Wir lernen Lagemaße und ggplot

Schmidt, Sophie C.

14 Mai, 2019

Im ersten Code chunk werden ein paar Dinge festgelegt, die uns im Moment nicht beschäftigen müssen.

## Pakete

Als allererstes laden wir die Pakete, die wir benötigen (der library-Befehl).

Falls sie noch nicht installiert sind, muss nur die Raute / der Hashtag (dieses Symbol: #) vor dem Befehl zur Paketinstallation gelöscht werden und der Befehl mit markieren und Strg+Enter ausgeführt werden. Danach ist das Paket auf dem System installiert und müsste (zumindest theoretisch) nie wieder installiert werden. Deswegen kann man den Befehl nach erfolgreicher Installation sofort wieder “auskommentieren” (den hashtag davor setzen). Auskommentierter Code innerhalb eines Code chunks wird nicht ausgeführt, wenn man das komplette Rmarkdown-Dokument ausführt bzw “knitted” (in ein Word- oder PDF-Dokument oder eine html-Datei umwandelt)

[Mit Strg+Alt+i kann man einen Codechunk eröffnen. Oder einfach eintippen:]

#install.packages("ggplot2")  
library(ggplot2) # Paket aktivieren  
#install.packages("archdata")  
library(archdata) #Paket aktivieren  
#install.packages("xlsx")  
#library(xlsx)

## Daten laden

data("BACups") # das geht, weil das Paket archdata geladen wurde und die Daten dort vorliegen  
  
View(BACups) #so kann ich mir die Tabelle angucken. Ich kann sie auch rechts unter dem Environments-Tab und "Data" anklicken  
  
  
# habe ich die Daten als csv vorliegen, gilt diese Funktion  
#mydata <- read.csv2("./data/BACups.csv") # Standard ist, das der Seperator ein Komma ist  
#mydata <- read.csv2("Pfad/datei.csv", sep = "\t") # mit "sep = " lässt sich ein anderer Seperator festlegen  
# mydata ist ein Synonym für eine sinnigere Bezeichnung wie zB BACups  
  
# habe ich Excel-Daten gilt dies:  
#library(xlsx)  
#mydata <- read.xlsx("c:/myexcel.xlsx", 1) #,1 bedeutet das erste sheet in der excel-Datei  
  
# read in the worksheet named mysheet  
#mydata <- read.xlsx("c:/myexcel.xlsx", sheetName = "mysheet")  
  
# Sollte jemand mal Daten speichern wollen, ist das der Code:  
  
#write.csv(BACups, "../data/BACups.csv")#  
#write.xlsx(BACups, "../data/BACups.xlsx")

## Lagemaße

Lagemaße sind einfache Berechnungen von Werten, die uns etwas über die Verteilung der Werte aussagen soll. Am bekanntesten ist der Mittelwert, wir berechnen jedoch auch Modus, Median, Standardabweichung und Varianz.

* Der Mittelwert berechnet sich in dem man alle Werte zusammenrechnet und durch die Anzahl der Werte teilt.
* Der Median ist der Wert, der, wenn ich meine Wertereihe nach Größe ordne, den Datensatz in genau zwei Hälften teilt.
* Der Modus ist der Wert, der am häufigsten vorkommt.
* Die Standardabweichung sagt, wie wie stark die Streuung der Werte um den Mittelwert ist. Sie ist die Wurzel aus der Varianz.
* Die Varianz ist die “mittlere quadratische Abweichung der Werte um den Mittelwert”. Sie wird quadriert zur Berechnung, damit die Plus- und Minusabweichungen um den Mittelwert sich nicht aufheben. Dadurch ist die Zahl aber immer sehr groß. Die Standardabweichung ist für uns einfacher zu verstehen, weil sie diese Quadrierung wieder aufhebt.

Mit einem “<-” wird ein Wert, den ich berechne einer Variablen zugewiesen, mit der ich später weiterrechnen kann. Das ist häufig eine gute Idee. Das Dollar-Zeichen bezeichnet den vector (“die Spalte”) eines data frames (“der Tabelle”). Also “BACups$RD” ist die Spalte RD in der Tabelle BACups.

RD\_mean <- mean(BACups$RD) # Hiermit wird der Mittelwert berechnet. (alle Werte zusammengerechnet / Anzahl der Werte)  
RD\_med <- median(BACups$RD) #das ist die Berechnung des Medians.   
table(BACups$RD) #hiermit kann man sich anschauen, wie häufig jeder Wert vorkommt (was der Modus ist)  
#>   
#> 6.6 8 8.4 8.5 8.7 8.8 9 9.5 9.8 10 10.1 10.5 10.7 10.8 11   
#> 1 2 1 1 1 1 5 3 1 1 2 2 1 1 3   
#> 11.1 11.5 11.7 12 12.1 12.9 13 13.2 13.3 15 15.2 15.5 15.8 17.2 18   
#> 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1   
#> 18.3 18.5 19 19.1 19.5 20 20.8 22 24 29 29.5   
#> 1 3 4 1 3 1 1 1 1 1 1  
RD\_sd <- sd(BACups$RD) #Standardabweichung  
RD\_var <- var(BACups$RD) # Varianz  
RD\_range <- range(BACups$RD) # kleinster und größter Wert  
  
## Aufgabe: Andere Variablen ebenso erkunden. Unklar welche? Schaut euch die Tabelle BACups mit View() an und entscheidet, welche sich eignen würden.

Euch ist vielleicht aufgefallen, dass es für den Modus keine Funktion in R zu geben scheint. Keine Ahnung warum. Aber man findet online schlaue Menschen, die eine Funktion geschrieben haben, mit der man ganz genauso den Modus abfragen kann, wie man den Median abfragen kann. Diese Funktion kann man einmal markieren und mit Strg+Enter ausführen und bekommt dann unter Environment und Functions angezeigt, dass man eine Funktion “getmode” erstellt hat. JETZT kann man sie anwenden, wie unten im Bsp.

# Funktion schreiben  
getmode <- function(v) { # die Funktion heißt getmode und wird auf einen Vektor v angewendet  
 uniqv <- unique(v) # die einzelnen Werte des Vektors (ohne Dopplungen) werden in einem Vektor uniqv gesammelt   
 uniqv[which.max(tabulate(match(v, uniqv)))] # welcher maximale Wert entsteht, wenn ich zähle, wie häufig die Werte von uniqv in v vorkommen (das ist der Modus)  
}  
  
mod\_RD <- getmode(BACups$RD) # Modus

Überprüft doch einmal, ob ihr mit der getmode genau den gleichen Modus bekommt, wie ihr selber mit table gefunden habt! :-)

## ggplot

ggplot wurde von Hadley Wickham entwickelt, ist ein Paket mit vielen Funktionen zur Visualisierung von Daten und folgt einer “Grammatik der Diagramme”.

Erarbeiten wir uns das Schritt für Schritt.

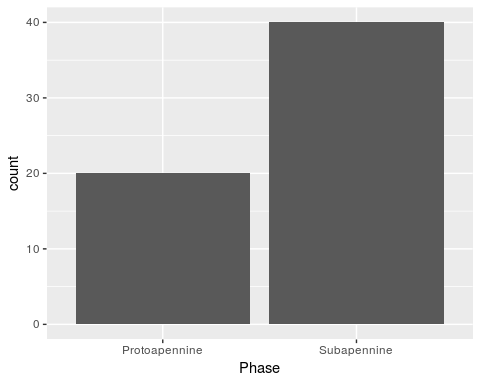
Wir müssen dem Programm sagen: Welche Daten es benutzen soll (data = ), welche Art von Diagramm es bauen soll (geom\_xxx) und wie das Diagramm aussehen soll (aes von aesthetics), damit überhaupt etwas entsteht.

Alles andere danach sind reine Verschönerungsmaßnahmen. ;-)

# ein Säulendiagramm

Ein Säulendiagramm eignet sich zur Darstellung nominaler und ordinaler Variablen. Ihr könnt es ja mal mit metrischen Probieren, dann seht ihr schnell, warum das nicht gut ist.

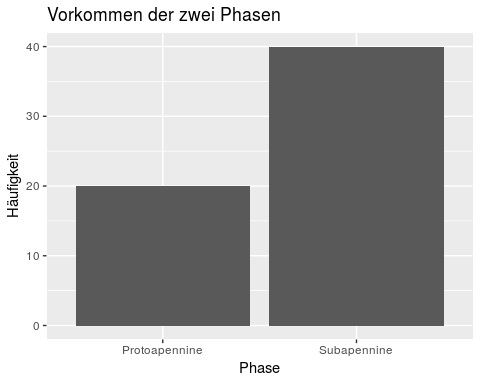
ggplot(data = BACups)+ #data = bezieht sich auf den Datensatz mit dem ich arbeite, meist ein dataframe  
 geom\_bar(aes(x = Phase)) #geom\_bar bedeutet, ich hätte gern ein Balkendiagramm, aesthetic: ich will, dass auf der X-Achse die Phasen abgetragen werden



Jetzt kann man viele Dinge verschönern

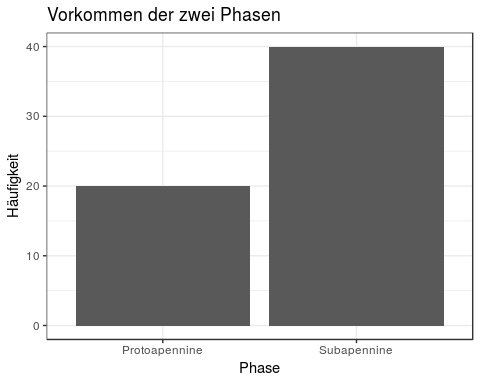
1. B. Den Achsen eine andere Beschriftung geben:

ggplot(data = BACups)+   
 geom\_bar(aes(x = Phase))+  
 labs(y = "Häufigkeit", #der y-Achse einen neuen Namen geben  
 title = "Vorkommen der zwei Phasen") # dem ganzen Plot eine Überschrift geben



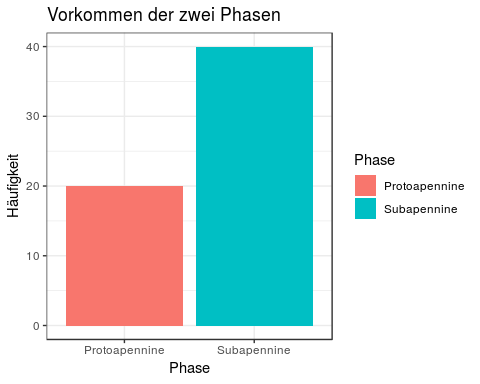
Oder einen anderen Look wählen (ein anderes Thema):

ggplot(data = BACups)+   
 geom\_bar(aes(x = Phase))+   
 labs(y = "Häufigkeit",  
 title = "Vorkommen der zwei Phasen")+  
 theme\_bw() #theme\_classic, theme\_grey, theme\_minimal



Oder die Säulen bunt einfärben:

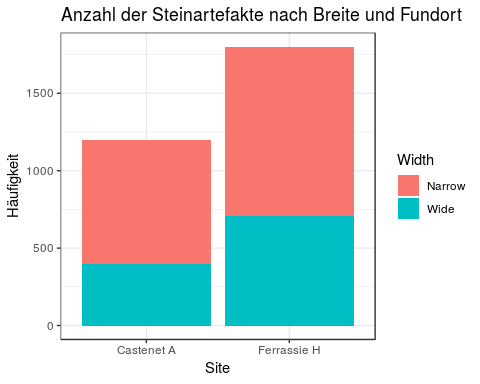
ggplot(data = BACups)+ #data = bezieht sich auf den Datensatz mit dem ich arbeite, meist ein dataframe  
 geom\_bar(aes(x = Phase, fill = Phase))+ # fill gibt den Balken unterschiedliche Farben, je nach den Angaben in der Spalte Phase  
 labs(y = "Häufigkeit",  
 title = "Vorkommen der zwei Phasen")+  
 theme\_bw()



EXTRA Aufgabe:

Überlegt bitte, was in dem nächsten Code Chunk passiert. Die Hilfe kann mit ?Suchbegriff abgerufen werden.

data("EndScrapers")  
ggplot(data = EndScrapers)+  
 geom\_col(aes(x = Site, fill = Width, y = Freq))+   
 labs(y = "Häufigkeit",  
 title = "Anzahl der Steinartefakte nach Breite und Fundort")+  
 theme\_bw()



##### pagebreak

# References

##### pagebreak

### Colophon

This report was generated on 2019-05-14 11:16:24 using the following computational environment and dependencies:

# which R packages and versions?  
devtools::session\_info()  
#> ─ Session info ──────────────────────────────────────────────────────────  
#> setting value   
#> version R version 3.4.4 (2018-03-15)  
#> os Ubuntu 16.04.6 LTS   
#> system x86\_64, linux-gnu   
#> ui X11   
#> language de\_DE   
#> collate de\_DE.UTF-8   
#> ctype de\_DE.UTF-8   
#> tz Europe/Berlin   
#> date 2019-05-14   
#>   
#> ─ Packages ──────────────────────────────────────────────────────────────  
#> package \* version date lib source   
#> archdata \* 1.2 2018-02-19 [1] CRAN (R 3.4.4)  
#> assertthat 0.2.1 2019-03-21 [1] CRAN (R 3.4.4)  
#> backports 1.1.3 2018-12-14 [1] CRAN (R 3.4.4)  
#> bookdown 0.9 2018-12-21 [1] CRAN (R 3.4.4)  
#> callr 3.2.0 2019-03-15 [1] CRAN (R 3.4.4)  
#> cli 1.1.0 2019-03-19 [1] CRAN (R 3.4.4)  
#> colorspace 1.2-4 2013-09-30 [3] CRAN (R 3.1.0)  
#> crayon 1.3.4 2017-09-16 [1] CRAN (R 3.4.4)  
#> desc 1.2.0 2018-05-01 [1] CRAN (R 3.4.4)  
#> devtools 2.0.2 2019-04-08 [1] CRAN (R 3.4.4)  
#> digest 0.6.9 2016-01-08 [3] CRAN (R 3.2.3)  
#> evaluate 0.13 2019-02-12 [1] CRAN (R 3.4.4)  
#> fs 1.2.7 2019-03-19 [1] CRAN (R 3.4.4)  
#> ggplot2 \* 3.1.1 2019-04-07 [1] CRAN (R 3.4.4)  
#> glue 1.3.1 2019-03-12 [1] CRAN (R 3.4.4)  
#> gtable 0.1.2 2012-12-05 [3] CRAN (R 3.0.0)  
#> htmltools 0.3.6 2017-04-28 [1] CRAN (R 3.4.4)  
#> knitr 1.22 2019-03-08 [1] CRAN (R 3.4.4)  
#> labeling 0.3 2014-08-23 [3] CRAN (R 3.1.1)  
#> lazyeval 0.2.2 2019-03-15 [1] CRAN (R 3.4.4)  
#> magrittr 1.5 2014-11-22 [1] CRAN (R 3.4.4)  
#> memoise 1.1.0 2017-04-21 [1] CRAN (R 3.4.4)  
#> munsell 0.5.0 2018-06-12 [1] CRAN (R 3.4.4)  
#> pillar 1.3.1 2018-12-15 [1] CRAN (R 3.4.4)  
#> pkgbuild 1.0.3 2019-03-20 [1] CRAN (R 3.4.4)  
#> pkgconfig 2.0.2 2018-08-16 [1] CRAN (R 3.4.4)  
#> pkgload 1.0.2 2018-10-29 [1] CRAN (R 3.4.4)  
#> plyr 1.8.3 2015-06-12 [3] CRAN (R 3.2.1)  
#> prettyunits 1.0.2 2015-07-13 [1] CRAN (R 3.4.4)  
#> processx 3.3.0 2019-03-10 [1] CRAN (R 3.4.4)  
#> ps 1.3.0 2018-12-21 [1] CRAN (R 3.4.4)  
#> R6 2.4.0 2019-02-14 [1] CRAN (R 3.4.4)  
#> Rcpp 0.12.3 2016-01-10 [3] CRAN (R 3.2.3)  
#> remotes 2.0.2 2018-10-30 [1] CRAN (R 3.4.4)  
#> rlang 0.3.3 2019-03-29 [1] CRAN (R 3.4.4)  
#> rmarkdown 1.12 2019-03-14 [1] CRAN (R 3.4.4)  
#> rprojroot 1.3-2 2018-01-03 [1] CRAN (R 3.4.4)  
#> scales 1.0.0 2018-08-09 [1] CRAN (R 3.4.4)  
#> sessioninfo 1.1.1 2018-11-05 [1] CRAN (R 3.4.4)  
#> stringi 1.4.3 2019-03-12 [1] CRAN (R 3.4.4)  
#> stringr 1.4.0 2019-02-10 [1] CRAN (R 3.4.4)  
#> testthat 0.11.0 2015-10-14 [3] CRAN (R 3.2.2)  
#> tibble 2.1.1 2019-03-16 [1] CRAN (R 3.4.4)  
#> usethis 1.4.0 2018-08-14 [1] CRAN (R 3.4.4)  
#> withr 2.1.2 2018-03-15 [1] CRAN (R 3.4.4)  
#> xfun 0.6 2019-04-02 [1] CRAN (R 3.4.4)  
#> yaml 2.2.0 2018-07-25 [1] CRAN (R 3.4.4)  
#>   
#> [1] /home/sophie/R/x86\_64-pc-linux-gnu-library/3.4  
#> [2] /usr/local/lib/R/site-library  
#> [3] /usr/lib/R/site-library  
#> [4] /usr/lib/R/library