endet auf `.r`
2. Ein Kommentar (grün) beginnt mit # und wird von R "überlesen"
3. `Source` führt ein komplettes R-Skript aus
   `Run` führt einen markierten Teilbereich aus
4. Im **Workspace** werden *Werte*, *Daten*, *Ergebnisse* und *Grafiken* angezeigt, die "flüchtig"gespeichert sind
5. **Files** = Dateibrowser
   **Plots** = Grafikausgabe
   **Packages** = Zusatzpakete betrachten und installieren
   **Help** = sehr hilfreiche Hilfe
6. **Console** = Textausgabe (und -eingabe)

# Direkt mit R loslegen...

1. Schnödes Rechnen (`+`, `-`, `*`, `/`)
2. Wahr oder falsch? (`<`, `>`, `==`, `TRUE`, `FALSE`)
3. Variablen-Zuweisung (`<-`)
4. Funktionen (`sum()`, `rep()`, `sqrt()`, `etc.`)

**Achtung beim =**

Wir verwenden möglichst nicht das Zeichen =. Stattdessen weisen wir einer Variablen etwas mit <- zu. Wollen wir Gleichheit überprüfen, verwenden wir ==.

# Direkt mit R loslegen...

1. Schnödes Rechnen (`+`, `-`, `*`, `/`)
2. Wahr oder falsch? (`<`, `>`, `==`, `TRUE`, `FALSE`)
3. Variablen-Zuweisung (`<-`)
4. Funktionen (`sum()`, `rep()`, `sqrt()`, `etc.`)

**Achtung beim =**

Wir verwenden möglichst nicht das Zeichen =. Stattdessen weisen wir einer Variablen etwas mit <- zu. Wollen wir Gleichheit überprüfen, verwenden wir ==.

# Direkt mit R loslegen...

1. Schnödes Rechnen (`+`, `-`, `*`, `/`)
2. Wahr oder falsch? (`<`, `>`, `==`, `TRUE`, `FALSE`)
3. Variablen-Zuweisung (`<-`)
4. Funktionen (`sum()`, `rep()`, `sqrt()`, `etc.`)

**Achtung beim =**

Wir verwenden möglichst nicht das Zeichen =. Stattdessen weisen wir einer Variablen etwas mit <- zu. Wollen wir Gleichheit überprüfen, verwenden wir ==.

# Direkt mit R loslegen...

1. Schnödes Rechnen (`+`, `-`, `*`, `/`)
2. Wahr oder falsch? (`<`, `>`, `==`, `TRUE`, `FALSE`)
3. Variablen-Zuweisung (`<-`)
4. Funktionen (`sum()`, `rep()`, `sqrt()`, `etc.`)

**Achtung beim =**

Wir verwenden möglichst nicht das Zeichen =. Stattdessen weisen wir einer Variablen etwas mit <- zu. Wollen wir Gleichheit überprüfen, verwenden wir ==.

# Direkt mit R loslegen...

1. Schnödes Rechnen (`+`, `-`, `*`, `/`)
2. Wahr oder falsch? (`<`, `>`, `==`, `TRUE`, `FALSE`)
3. Variablen-Zuweisung (`<-`)
4. Funktionen (`sum()`, `rep()`, `sqrt()`, `etc.`)

**Achtung beim =**

Wir verwenden möglichst nicht das Zeichen =. Stattdessen weisen wir einer Variablen etwas mit <- zu. Wollen wir Gleichheit überprüfen, verwenden wir ==.

# Am Anfang steht ein Datensatz

R kann sehr viele verschiedene Formate einlesen, unter anderem

1. SPSS
2. Excel
3. SQL-Datenbanken

## CSV-Format

Das einfachste (und beste) Datenformat besteht jedoch aus einer simplen Textdatei. Die Endung `.csv` steht für *comma seperated values*. Auch ein Semikolon oder ein Tabulator sind geeignete Trenner. Die erste Zeile wird für die Variablennamen verwendet.

```
bsp <- read.csv("bsp.csv")
```

# Vektoren mit Werten füllen

Ein Vektor ist schlicht eine Liste von Werten. `c()` steht für "combine"

```
vektor.1 <- c(3, 6, 9, 15, 15)
vektor.2 <- seq(1, 10, 0.5)
vektor.3 <- c("Köln", "Wuppertal", "Dortmund", "Essen")
vektor.4 <- rep(vektor.3, 10)
```

# Auf Inhalte in Vektoren wieder zugreifen

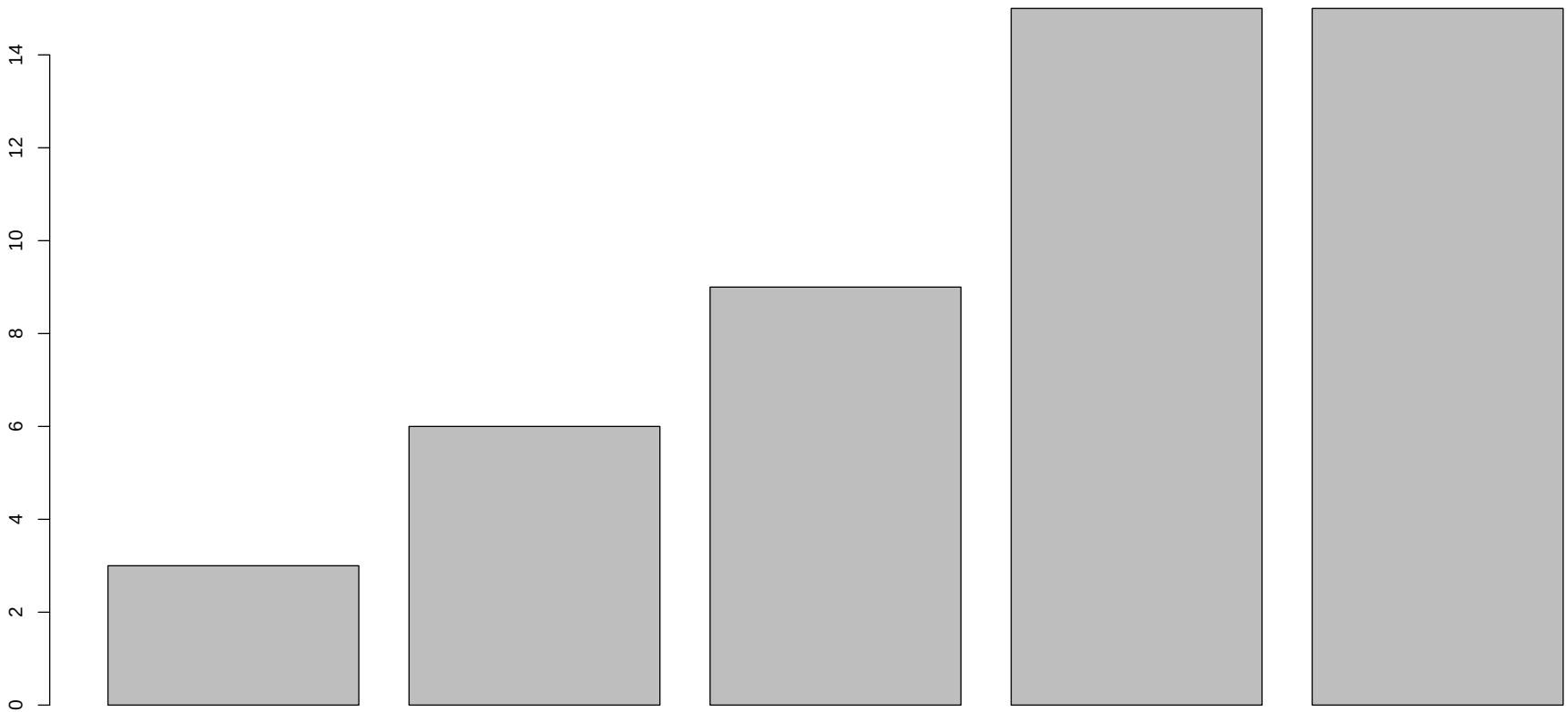
Der Zugriff auf einzelne Werte innerhalb eines Vektors erfolgt über einen `[INDEX]`

```
vektor.1[3]  #Wert an Position 3

## [1] 9

vektor.2[c(5, 6, 7)]  #Werte an den Positionen 5,6 & 7

## [1] 3.0 3.5 4.0

vektor.3[5] <- c("Bochum")  #ergänzen
print(vektor.3, quote = FALSE)

## [1] Köln       Wuppertal Dortmund  Essen
## [5] Bochum
```

# Einen Vektor grafisch darstellen, z.B. als Balkendiagramm

Zur Erinnerung: `vektor.1 <- c(3, 6, 9, 15, 15)`

```
barplot(vektor.1)
```



# Mit Vektoren arbeiten

Zur Erinnerung: `vektor.1 <- c(3, 6, 9, 15, 15)`

```
print(vektor.1/3)  #Multiplikation aller Werte mit 5

## [1] 1 2 3 5 5

vektor.1 == c(3, 6, 9, 12, 15)  #Vergleichen

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

# Vektoren mit Missings (NA)

```
vektor.5 <- c(3, 5, 2, 9, NA, 6, 9)
print(vektor.5)

## [1]  3  5  2  9 NA  6  9

sum(vektor.5)  #alle Werte aufsummieren

## [1] NA

sum(vektor.5, na.rm = TRUE)  #jetzt aber!

## [1] 34
```

# Was ist eine Matrix?

Eine Matrix hat mehrer Dimensionen, oft zwei. Sie ist perfekt geeignet, um ganze Datensätze aufzunehmen. Z.B. in den Reihen Versuchspersonen und in den Spalten Variablen.

```
matrix.1 <- matrix(seq(1, 12), 3, 4)  #3*4-Matrix befüllen
print(matrix.1)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    4    7   10
## [2,]    2    5    8   11
## [3,]    3    6    9   12
```

# Auf Inhalte einer Matrix wieder zugreifen

Der Zugriff erfolgt ähnlich, dem auf Vektoren: `[ZEILE, SPALTE]`

```
print(matrix.1[2, ])  #ganze zweite Spalte

## [1]  2  5  8 11

print(matrix.1[, 2])  #ganze zweite Zeile

## [1] 4 5 6

print(matrix.1[2, 2])  #Wert in Zeile 2 und Spalte 2

## [1] 5
```

# Faktoren sind für kategoriale Daten

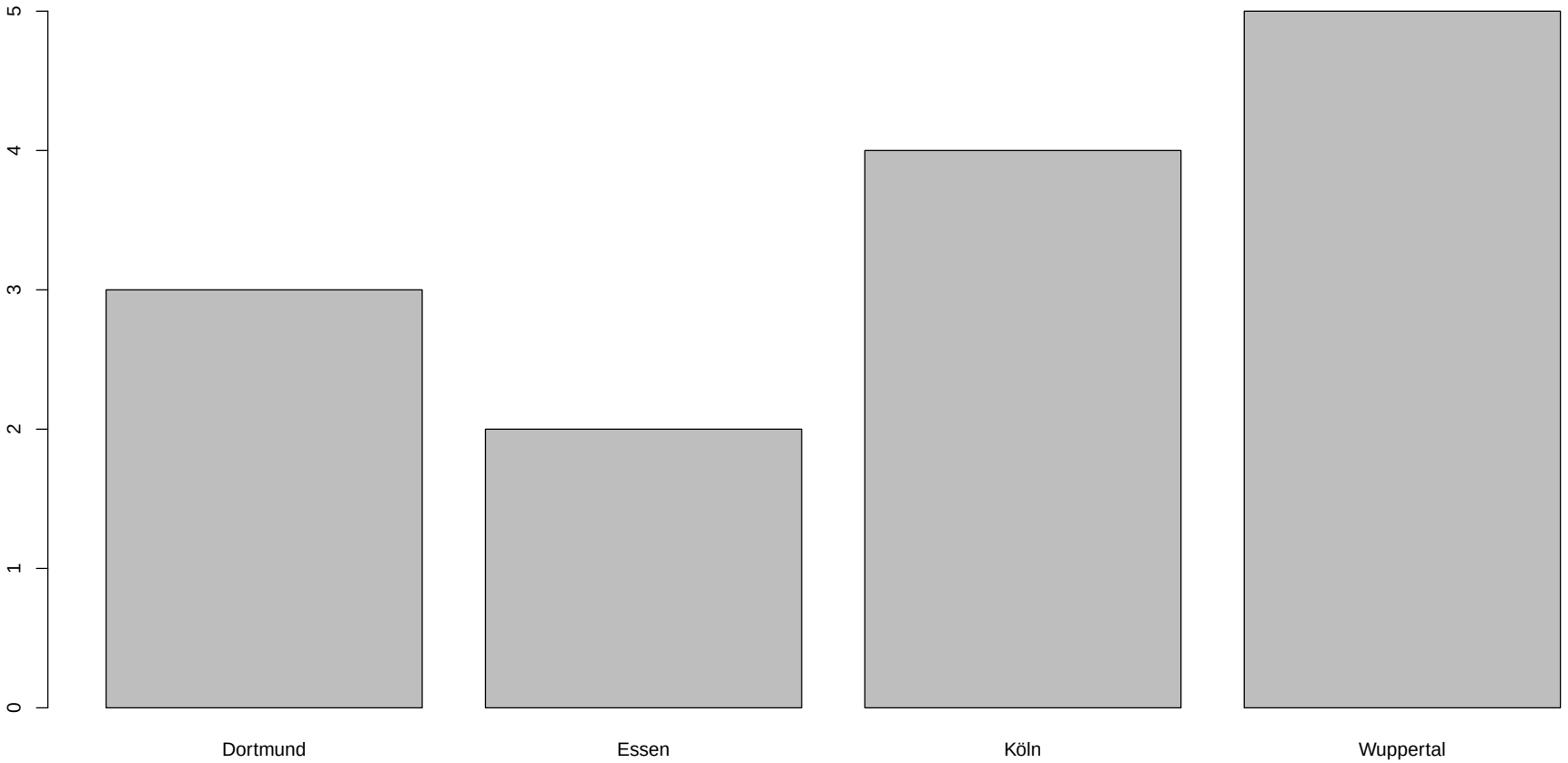
```
wohnort <- factor(c("Köln", "Wuppertal", "Dortmund",
    "Essen", "Köln", "Köln", "Wuppertal", "Köln",
    "Dortmund", "Wuppertal", "Essen", "Wuppertal",
    "Dortmund", "Wuppertal"))
print(wohnort)
```

```
##  [1] Köln      Wuppertal Dortmund  Essen
##  [5] Köln      Köln      Wuppertal Köln
##  [9] Dortmund  Wuppertal Essen     Wuppertal
## [13] Dortmund  Wuppertal
## Levels: Dortmund Essen Köln Wuppertal
```

# Faktoren visualisieren

Faktoren werden von den Plot-Funktionen als solche anders behandelt

```
plot(wohnort)
```



# Unser täglich Brot: Dataframes

Zum Schluss noch die Datenstruktur, die unterschiedliche Typen aufnehmen kann - alo einen kompletten Datensatz

```
alter <- c(24, 30, 21, 21, 34, 28, 23, 51, 23, 43,
    33, 22, 26, 26)
vegetarier <- c(TRUE, FALSE, FALSE, FALSE, TRUE, FALSE,
    FALSE, FALSE, TRUE, TRUE, FALSE, FALSE, FALSE,
    FALSE)
datensatz <- data.frame(wohnort, alter, vegetarier)
head(datensatz, n = 3)

##     wohnort alter vegetarier
## 1      Köln    24       TRUE
## 2 Wuppertal    30      FALSE
## 3  Dortmund    21      FALSE
```

# Auf die Inhalte von Dataframes zugreifen

```
print(datensatz$alter)

##  [1] 24 30 21 21 34 28 23 51 23 43 33 22 26 26

print(datensatz$alter[3])

## [1] 21
```

Wie ist der Datensatz eigentlich strukturiert?

```
str(datensatz)

## 'data.frame': 14 obs. of  3 variables:
##  $ wohnort   : Factor w/ 4 levels "Dortmund","Essen",..: 3
##  $ alter     : num  24 30 21 21 34 28 23 51 23 43 ...
##  $ vegetarier: logi  TRUE FALSE FALSE FALSE TRUE FALSE ...
```

# subsetting!

```
subset(datensatz, vegetarier == TRUE)  # Fälle auswählen
```

```
##       wohnort alter vegetarier
## 1        Köln    24       TRUE
## 5        Köln    34       TRUE
## 9   Dortmund    23       TRUE
## 10 Wuppertal    43       TRUE
```

```
subset(datensatz, select = c(wohnort, alter))  # Spalten wähle
```

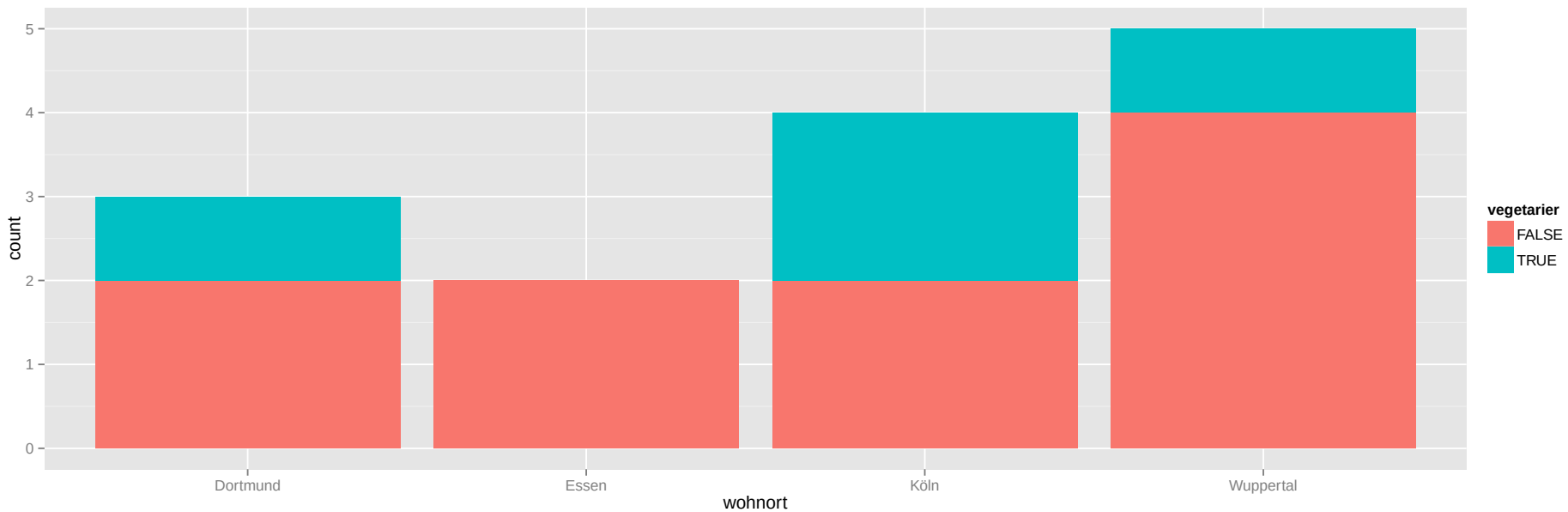
```
##       wohnort alter
## 1        Köln    24
## 2  Wuppertal    30
## 3   Dortmund    21
## 4       Essen    21
## 5        Köln    34
```

# ggplot2 - The Grammar of Graphics I

## ggplot2-Paket installieren

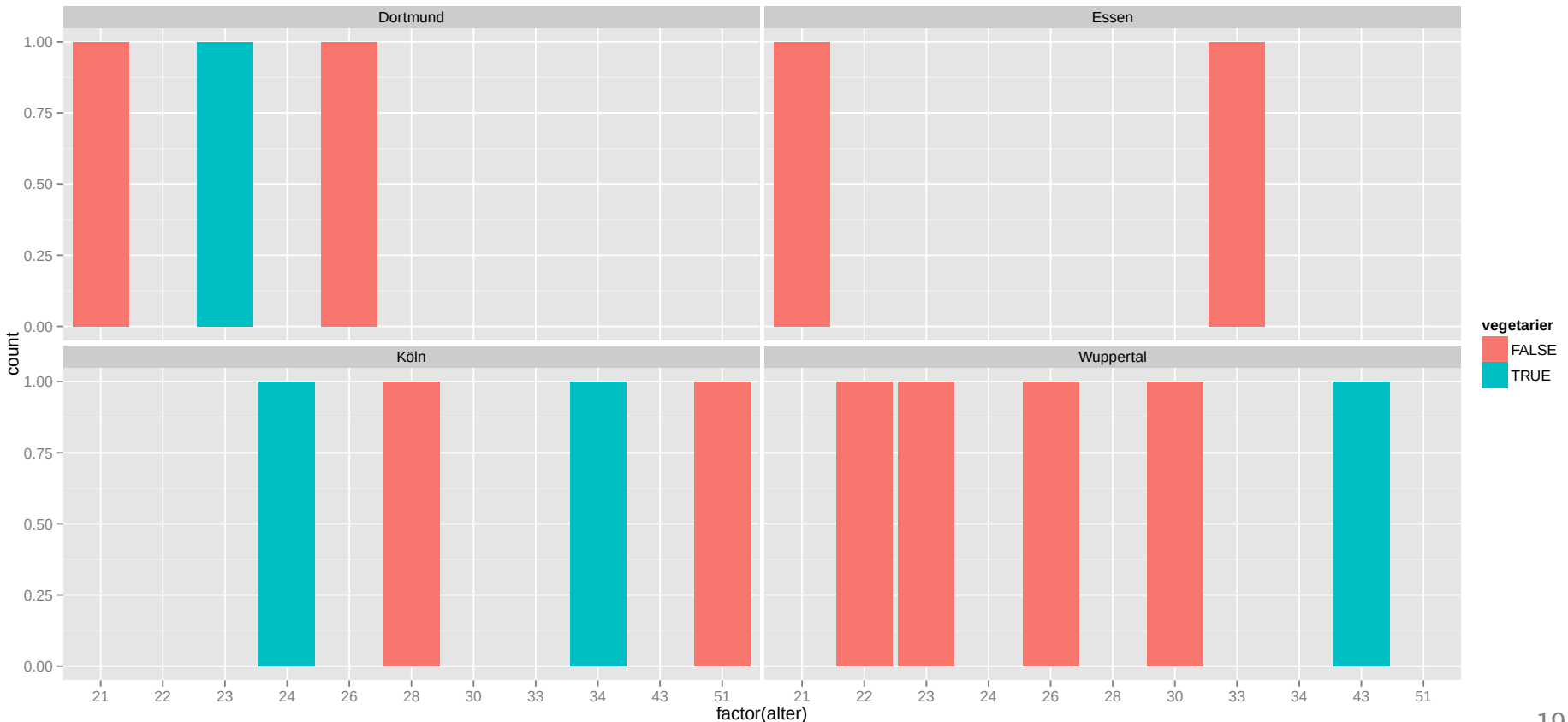
```
install.packages("ggplot2")
```

```
library(ggplot2)  #Paket laden
ggplot(datensatz, aes(x = wohnort, fill = vegetarier)) +
    geom_bar()
```



# ggplot2 - The Grammar of Graphics II

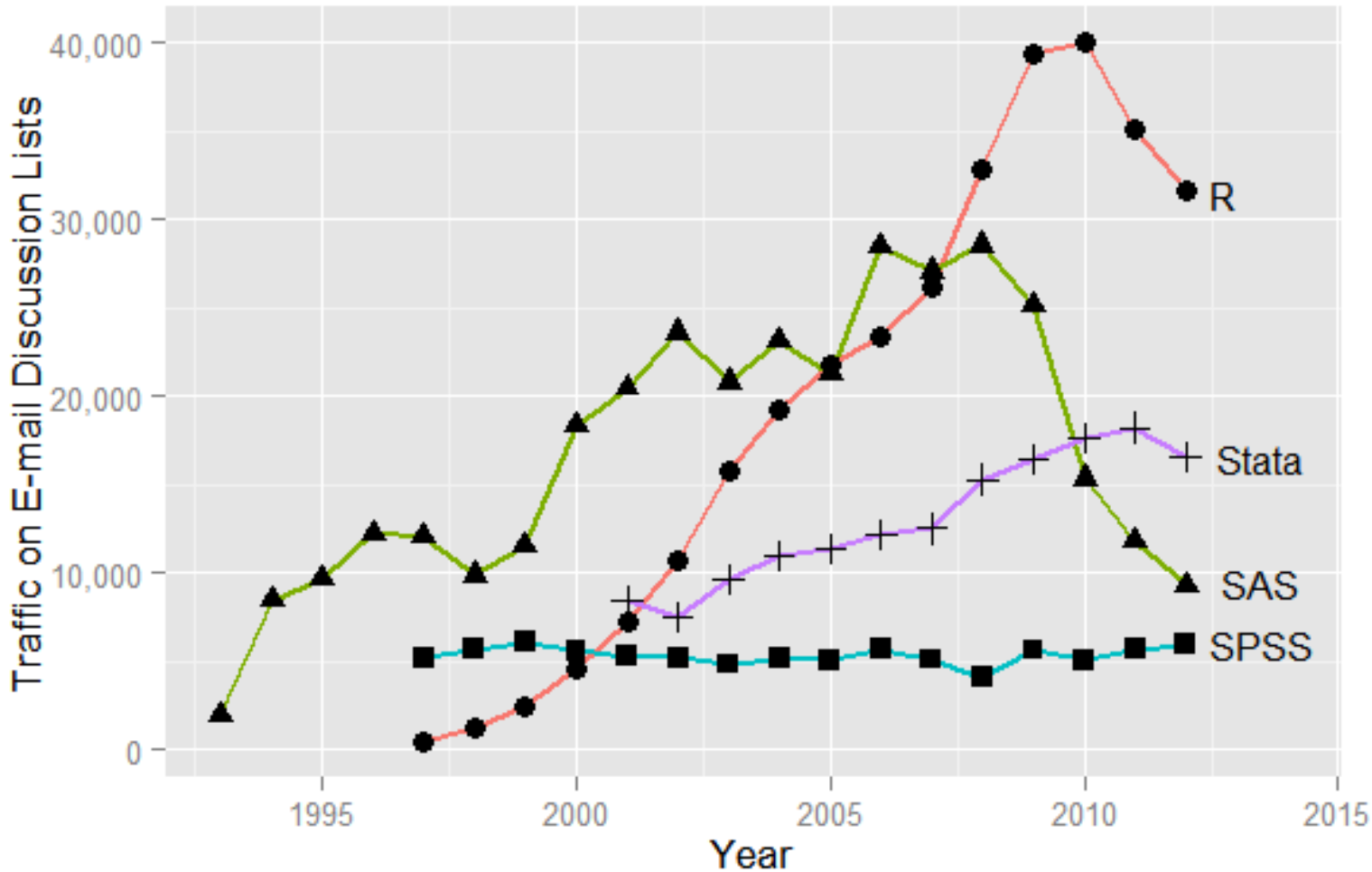
```
library(ggplot2)  #Paket laden
ggplot(datensatz, aes(x = factor(alter), fill = vegetarier)) +
    geom_bar() + facet_wrap(~wohnort)
```



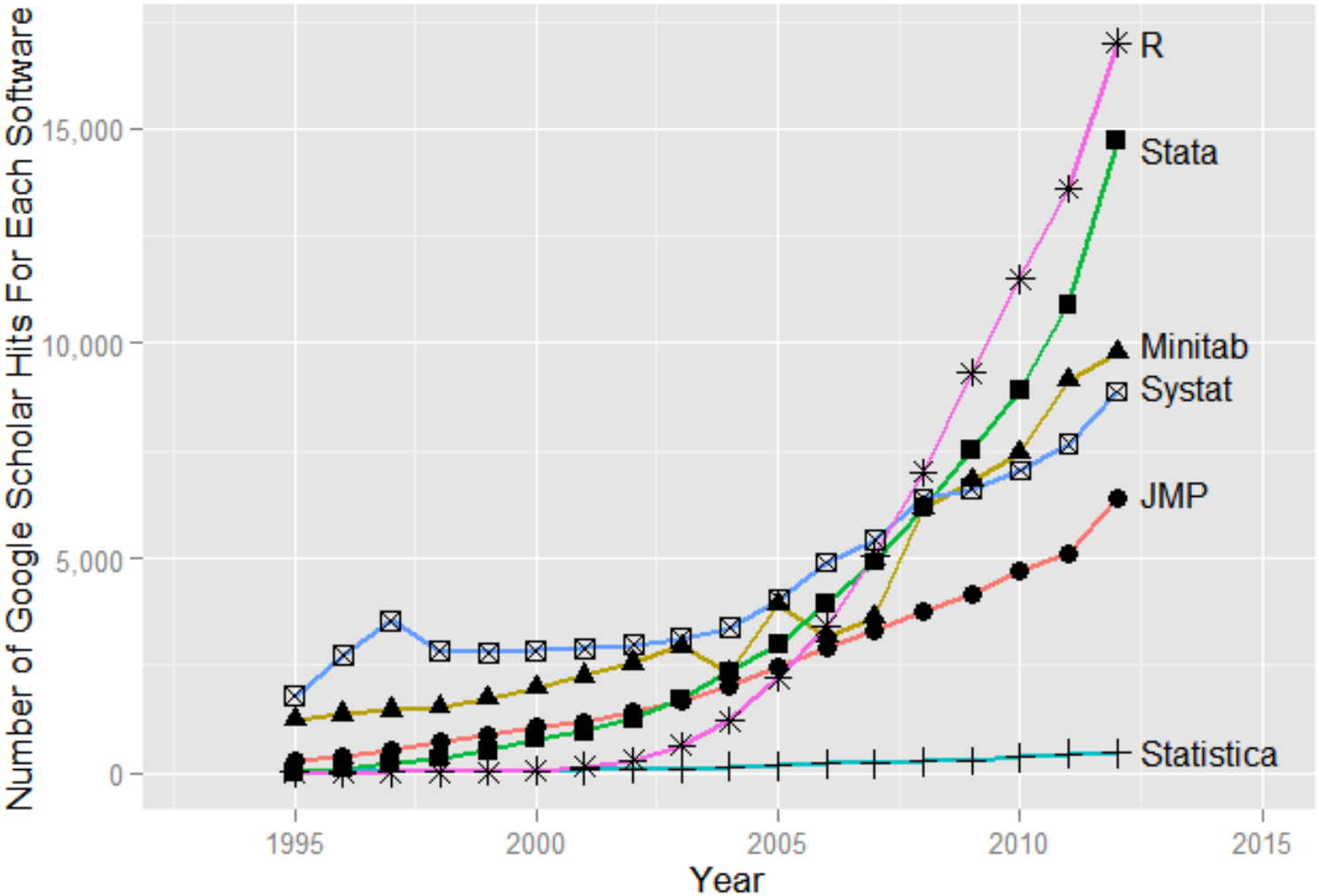
# Lust auf mehr? 5 Tips zur weiteren Vertiefung

1. Toller Online-Kurs: http://tryr.codeschool.com
2. Quick-R (Anleitung mit Codebeispielen): http://www.statmethods.net
3. Videos - R in zwei Minuten: http://www.twotorials.com
4. R-bloggers - R news and tutorials: http://www.r-bloggers.com
5. ggplot2 - Online Referenz: http://docs.ggplot2.org/current/

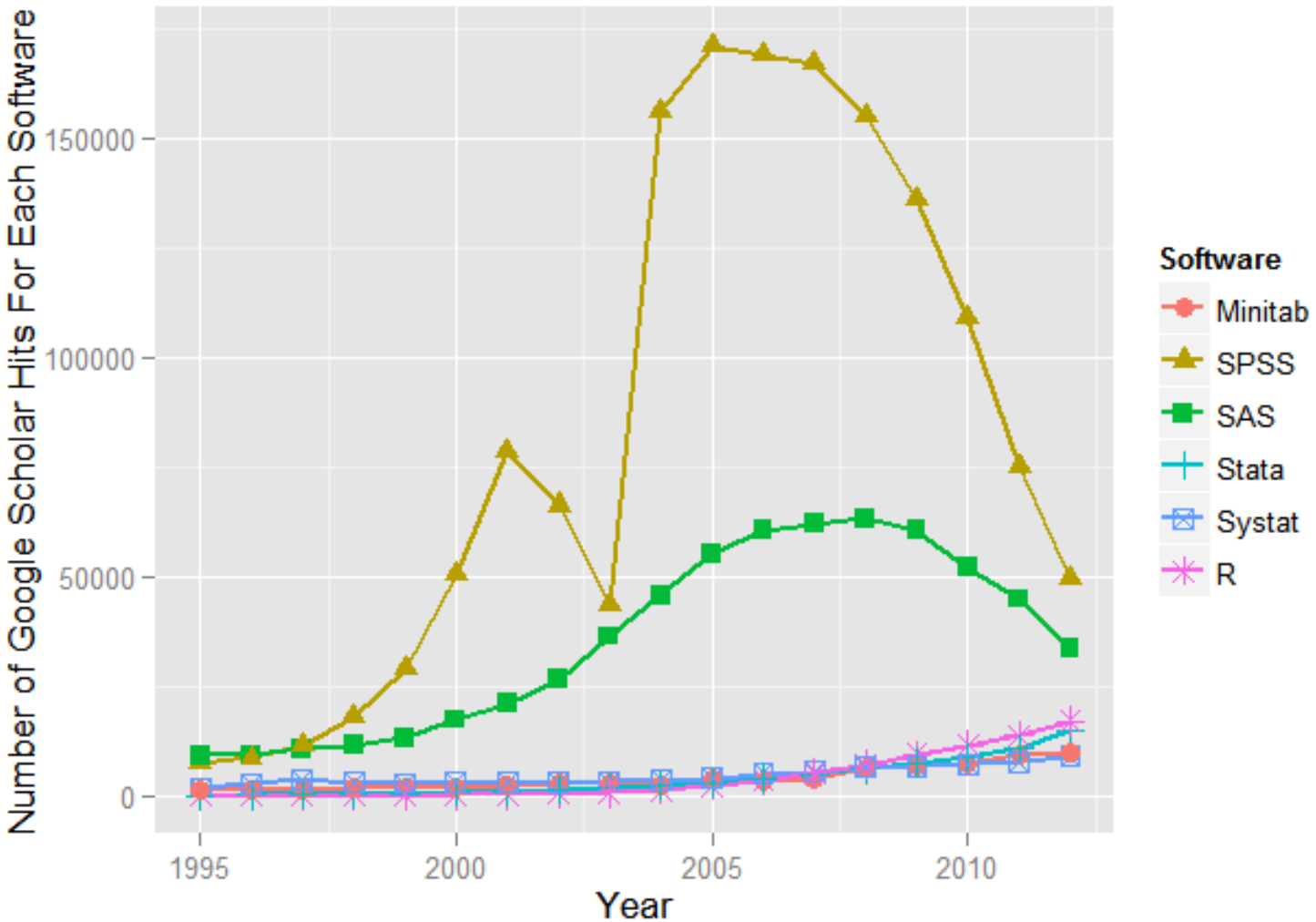
# Traffic on E-Mail discussion lists



# Google Scholar Hits (without SPSS & SAS)



# Google Scholar Hits (incl. SPSS & SAS)



# WICHTIG: Hausaufgabe

1. install.packages('psych')
2. install.packages('GPArotation')
3. install.packages('cluster')
4. install.packages('MASS')
5. Download Datensätze: fa_data.csv, bsp.csv, ac.csv