Wir ggploten weiter

Sophie C. Schmidt

10 Mai 2019

## Metrische Daten

In der letzten Stunde haben wir vor allem mit nominalen Daten gearbeitet und sie dargestellt. Dafür eignen sich Balkendiagramme ganz hervorragend. Heute geht es aber um metrische Daten.

Deshalb lernen wir ein paar neue Diagramme kennen.

* Histogramme
* Streudiagramme
* Boxplot-Diagramme

Aber zuerst die Daten laden:

#install.packages("ggplot2")  
library(archdata)  
data("BACups")  
data("Snodgrass")  
library(ggplot2)

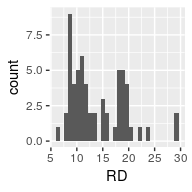
# Histogramm

Fangen wir an mit Histogrammen. Histogramme sind sehr beliebt für metrische Daten, weil man relativ einfach die Verteilung der Werte erkennen kann. Ein Histogramm sieht manchmal aus wie ein Balkendiagramm, ist es aber nicht! Histogramme haben auf der x-Achse eine klassifizierte metrische Variable und auf der y-Achse entweder die Häufigkeit dieses Wertes oder die Dichte.

Ein Beispiel:

ggplot(data = BACups)+  
 geom\_histogram(aes(x = RD))

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.

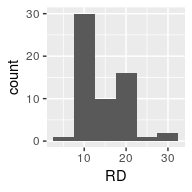


Wenn ihr das ausgeführt habt, sollte eine Meldung aufgeploppt sein: “stat\_bin() using bins = 30. Pick better value with binwidth.”

“Binwidth” bezeichnet die Größe der Klasse. “bins = 30” heißt, dass der gesamte Datensatz in 30 gleich große Klassen unterteilt wurde.

Probiert einmal verschiedene Werte in dem Beispiel aus:

ggplot(data = BACups)+  
 geom\_histogram(aes(x = RD), binwidth = 5) # Klassengröße 5. Versucht doch einmal andere Werte und schaut, was passiert!



Was lernen wir daraus? Die Wahl der Klassengröße macht eine Menge aus, wie ich die Daten wahrnehme und welche Aussagen ich über sie treffen werde.

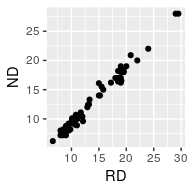
Was gibt es noch für Möglichkeiten der Datenvisualisierung?

# Streudiagramme

Bei Streudiagrammen kann ich zwei Variablen einer Einheit gegeneinander plotten.

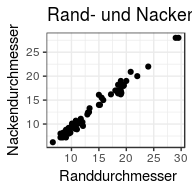
Wir tragen auf der X- und auf der Y-Achse metrische Daten ab. Das gehört zu den aesthetics-Elementen, deshalb tun wir die Info in die Klammern hinter aes():

ggplot(data = BACups)+  
 geom\_point(aes(x = RD, y = ND))



Jetzt können wir damit wieder die Dinge tun, die wir mit dem Balkendiagramm gemacht hatten, also die Achsen beschriften, einen Titel vergeben und den Style ändern:

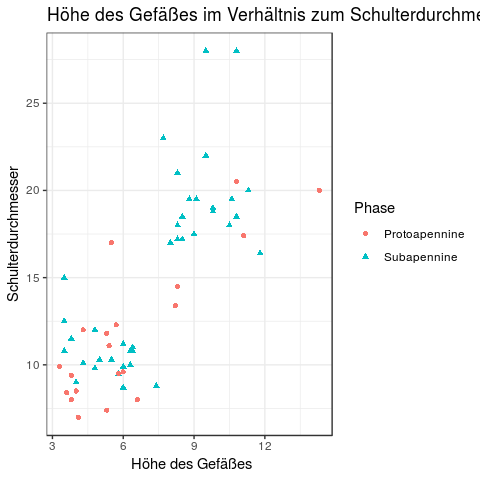
ggplot(data = BACups)+  
 geom\_point(aes(x = RD, y = ND)) +   
 labs(x =" Randdurchmesser",  
 y ="Nackendurchmesser",  
 title = "Rand- und Nackendurchmesserim Verhältnis zueinander")+  
 theme\_bw()



Was kann man noch tolles machen? Die Form der Punkte von einer Variablen bestimmen lassen! Und die Farbe!

Welches Merkmal, das ich in der Tabelle als Spalte aufgenommen habe die Form der Punkte bestimmt lege ich mit “shape” fest, die Farbe mit “color”.

ggplot(data = BACups)+  
 geom\_point(aes(x = H, y = SD, shape = Phase, color = Phase)) +   
 labs(x =" Höhe des Gefäßes",  
 y ="Schulterdurchmesser",  
 title = "Höhe des Gefäßes im Verhältnis zum Schulterdurchmesser")+  
 theme\_bw()

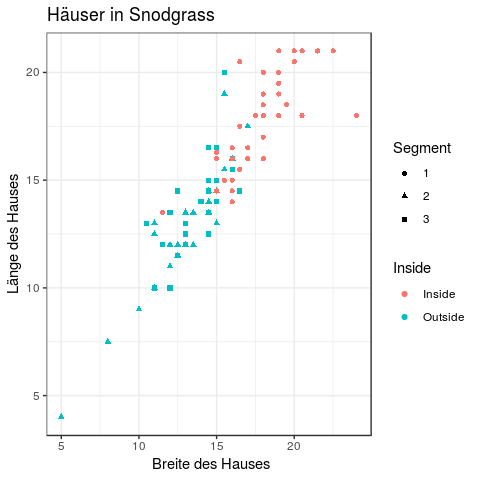


Oooooh, schaut euch mal das Ergebnis an! Da könnte man schon fast was interpretieren!

Probiert doch einmal noch 2-3 andere Parameter aus, ob die vielleicht auch einen Unterschied zwischen den zwei Phasen erkennen lassen?

Form und Farbe kann man natürlich auch von unterschiedlichen Parametern bestimmen lassen. Da diese Eigenschaften jedoch nominaler Art sein müssen und wir keinen zweiten nominale Variable in dem BACups-Datensatz haben, benutzen wir doch mal einen anderen.

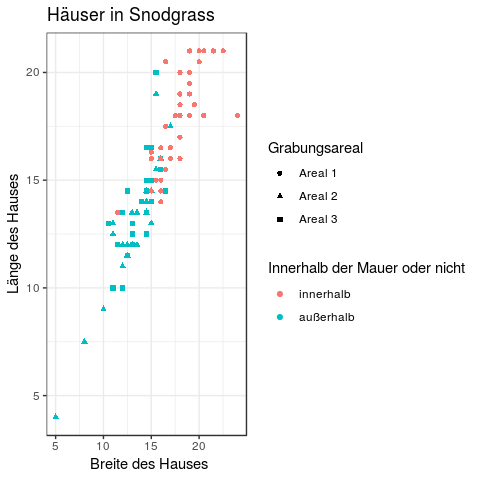
data("Snodgrass")  
  
  
ggplot(data = Snodgrass)+  
 geom\_point(aes(x = Width, y = Length, shape = Segment, color = Inside))+   
 labs(x =" Breite des Hauses",  
 y ="Länge des Hauses",  
 title = "Häuser in Snodgrass")+  
 theme\_bw()



Hmmmhh, interessant. Aber ich vermute, der normale Leser des Diagramms kann nicht erkennen, was “Inside” für eine Information beinhaltet. Kann man vllt die Beschriftung der Legende ändern?

Man kann!!

ggplot(data = Snodgrass)+  
 geom\_point(aes(x = Width, y = Length, shape = Segment, color = Inside))+   
 labs(x =" Breite des Hauses",  
 y ="Länge des Hauses",  
 title = "Häuser in Snodgrass")+  
 theme\_bw()+  
 scale\_colour\_discrete(name ="Innerhalb der Mauer oder nicht",  
 breaks=c("Inside","Outside"),  
 labels=c("innerhalb", "außerhalb")) +  
 scale\_shape\_discrete(name ="Grabungsareal",  
 breaks=c("1","2","3"),  
 labels=c("Areal 1", "Areal 2", "Areal 3"))



Was beudetet das alles?

Mit scale\_colour\_discrete kann ich Legenden (scales) verändern, die mit “color” innerhalb des aesthetics-Bereichs meines Codes für die Graphik definiert werden und die DISKRET sind (also v.a. nominale / ordinale Daten).

Hier benenne ich den Legendentitel mit “Name =” um.

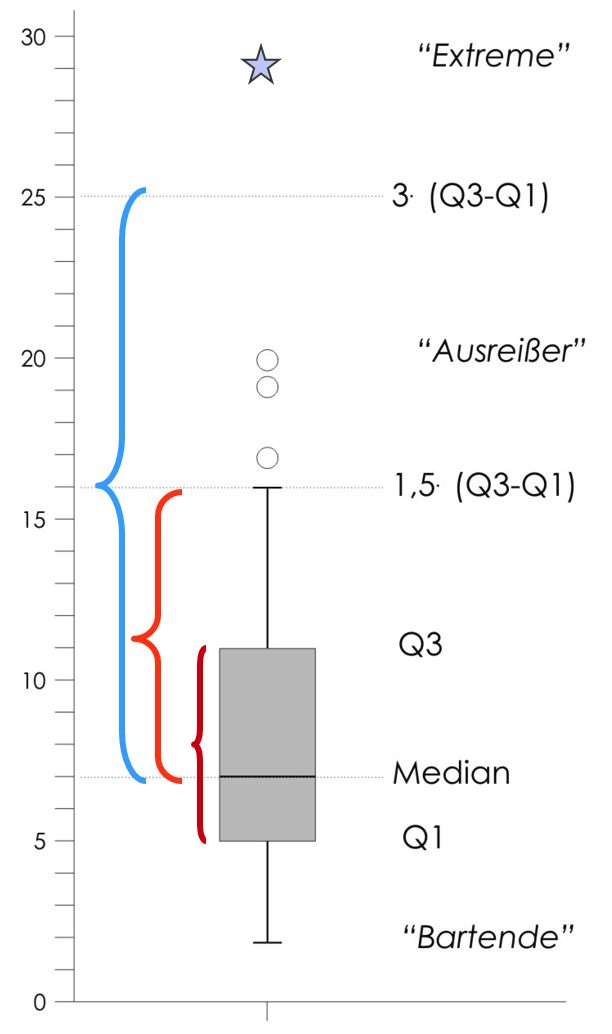
“breaks” bezeichnet die Werte in meiner Spalte, die dann mit den “labels” in der nächsten Zeile umbenannt werden.

Das gleiche kann ich mit der Legende für die FORM der Punkte machen: scale\_shape\_discrete.

Voll gut! Dann lernen wir doch noch andere Visualisierungsmöglichkeiten kennen.

# Boxplot

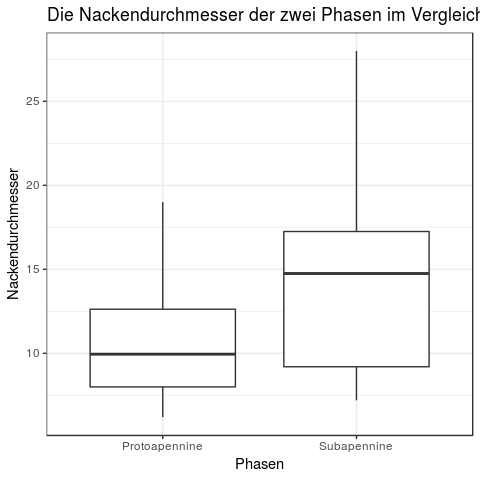
Aufbau eines Boxplotdiagramms

* Q1 = 1. Quartil. Bis hier liegen die ersten 25% meiner Werte, wenn ich sie aufsteigend sortiere
* (Q2 =) Median, den kennen wir schon. Bis hier liegen 50 % meiner Werte, wenn ich sie aufsteigend sortiere
* Q3 = 3. Quartil, bis hier liegen 75% meiner Werte, wenn ich sie aufsteigend sortiere
* Q3 - Q1 ist der Quartilsabstand: In diesem Bereich um den Median herum liegen 50% der “mittleren” Werte. Er wird durch die Box gekennzeichnet
* Bartenden sind das 1,5fache des Quartilsabstandes vom Median aus gerechnet (oder am Ende der Verteilung)
* Extreme liegen außerhalb der Bartenden
* Ausreißer sind mehr als das 3fache des Quartilsabstandes vom Median entfernt

Ein Boxplottdiagramm eignet sich sehr gut, um mehrere Verteilungen EINER Variablen zu vergleichen. Also mehrere Gruppen in meinem Datensatz, aber immer die gleiche Variable.

Ein Beispiel:

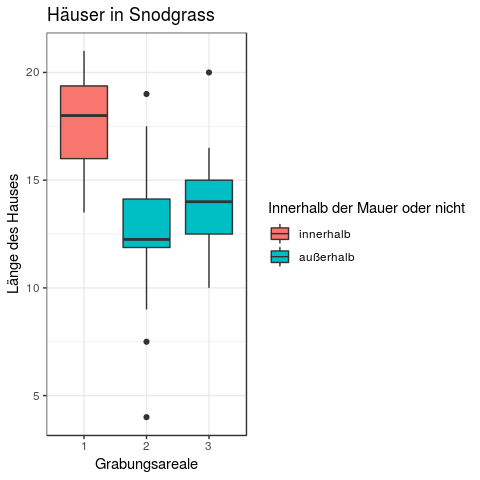
ggplot(data = BACups)+  
 geom\_boxplot(aes(x = Phase, y = ND)) +   
 labs(x ="Phasen",  
 y ="Nackendurchmesser",  
 title = "Die Nackendurchmesser der zwei Phasen im Vergleich")+  
 theme\_bw()



Ich kann gut erkennen, dass die subappeninen Nackendurchmesser größer sind als die der Protoappeninen Phase. Yay!

Und jetzt noch einmal für Daten aus Snodgrass:

ggplot(data = Snodgrass)+  
 geom\_boxplot(aes(x = Segment, y = Length, fill = Inside))+   
 labs(x ="Grabungsareale",  
 y ="Länge des Hauses",  
 title = "Häuser in Snodgrass")+  
 theme\_bw()+  
 scale\_fill\_discrete(name ="Innerhalb der Mauer oder nicht",  
 breaks=c("Inside","Outside"),  
 labels=c("innerhalb", "außerhalb"))



Das kann man doch zur Interpretation nutzen, nicht wahr?!

# Bilder sichern

Jetzt haben wir die Bilder bisher nur in R erstellt, wir haben sie noch nicht sinnvoll abgespeichert.

Das ist ganz einfach.

Den *letzten* Plot, den wir erstellt haben, können wir mit diesem Code speichern:

ggsave("Plotname.png", width = 4, height = 4, units = "cm", dpi = 300)

Wir geben den Namen den die Datei haben soll (das kann auch ein Pfad werden, damit es in einem bestimmten Ordner gespeichert wird), wir geben mit der Endung das Dateiformat udn sagen mit width und height wie groß das Bild in units = cm abgespeichert werden soll. Außerdem die Auflösung mit dpi. Super praktisch!

(D. h. ich hab das Bild als png in einer Größe von 4x4cm abgespeichert. Es hat eine Auflösung von 300 dpi und liegt in dem gleichen Ordner wie mein Rmarkdown-Dokument, weil ich keinen Pfad angegeben habe)