Statistische Datenanalyse mit R, Teil 2 online, R-Studio

Dr. Andrea Denecke
Leibniz Universität IT-Services

R Studio

- Ist ein separat zu installierendes Programm, das aber mit dem eigentlichen R-Programm kommuniziert. Basisinstallation ist kostenlos verfügbar unter www.rstudio.com
- 4 Fenster:
  1. Konsole
  2. Datensatz/ Skript
  3. Environment (Übersicht aller Objekte im Workspace)/ History
  4. Viewer/ Packages/ Help/ Plots/ Files
- Anwendungsfreundlicher, übersichtlicher gestaltet, gleiche Funktionalität. Die Eingabe der Befehle und Variablennamen wird unterstützt, über F1 kann direkt die Hilfeseite zur jeweiligen Funktion angefordert werden.

2 Datenmanagement: Teilmengen

Eine Teilmenge des Datensatzes „bundesliga“ mit Hilfe der Funktion subset anlegen:

```
bund_Hamb <- subset(bundesliga,
gastteam=="Hamburg"|heimteam=="Hamburg")
```

Es soll untersucht werden, ob Unterschiede hinsichtlich der Ergebnisse bei Heim- & Auswärtsspielen bestehen (**Hypothese: “Zuhause spielt Hamburg besser als auswärts“**).

1. „**Zuhause“** definieren: neue Variable erstellen, die angibt, ob es sich um ein Heim- oder Auswärtsspiel handelt.
2. **Spielergebnis** klar darstellen (gewonnen, unentschieden, verloren)
3. **Statistischen Test** anwenden

Datenmanagement: Rekodieren

1. *Neue Variable erstellen, die angibt, ob es sich um ein Heim- oder Auswärtsspiel handelt.*

Neue Variable „spielort“ über Rekodieren der Variable „heimteam“:

```
bund_Hamb$spielort <-
recode(bund_Hamb$heimteam,'"Hamburg"=1; else=2‘,
as.factor=T)
```

Funktion recode Paket {car}

In der Variable „spielort“ bedeutet 1=heim und 2=auswärts.

Alternativ kann man auch eine Variable in eine andere numerische Variable über eine einfache TRUE/ FALSE Abfrage rekodieren, hierzu später (Folie 8).

Datenmanagement

2. *Spielergebnis klar darstellen (gewonnen, unentschieden, verloren)*

Ziel ist, eine Variable zu erhalten, die Auskunft darüber gibt, ob das jeweilige Spiel aus Hamburger Sicht gewonnen, verloren oder unentschieden ausgegangen ist.

Ausgangspunkt in unserem Datensatz sind die Tore.

Die von Hamburg erzielten Tore sind teils unter der Variable „heimtore“, teils unter der Variable „gasttore“ zu finden, je nach Spielort. Wir sortieren dieses daher vorab über eine weitere Variable „tore_Hamb“.

5 Datenmanagement: logischer Test

Man kann „tore_Hamb“ über die Funktion `ifelse()` erzeugen. Diese braucht drei Argumente:

1. einen logischen Test
2. den Wert, der bei Zustimmung ausgegeben wird
3. den Wert, der bei Verneinung ausgegeben wird

```
bund_Hamb$tore_Hamb <- ifelse(
test = bund_Hamb$spielort == "1",
yes=bund_Hamb$heimtore,
no=bund_Hamb$gasttore)
```

Übung 2

- Erzeugen Sie den Teildatensatz bund_Hamb (Folie 2), darin die Variable „spielort“ (Folie 3) und die Variable „tore_Hamb“ (Folie 5)

1. Berechnen Sie in der gleichen Weise die Variable „tore_gegn“ (Folie 5).
2. Berechnen Sie nun eine Variable „tore_diff“ (Tordifferenz, aus Sicht von Hamburg) aus den Variablen „tore_Hamb“ und „tore_gegn“ (Teil 1- Folie 5)
3. Berechnen Sie die jeweiligen Mittelwerte für die beiden Variablen „tore_Hamb“ und „tore_gegn“ (Teil 1- Folie 10)
4. Geben Sie den Befehl `str(bund_Hamb)` ein.

Exkurs Datentypen

logical	logische Werte	TRUE
character	Zeichenfolge	„Schalke“
numeric	ganze und reelle Zahlen	3.78
integer	nur ganze Zahlen	30

Anzeigen des **Datentyps**: `mode(bund_Hamb$tore_diff)`

Klassifizierung des Datentyps:

- „factor“-Variablen dienen zur Einteilung in Klassen/ Gruppen. Die numerische Variable „spielort“ (1=heim, 2=auswärts) wird als „factor“ bewertet, d.h. sie dient als Gruppierungsvariable. Versuchen Sie die Funktion `mean(bund_Hamb$spielort)` anzuwenden!

-„Numeric“- oder „integer“-Variablen (auch „logical“) dienen zur Berechnung.

Abfragen der Struktur/ Klasse des Datentyps:
`class(bund_Hamb$spielort)` oder `str(bund_Hamb)`

Ändern der Struktur des Datentyps: s. nächste Folie

Exkurs Datentypen

Andere Klassifizierung zuweisen:

`bund_Hamb$rundeF <- as.factor(bund_Hamb$runde)` bzw. `as.integer, as.numeric…`

Eine „logical“- Variable kann man sich erzeugen über
`bund_Hamb$logical <- bund_Hamb$tore_diff ==0`

Es wird die Abfrage durch geführt, ob die Tordifferenz gleich Null ist; wenn ja, dann wird ein TRUE eingesetzt, wenn nein, ein FALSE. Intern wird TRUE=1 und FALSE=0 gesetzt, man kann also auch mit Wahrheitswerten rechnen, z.B. `sum(bund_Hamb$logical)`.

Die Rekodierung der Variable spielort (Folie 3) wäre so gegangen:

```
bund_Hamb$spielortN <- (bund_Hamb$heimteam
=="Hamburg")*1 + (bund_Hamb$heimteam !="Hamburg")*2
```

Datenmanagement: Löschen von Daten

Löschen von Variablen/ Fällen/Datensätzen

Variablen: `dataset$varname <- NULL`

Fälle: `dataset <- dataset[-c(1,2,3),]` oder
`dataset <- dataset[-c(1:3),]`
löscht die ersten drei Zeilen des Datensatzes

Datensatz: aus dem Workspace `rm(Objektname)` „**rem**ove“
Löschen mehrerer Objekte mit `rm(test,Daten,Name)`
Löschen des kompletten Workspace mit `rm(list=ls())` oder
über das Menü: **Session → Clear Workspace**
oder mit dem Besen- Icon beim Global Environment.

Datenmanagement: Fehlende Werte

Fehlende Werte sollten immer die Bezeichnung NA haben.

Treten andere Kodierungen auf (z.B. -1, 999,…), sollte man diese umcodieren, z.B.: `recode(datensatz,'-1=NA;999=NA')`

Zählen fehlender Werte in einem Datensatz:
`sum(is.na(dataset))`

An welcher **Stelle im Vektor** treten die Werte auf?
`which(is.na(vector))`

Berechnungen bei Vektoren mit fehlenden Werten:

`mean(na.omit(vector))`

Die Funktion `na.omit()` bereinigt den Vektor um die fehlenden Werte, da man sonst als Ergebnis auch ein NA erhalten würde.
Alternativ funktioniert das Argument `na.rm=TRUE`
`mean(vector, na.rm=TRUE)`

Grafiken

Der generelle Grafik-Befehl lautet `plot()`, unter Angabe diverser grafischer Parameter (s. R Reference Card https://cran.r-project.org/doc/contrib/Short-refcard.pdf oder `?plot`) können hiermit Linien-, Punkt- und weitere Diagramme erstellt werden.

Beispiel: `plot(bund_Hamb$runde, bund_Hamb$tore_diff, type=„p“)`

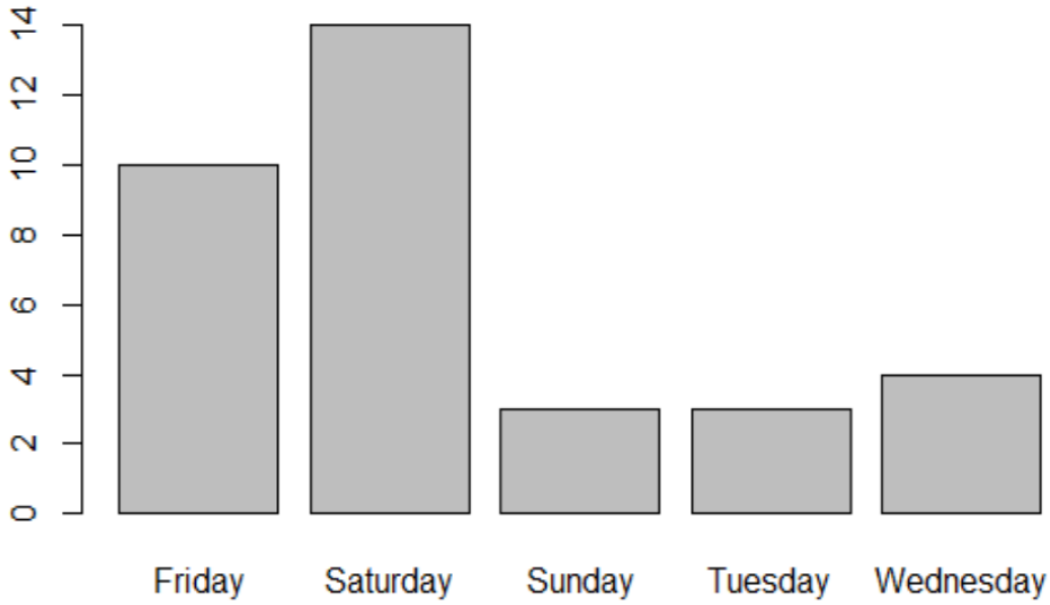
`abline(h=0)` erzeugt eine Bezugslinie auf Höhe 0

Funktion `barplot()`: zuerst Tabelle anlegen, hier von den Wochentagen, an denen gespielt wurde: `tbH <- table (bund_Hamb$wochtag)`, um dann diese Tabelle im Balkendiagramm zu verwenden: `barplot(tbH)`

Grafiken: Grafische Parameter

An dieser Grafik können diverse Änderungen vorgenommen werden, s. „***Graphical Parameters***“:

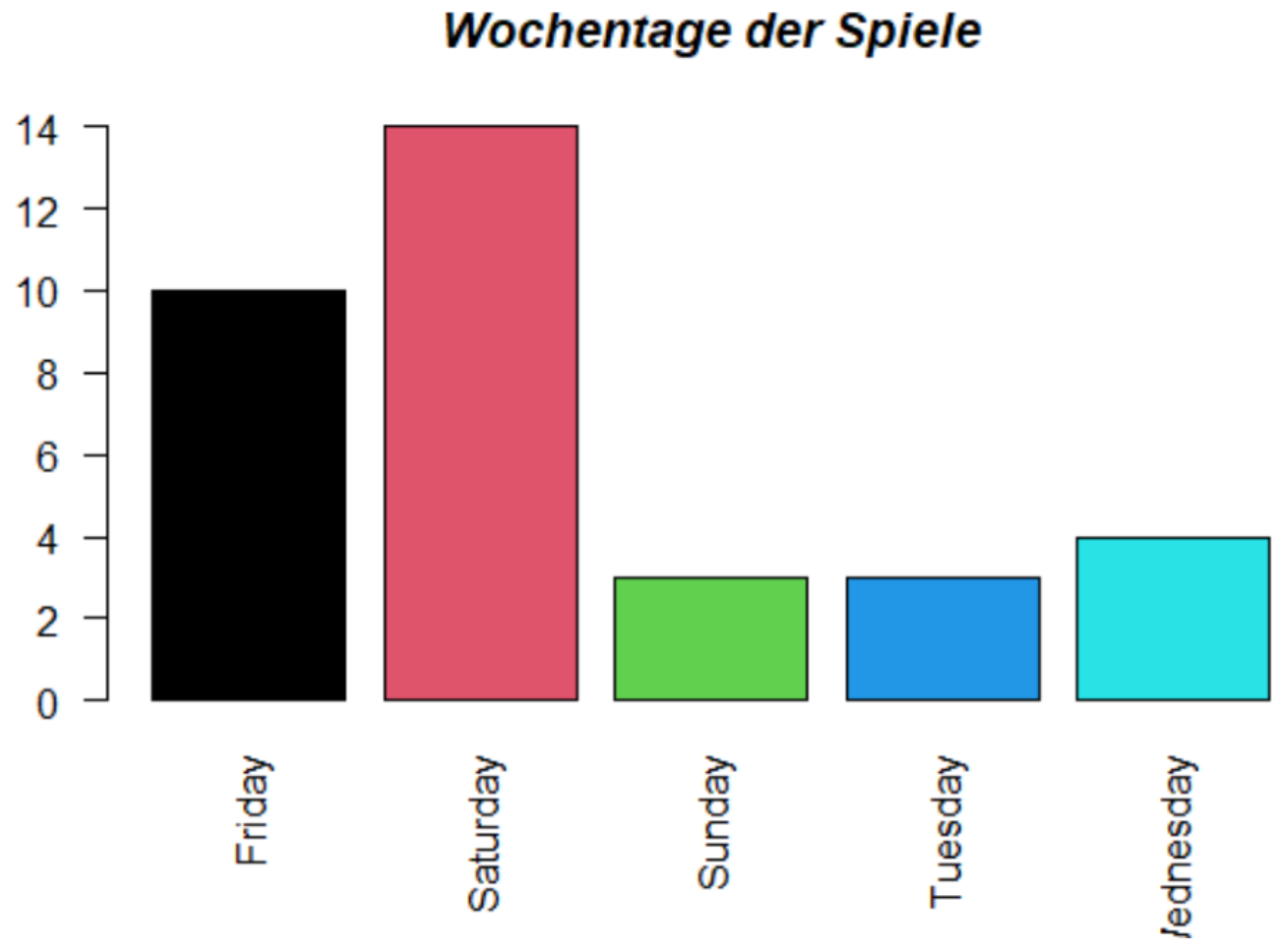
- Farbe der Balken über ***col***
- Titel über ***main***
- Schriftart des Titels über ***font.main***
- Ausrichtung der Achsenbeschriftung über ***las***
- …
  Beispiel s. nächste Seite



`barplot(tbH)`

Grafiken: Grafische Parameter

```
barplot(tbH, col=c(1:5), las=2,
main="Wochentage der Spiele", font.main=4)
```



14

Grafiken: Farben, Fenster teilen

`colors()` zeigt alle verfügbaren Farben für die Grafiken, es kann entweder der Name der Farbe eingetragen werden (`col=„rosybrown“`) oder eine Farbzahl. Diese entspricht aber nicht der Aufzählung über `colors()`, sondern der von `palette()`.

Möchte man zwei Grafiken in einem Fenster haben, geht dies über den Befehl `par(mfrow=c(1,2))`
Die beiden danach erzeugten Grafiken werden nebeneinander in das Grafikfenster gelegt, dementsprechend erscheinen die Grafiken bei `par(mfrow=c(2,1))` untereinander, bei …`c(2,2)` erhält man vier Teile usw…
Zurück setzen geht in dieser Weise über …`c(1,1))`

Übung 3

1. Erzeugen Sie in dem Datensatz „bund_Hamb“ eine Variable „ergebnis“, die angibt, ob es sich um einen Sieg von Hamburg, eine Niederlage oder ein Unentschieden handelt (Folie 3/ 5)
2. Wie oft hat Hamburg gewonnen, wie oft verloren und wie oft unentschieden gespielt? Wie ist die Verteilung in Prozent? (Folie 11 & Teil 1-Folie 12)
3. Erzeugen Sie ein Balkendiagramm für die Variable „ergebnis“. Verändern Sie die Farbe der Balken. Fügen Sie einen Titel „Spielergebnisse des Hamburger SV“ hinzu.
   Probieren Sie ggf. noch andere low-level plotting commands aus! (Folien 11-13)
4. Speichern Sie dieses Diagramm als .jpeg (selbst probieren)

Statistisches Testen

Nun soll schlussendlich noch die Hypothese überprüft werden, ob Hamburg zuhause besser gespielt hat als auswärts. Dies kann anhand der Variable „ergebnis“ geschehen.
Zuerst grafisch darstellen: für das Balkendiagramm („barplot“) eine Tabelle erstellen, die die Ergebnisse nach Spielort anzeigt:

```
tb_ergebnis <- table(bund_Hamb$spielort, bund_Hamb$ergebnis)
```

Hiermit erzeugt man das Balkendiagramm:

```
barplot(tb_ergebnis)
```

Einige „low-level-plotting-commands“ verschönern das Ganze:

```
barplot(tb_ergebnis, beside=T, legend.text=c(„heim“, „auswärts“)
```

Statistisches Testen

Nun möchten wir prüfen, ob signifikante Unterschiede beim „ergebnis“ hinsichtlich des „spielort“ vorliegen. Allerdings kann in einem statistischen Test die „character“-Variable „ergebnis“ nicht verarbeitet werden*, wir müssen sie daher erst in eine numerische Variable umwandeln:

Entweder über die „recode“-Funktion:

```
bund_Hamb$ergebnisN <-
recode(bund_Hamb$ergebnis,'"Sieg"=1;"Niederlage" =-1;
„unentschieden"=0')
```

oder über eine verschachtelte „ifelse“-Funktion:

```
bund_Hamb$ergebnis1 <-
ifelse(test = bund_Hamb$ergebnis == "Sieg",
   yes=1,
   no=ifelse(test=bund_Hamb$ergebnis==„unentschieden",
             yes=0,
             no=-1)
```

*s. nächste Folie

Statistisches Testen

Für die nun ordinale Variable „ergebnisN“ bietet sich ein nicht-parametrischer Test an, z.B. der Wilcoxon-Rangsummen-Test. Fragestellung ist einseitig („daheim häufiger gewonnen als auswärts“; zweiseitig hieße das: „bestehen Unterschiede in den Ergebnissen hinsichtlich des Spielortes“).

```
wilcox.test(bund_Hamb$ergebnisN ~ bund_Hamb$spielort, alternative=„greater“)
```

Das Ergebnis ist p= 0.01869, d.h. p<0.05, also kann man mit 5%iger Irrtumswahrscheinlichkeit sagen, dass in dieser Saison Hamburg zuhause signifikant bessere Spielergebnisse erzielt hat als auswärts.

*Prinzipiell hätte man auf die „character“-Variable auch schon den Chi-Quadrat Test (`chisq.test`) anwenden können.

Exkurs

Die zuvor erzeugte Grafik (Folien 11-13) mit den Wochentagen ändern:

Hinzufügen der fehlenden Wochentage: innerhalb der Tabelle über `fix(tbh)` oder

```
tbh[„Monday“]<-0; tbh[„Thursday“] <- 0
```

Sortieren nach Häufigkeit:

`sort(tbh)`, andersherum:

```
sort(tbh, decreasing=TRUE)
```

Grafiken für Fortgeschrittene: Paket {ggplot2}

```
ggplot(bund_Hamb, aes(runde, tore_diff, colour=
spielort==1))+geom_point()
```

`+geom_density_2d()+geom_boxplot()` (Sinnfrei, aber toll)