Einführung R

WS 2023-24

DI Emil Marinov

SPENGERGASSE
ausbildung mit zukunft

Übersicht

SPENGERGASSE
ausbildung mit zukunft

Warum R?

Was ist R?

- Software für statistische Analysen und Visualisierungen
- Programmiersprache
- open source, freie Software, lizensiert unter GNU General Public Licence
- läuft unter Windows, MacOS und UNIX
- https://www.r-project.org/
- erweiterbar durch zusätzliche Bibliotheken (packages):
  > 18.000 frei verfügbar auf CRAN (Comprehensive R Archive Network: https://cran.r-project.org/)

Warum R?

- reproduzierbare Ergebnisse
- leicht automatisierbar und integrierbar mit anderen Technologien
- einfacher Umgang mit großen Datensätzen aus verschiedenen Quellen (Excel ist beschränkt auf 1.048.576 Zeilen und 16.384 Spalten)
- Erstellung von automatisierten Berichten und interaktiven Apps
- kostenlos
- weit verbreitet und dadurch zahlreiche Handbücher, Kurse, Internet-Foren vorhanden
- großer Funktionsumfang im Bereich der statistischen Auswertungen und Visualisierungen
- einfach erweiterbar durch R Packages

(siehe auch https://www.northeastern.edu/graduate/blog/r-vs-excel/)

SPENGERGASSE
ausbildung mit zukunft

Basics

Basics

Definition von Variablen:

```
x <- 5
y <- "apple"
```

Operatoren:

+	plus
-	minus
*****	mal
/	geteilt
^	hoch
==	ist gleich
!=	ist nicht gleich

<=	kleiner gleich
>=	größer gleich
&	und (elementweise)
|	oder (elementweise)
%in%	ist in
is.na(x)	Test auf fehlenden Wert

Basics

Mathematische Funktionen

max(x)	größtes Element
min(x)	kleinstes Element
sum(x)	Summe aller Elemente
round(x, n)	Runden auf n Dezimalstellen
signif(x, n)	Runden auf n signifikante Stellen
mean(x)	arithmetisches Mittel
median(x)	Median
var(x)	Varianz
sd(x)	Standardabweichung

Basics

Datentypen und Umwandlungen

Logical	`as.logical`	logische Werte (TRUE, FALSE)
Integer	`as.integer`	ganze Zahlen
Numeric	`as.numeric`	reelle Zahlen
Character	`as.character`	Zeichenkette (String, Text)
Factor	`as.factor`	kategoriale Variable mit vorgegebenen Ausprägungen
Vector	`as.vector`	Vektor mit einheitlichen Datentyp
List	`as.list`	Liste mit Daten (unterschiedlicher Datentyp möglich)
Matrix	`as.matrix`	Tabelle mit einheitlichem Datentyp
Data frame	`as.dataframe`	Tabelle, deren Spalten unterschiedliche Datentypen haben können

Vektoren

Vektoren erzeugen

`x <- c(1, 3, 7, 5, 9)`	`1 3 7 5 9`	Vektor durch Aufzählen definieren
`1:5`	`1 2 3 4 5`	ganzzahlige Zahlenfolge
`seq(2,3, by=0.5)`	`2 2.5 3`	Zahlenfolge erzeugen
`rep(1:2, times=3)`	`1 2 1 2 1 2`	Wiederholung von Zahlen

auf Elemente von Vektoren zugreifen

`x[4]`	`5`	4. Element eines Vektores
`x[-4]`	`1 3 7 9`	alle Element außer dem 4. Element
`x[2:4]`	`3 7 5`	2. – 4. Element eines Vektors
`x[c(1, 5)]`	`1 9`	1. und 5. Element
`x[x < 5]`	`1 3 7`	alle Elemente bis zum ersten Datenpunkt 5
`x[x %in% c(1, 5)]`	`1 5`	alle Elemente, die in (1, 5) enthalten sind

Vektoren

```
x <- c(1, 3, 7, 5, 3, 4, 5, 3)
```

Funktionen für Vektoren

`sort(x)`	`1 3 3 3 4 5 5 7`	Werte sortieren
`table(x)`	`1 3 4 5 7` `1 3 1 2 1`	absolute Häufigkeiten der Werte
`rev(x)`	`3 5 4 3 5 7 3 1`	Vektor umdrehen
`unique(x)`	`1 3 7 5 4`	verschiedene Werte eines Vektors

Listen

Liste definieren

```
l <- list(x=1:5, y=c("a", "b"))
```

auf Elemente einer Liste zugreifen

`l[[2]]`	`"a" "b"`	zweites Element einer List
`l[1]`	`1 2 3 4 5`	neue Liste mit erstem Element der Liste l
`l$x`	`1 2 3 4 5`	auf Element mit Namen zugreifen
`l["y"]`	`"a" "b"`	neue Liste mit dem Element “y”

Schleife über Listenelemente

```
l <- lapply(list_name, function(x){....})
```

Matrizen

Matrix definieren

```
x <- 1:9
m <- matrix(x, nrow = 3, ncol = 3)
m
```

```
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

auf Elemente einer Matrix zugreifen

`m[2,  ]`	`2 5 8`	Auswahl einer Zeile
`m[ , 1]`	`1 2 3`	Auswahl einer Spalte
`m[2, 3]`	`8`	Auswahl eines Elements

Matrix-operationen

`t(m)`	Matrix transponieren
`m %*% n`	Matrix multplizieren

Dataframes

Matrix definieren

```
df <- data.frame(x = 1:3,
      y = c("a", "b", "c"))
```

```
> df
  x y
1 1 a
2 2 b
3 3 c
```

auf Elemente eines Dataframes zugreifen

`df$x`	`1 2 3`	Spalte mit Namen auswählen
`df[[2]]`	`"a" "b" "c"`	Spalte mit Index auswählen
`df[1,2]`	`"a"`	eine Element mit Zeilen und Spaltenindex auswählen

Funktionen für Dataframes

`head``(df)`	ersten 6 Zeilen eines Dataframes
`nrow``(df)`	Zeilenanzahl

Kontrollstrukturen und Funktionen

FOR-Schleife

```
for(i in 1:n){
    ....
}
```

WHILE-Schleife

```
while(condition){
    ....
}
```

IF-Abfrage

```
if(condition){
    ....
} else {
    ....
}
```

Funktion defniernen

```
func_name <- function(var){
    ....
    return(new_var)
}
```

Einlesen und Schreiben von Dateien

Dateiformat	Funktion	Beschreibung
csv	**read.csv**("file.csv") **write.csv**(df, "file.csv")	CSV Datei
xlsx	**library**(openxlsx) **read.xlsx**("file.xlsx", sheet = 1) **write.xlsx**(df, "file.xlsx")	Excel Datei
Rdata	**load**("file.RData") **save**(df, "file.Rdata")	R spezifische Datei
fst	**library**(fst) **read_fst**("file.fst") **write_fst**(df, "file.fst")	R spezifischer “fast storage” Dateityp (nur für Tabellen als data frame möglich)

SPENGERGASSE
ausbildung mit zukunft

Datentransformationen

mit dplyr und tidyr

dplyr

- Package laden: `library(dplyr)`
- **dplyr** stellt einfache Funktionen für die wichtigsten Aufgaben bei der Datenmanipulation zur Verfügung.
- Voraussetzung: **“tidy data”**
  - jede Variable ist ein einer eigenen Spalte
  - jede Beobachtung (Fall) ist in einer eigenen Zeile
- dplyr Funktionen verwenden „**pipes**“:

  `x %>% func(y)` entspricht `func(x, y)`
- Pipes können verschachtelt sein. Das Ergebnis von einem Schritt dient als Input für den nächsten Schritt. `x %>% func1(y) %>% func2(z)`

dplyr

Zeilen manipulieren

`df %>% filter(Var1 > 1)`	Zeilen basierend auf Variablenwerten auswählen
`df %>% arrange(Var1)` `df %>% arrange(desc(Var1))`	Tabelle sortieren (aufsteigend oder absteigend nach Variablenwerten)
`df %>% distinct()`	doppelte Zeilen entfernen
`df %>% slice(10:15)`	Zeilen mittels Index auswählen

Spalten manipulieren

`df %>% select(Var1, Var2)`	Spalten auswählen
`df %>% mutate(Var3 = Var1 + Var2)`	neue Spalten berechnen
`df %>% rename(Var_new = Var_old)`	Spalte umbenennen
`df %>% pull(Var1)`	Werte einer Spalte als Vektor extrahieren

dplyr

Fälle zusammenfassen

`df %>% summarize(avg =mean(Var1))`	Tabelle aggregieren
`df %>% count(Var1)`	Häufigkeiten einer Variable bestimmen
`df %>% summarize_all(mean)`	Funktion auf alle Spalten anwenden

Fälle gruppieren

`df %>% group_by(Var1) %>% summarize(avg = mean(Var2))`	Zeilen gruppieren und Werte nach Gruppen getrennt auswerten
`grouped_df %>% ungroup()`	Dataframe degruppieren

dplyr

Tabellen kombinieren

left_join()

right_join()

`df1 %>% left_join(df2, by = "Key1")`	**left join** von Dataframes
`df1 %>% right_join(df2, by = "Key1")`	**right join** von Dataframes

dplyr

Tabellen kombinieren

inner_join()

full_join()

`df1 %>% inner_join(df2, by = "Key1")`	**inner join** von Dataframes
`df1 %>% full_join(df2, by = "Key1")`	**outer join** von Dataframes
`df_list = list(name1 = df1, name2 = df2, name3 = df3)` `bind_rows(df_list, .id = "Name")`	Liste von Dataframes zusammenfügen und ID-Spalte mit Listennamen erzeugen

tidyr

- Package laden: `library(tidyr)`
- Tabellen umformatieren:

gather(df, key, value)	“wide”-Format → “long”-Format
spread(df, key, value)	“long”-Format → “wide”-Format

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

→

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

key value

gather(table4a, `1999`, `2000`, key = "year", value = "cases")

table2

country	year	type	count
A	1999	cases	0.7K
A	1999	pop	19M
A	2000	cases	2K
A	2000	pop	20M
B	1999	cases	37K
B	1999	pop	172M
B	2000	cases	80K
B	2000	pop	174M
C	1999	cases	212K
C	1999	pop	1T
C	2000	cases	213K
C	2000	pop	1T

→

country	year	cases	pop
A	1999	0.7K	19M
A	2000	2K	20M
B	1999	37K	172M
B	2000	80K	174M
C	1999	212K	1T
C	2000	213K	1T

key value

spread(table2, type, count)

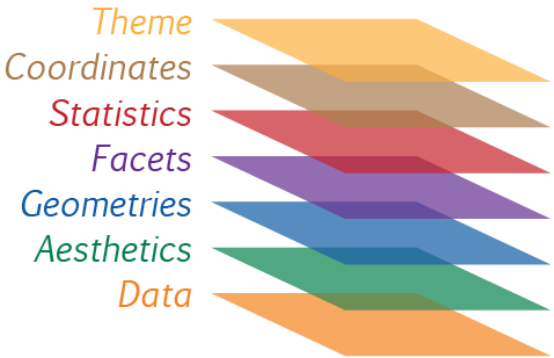
SPENGERGASSE
ausbildung mit zukunft

Datenvisualisierung

mit ggplot2

ggplot2

- Package laden: `library(ggplot2)`
- **ggplot2** baut auf einer Grammatik von Grafikelementen auf.
- Jede Grafik wird schrittweise aus mehreren Schichten von Komponenten gebaut.




```
ggplot(data, aes(...)) + geom_point(...)
                       + facet_wrap(...)
                       + .....
```

ggplot2

- Daten und „Aesthetics“: `ggplot(data, aes(...)) + ...`
- Optionen für aes():

`x = Var1`	Variable zu x-Achse zuweisen
`y = Var2`	Variable zu y-Achse zuweisen
`color = Var3`	Variable zu Linienfarbe zuweisen
`fill = Var4`	Variable zu Füllfarbe zuweisen
`size = Var5`	Variable zu Größe zuweisen
`shape = Var6`	Variable zu Form zuweisen

ggplot2

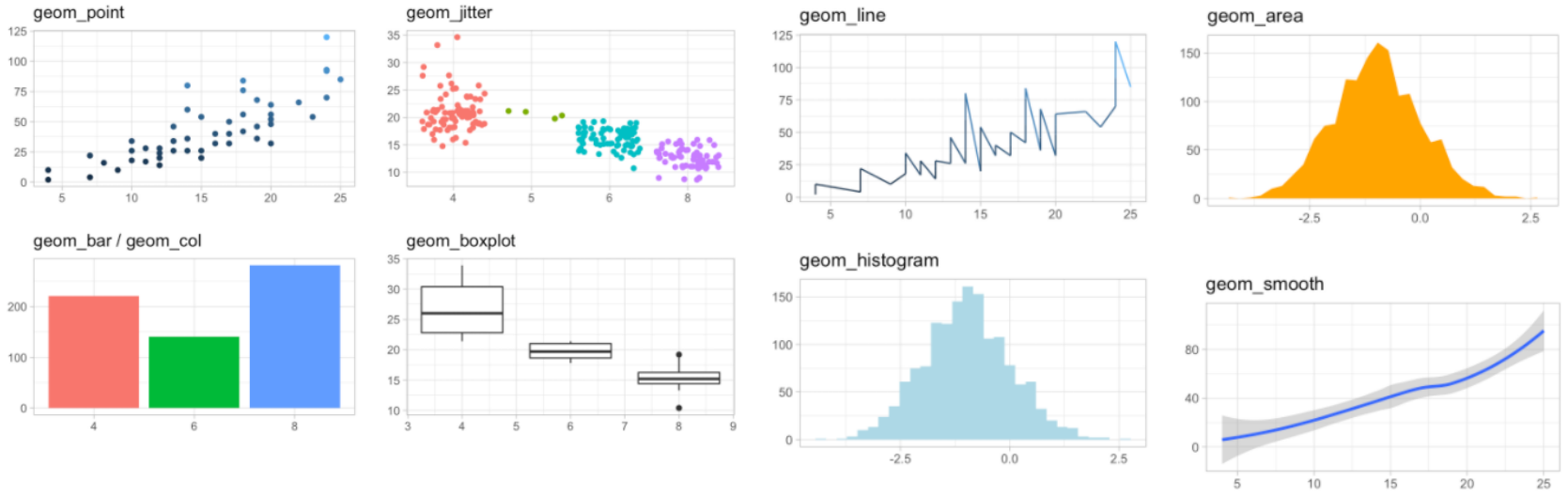
Geometrien

```
ggplot(data, aes(...)) + geom_...
```

+ **geom_point**()	Punktdiagramm
+ **geom_line**()	Liniendiagramm
+ **geom_bar**()	Säulendiagramm aus Rohdaten
+ **geom_col**()	Säulendiagramm aus Häufigkeitswerten
+ **geom_histogram**()	Histogramm
+ **geom_boxplot**()	Boxplot
+ **geom_density**()	Verteilungsdichte
+ **geom_vline**()	vertikale Linie
+ **geom_hline**()	horizontale Linie

ggplot2

Geometrien



ggplot2

Facetten

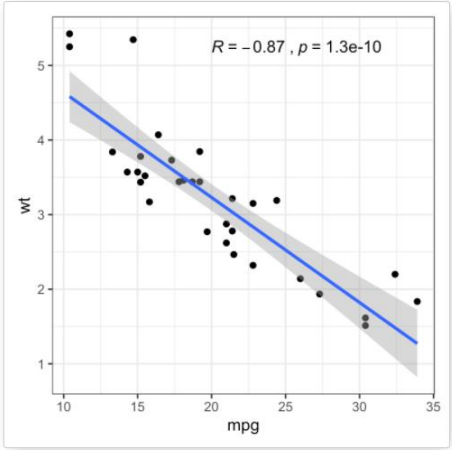
```
ggplot(data, aes(...)) +
    geom_... +
    facet_...
```

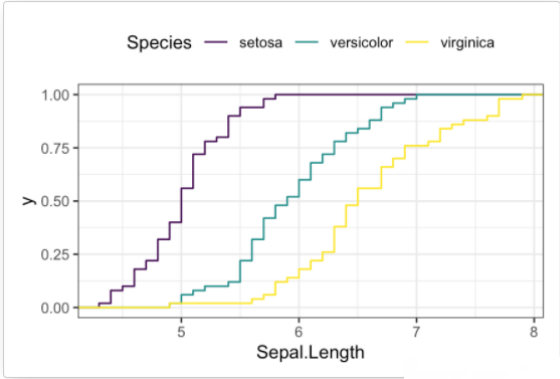
+ **facet_grid(x ~ y)**	Facetten in Zeilen und Spalten
+ **facet_wrap(.~ x)**	Facetten in einem rechteckigen Layout

ggplot2

Statistiken

+ `geom_smooth()`	Kurven glätten
+ `stat_ecdf()`	empirische Verteilungsfunktion
+ `stat_summary()`	Zusammenfassung von y-Werten für verschiedene x-Werte

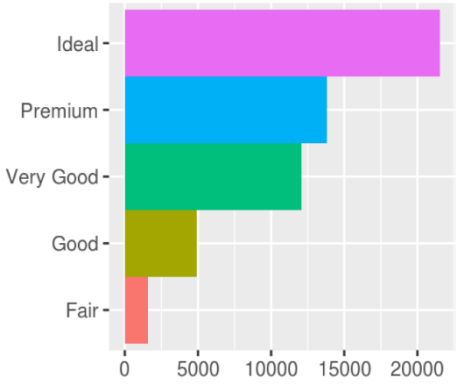


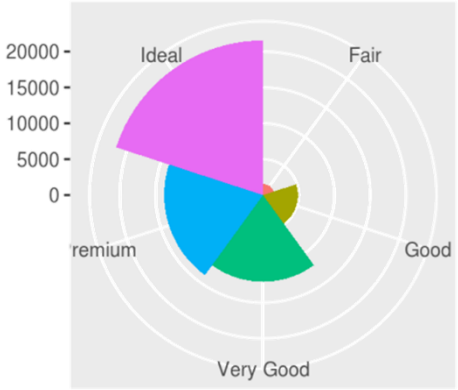


ggplot2

Koordinaten

+ `coord_flip()`	Achsen vertauschen
+ `coord_polar()`	Polarkoordinaten

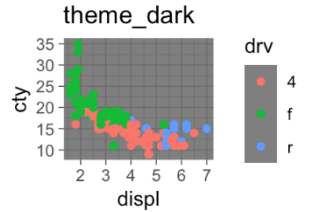


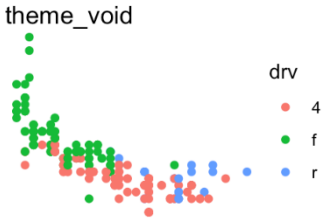


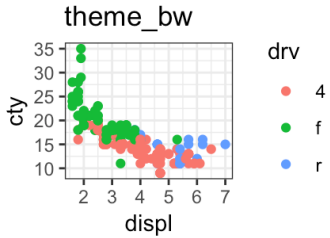
ggplot2

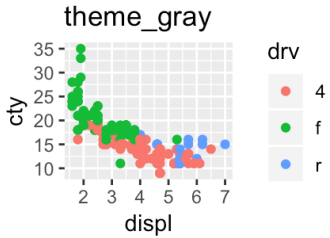
Themen

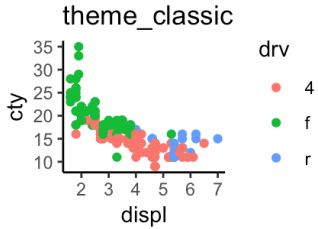
+ **theme_gray()**	grauer Hintergrund (default)
+ **theme_bw()**	schwarz-weiß
+ **theme_minimal()**	minimal
+ **theme_void()**	leer

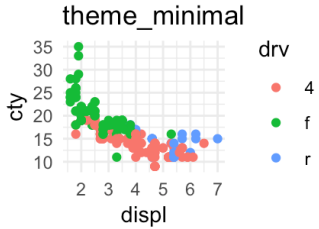












ggplot2

Skalen

`+ scale_color_manual(values = c(...))`	Linienfarben eines Plots verändern
`+ scale_fill_manual(values = c(...))`	Füllfarben eines Plots verändern
`+ scale_x_log10()`	logarithmische Skala auf der x-Achse