Hauptkomponentenanalyse

Schmidt, Sophie C.

23 September, 2023

Die Hauptkomponentenanalyse ist in R base implementiert. Wir nutzen hier die Funktion prcomp().

Nehmen wir als Beispiel unsere Piraten und schauen, wie die metrischen Variablen height, weight, beard.length und sword.time mithilfe einer PCA auf zwei Faktoren reduziert werden können.

# Pakete und Daten  
library(yarrr)  
data("pirates")

Da der Datensatz sehr groß ist (eigentlich ja gut), für die Visualisierung das aber schwierig wird, ziehen wir eine Stichprobe:

# Stichprobe definieren  
pir <- pirates[sample(nrow(pirates), 50), ] # aus der Anzahl der Zeilen im Datensatz (nrow) wird eine Zufallsstichprobe (sample) von 50 gezogen und dieses sample genutzt, um Zeilen auszuwählen -- in eckigen Klammern hinter dem eigentlichen Datensatz.

ACHTUNG! Wenn wir eine Zufallsstichprobe ziehen, dann sieht diese Zufallsstichprobe bei jedem von uns anders aus. Und jedesmal, wenn wir diesen Code wiederholen, wird das Ergebnis anders aussehen. Da wir aber bei einer Stichprobe von 50 schon hoffen können, dass das Ergebnis halbwegs repräsentativ ist, sollten wir keine komplett unterschiedlichen Ergebnisse bekommen.

Hier haben wir schon das Problem, das angesprochen wurde: Die Bartlänge reicht nur bis 40, wohingegen Gewicht erst bei 33 beginnt und bis 105.6 geht. Also müssen wir bei der Berechnung der PCA die Werte normalisieren. Das machen wir, in dem wir die Argumente center und scale. auf TRUE setzen:

# Ergebnis PCA als Objekt definieren  
pirates.pca <- prcomp(pir[,c(4:5,14:15)], # Auswahl der metrischen stetigen Spalten  
 center = TRUE, # zentrieren  
 scale. = TRUE) # normalisieren

Jetzt haben wir die PCA einem Objekt zugewiesen und können uns Details mit dem üblichen summary anzeigen lassen:

# Zusammenfassung der Ergebnisse  
sum <- summary(pirates.pca)  
sum  
#> Importance of components:  
#> PC1 PC2 PC3 PC4  
#> Standard deviation 1.624 0.9560 0.62233 0.25040  
#> Proportion of Variance 0.659 0.2285 0.09682 0.01567  
#> Cumulative Proportion 0.659 0.8875 0.98433 1.00000

Wir haben “nur” 4 PCs, d.h. principal compontents (Hauptkomponenten), weil wir 4 Variablen in die Analyse einbezogen haben. Wir sehen jetzt die Standardabweichung, Anteil der erklärten Varianz und “cumulative proportion”, dh. den Wert, wenn wir die Varianz dieser und aller “vorhergehenden Hauptkomponenten” zusammenrechnen.

Wir sehen also, dass die erste Hauptkomponente 66% der Varianz erklärt, die zweite erklärt 23% und zusammen erklären sie 89% der Varianz.

Wenn ihr auf den kleinen Pfeil neben pirates.pca in der Environment klickt oder str(pirates.pca) ausführt, seht ihr, was noch alles in der Variablen “drinsteckt”

Das Zentrum (center), die (normalisierten) Mittelwerte (scale), Standardabweichung (sdev) jeder Hauptkomponente, die Faktorenladungen der ursprünglichen Ladungen im Bezug auf die Hauptkomponenten (rotation) sowie die Werte jedes Punktes beschrieben durch die neuen Achsen (x).

Wenn ihr euch head(pirates$x) einmal anschaut seht ihr die ersten 6 “neuen” Variablen. Es werden die gleichen Punkte beschrieben wie vorher durch height, weight, beard.length und sword.time, aber diesmal mit neuen Werten entlang neuer “hypothetischer” Achsen.

Das würden wir jetzt eigentlich gern visualisieren. Das “schönste” Paket dafür liegt unter Github und heißt ggbiplot. Wir haben “nur” 4 PCs, d.h. principal compontents (Hauptkomponenten), weil wir 4 Variablen in die Analyse einbezogen haben. Wir sehen jetzt die Standardabweichung, Anteil der erklärten Varianz und “cumulative proportion”, d.h. den Wert, wenn wir die Varianz dieser und aller “vorhergehenden Hauptkomponenten” zusammenrechnen.

Wenn ihr auf den kleinen Pfeil neben pirates.pca in der Environment klickt oder str(pirates.pca) ausführt, seht ihr, was noch alles in der Variablen “drinsteckt”

Das Zentrum (center), die (normalisierten) Mittelwerte (scale), Standardabweichung (sdev) jeder Hauptkomponente, die Faktorenladungen der ursprünglichen Ladungen im Bezug auf die Hauptkomponenten (rotation) sowie die Werte jedes Punktes beschrieben durch die neuen Achsen (x).

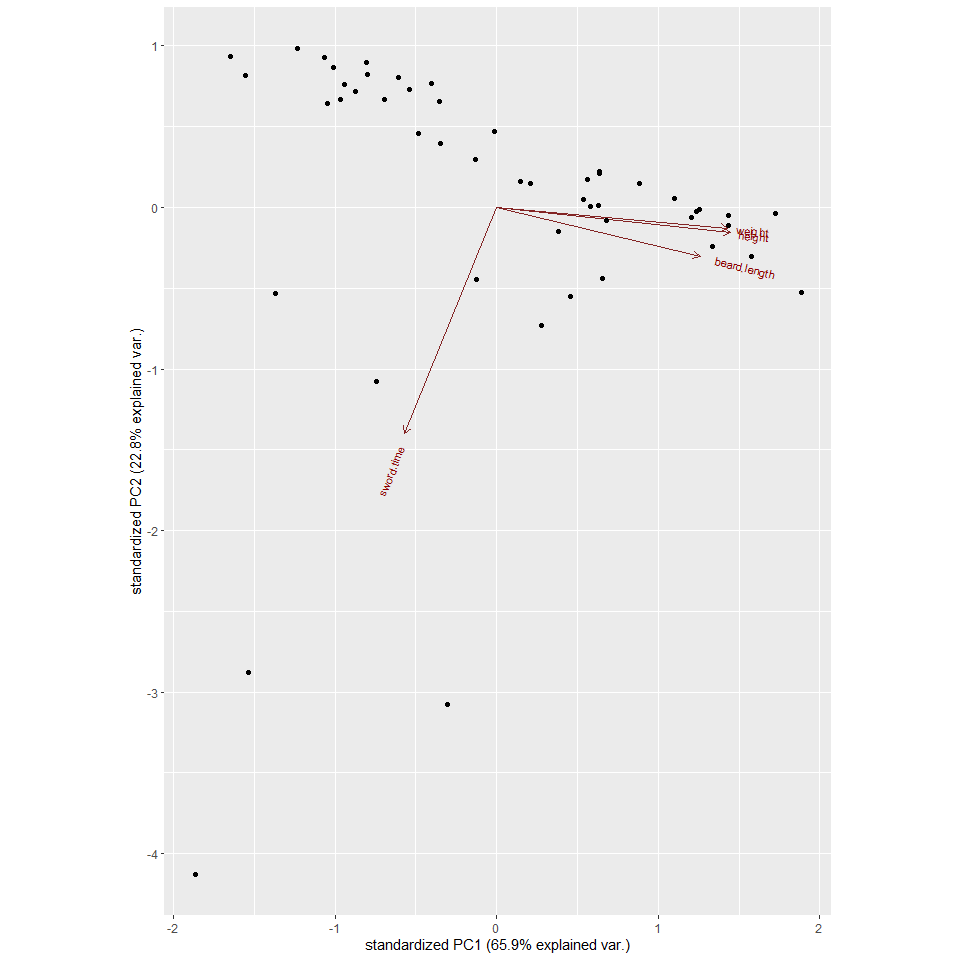
Wenn ihr euch head(pirates$x) einmal anschaut seht ihr die ersten 6 “neuen” Variablen. Es werden die gleichen Punkte beschrieben wie vorher durch height, weight, beard.length und sword.time, aber diesmal mit neuen Werten entlang neuer “hypothetischer” Achsen.

Das würden wir jetzt eigentlich gern visualisieren. Das “schönste” Paket dafür liegt unter Github und heißt ggbiplot. Wenn man sich ein Paket von Github installieren möchte, dann braucht man dafür das Paket devtools. Installiert Euch das wie immer und und holt euch dann ggbiplot von Github.

# Pakete installieren und laden   
install.packages("devtools")  
library(devtools)  
install\_github("vqv/ggbiplot")

Jetzt plotten wir damit erst einmal ganz einfach unsere PCA:

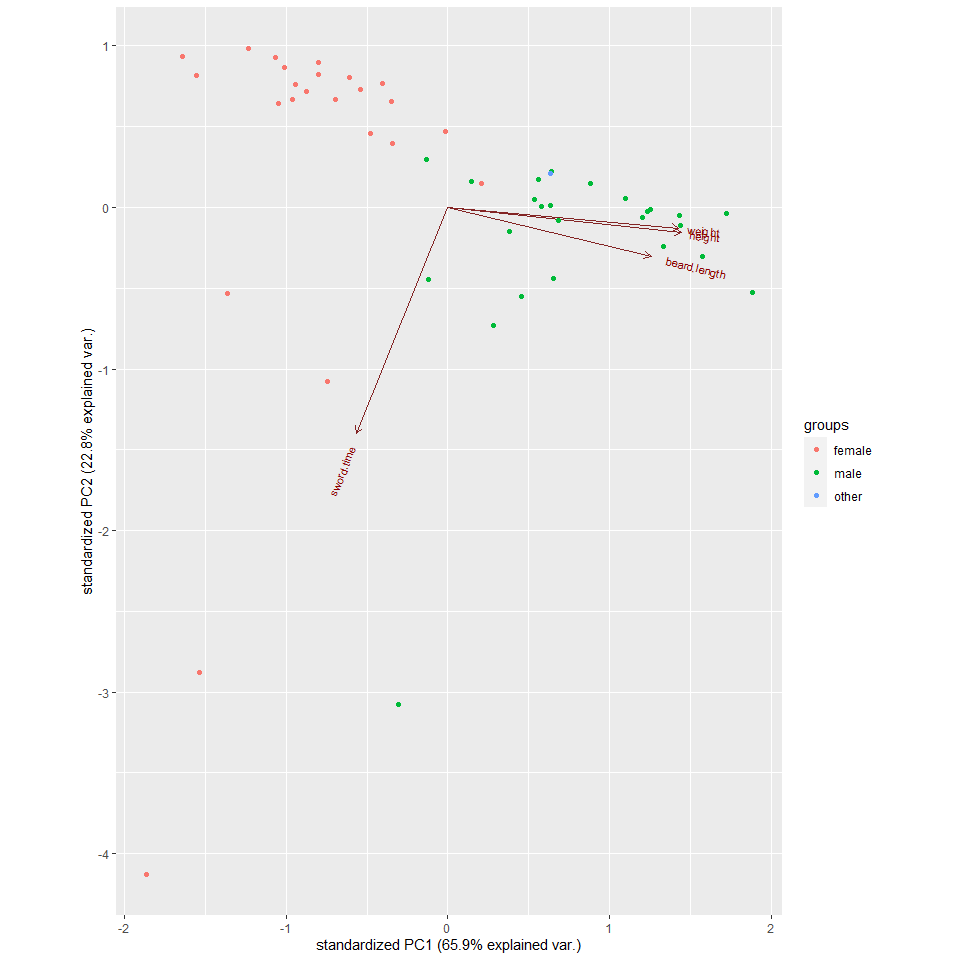
# Paket laden  
library(ggbiplot)  
  
# Biplot erstellen  
ggbiplot(pirates.pca)

 Wie praktisch! Wir sehen: Die Achsen wurden für uns schon beschriftet, sogar mit der Beschreibung, wie viel Prozent der Varianz erklärt wird. Die roten Pfeile mit der roten Beschriftung zeigen in welche Richtungen die unterschiedlichen ursprünglichen Variablen die Daten “ziehen”. Wie man sieht, liegen height und weight fast direkt übereinander: Diese beiden korrelieren ja auch stark, wie wir schon wissen.

Jetzt hübschen wir das alles noch ein bisschen auf. Wir versuchen ja eigentlich mit der Darstellung etwas zu erklären. Welche Faktoren könnten denn im Hintergrund die Verteilung dieser Punkte erklären?

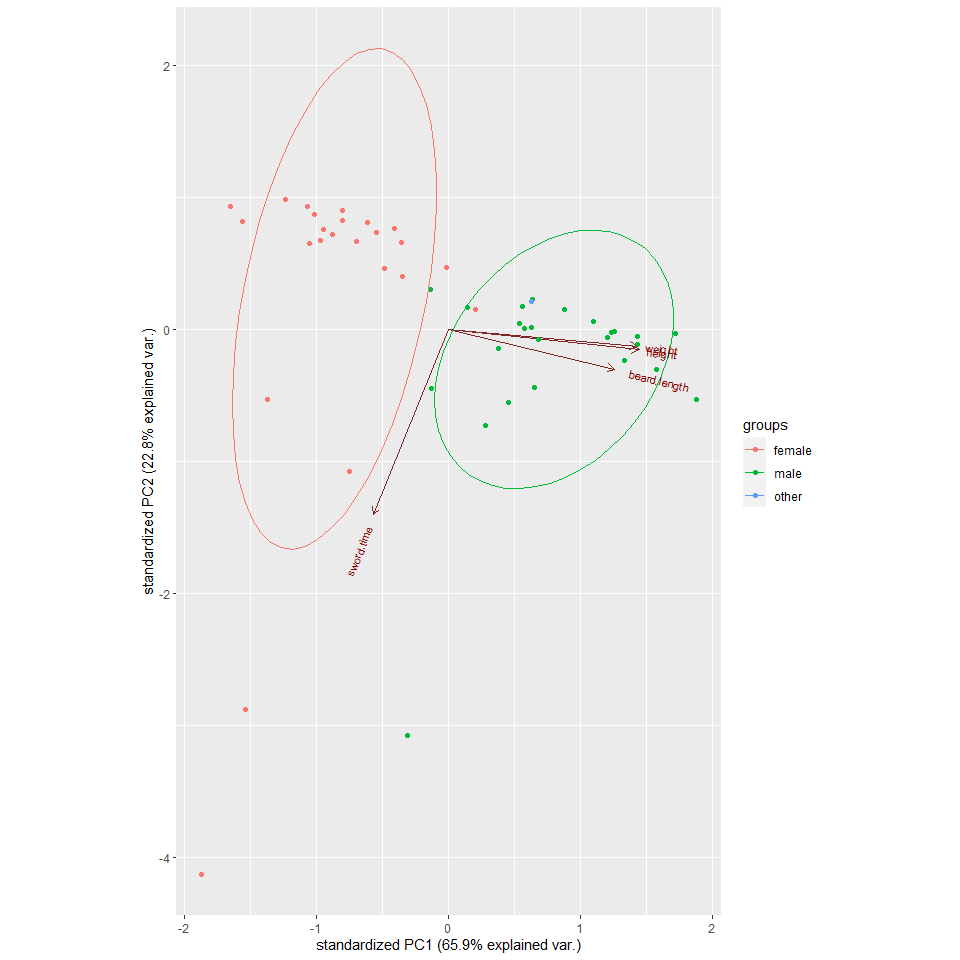
Ein typisches Thema beim Bartwuchs wäre doch zB das Geschlecht. Färben wir die Punkte also nach Geschlecht ein. pir$sex ist der Vektor, der die Gender-Identität unserer Piraten in der Stichprobe beschreibt. Den wählen wir jetzt, um unsere Daten zu gruppieren.

# Biplot mit Gruppenvariable erstellen  
ggbiplot(pirates.pca, groups = pir$sex)



Ach sie man einer an! Die ganzen Frauen liegen nah bei einander. Und die Männer bilden auch eine Gruppe! Können wir das noch klarer darstellen? Ja, mit dem Befehl ellipse = TRUE wird die Gruppe umrandet:

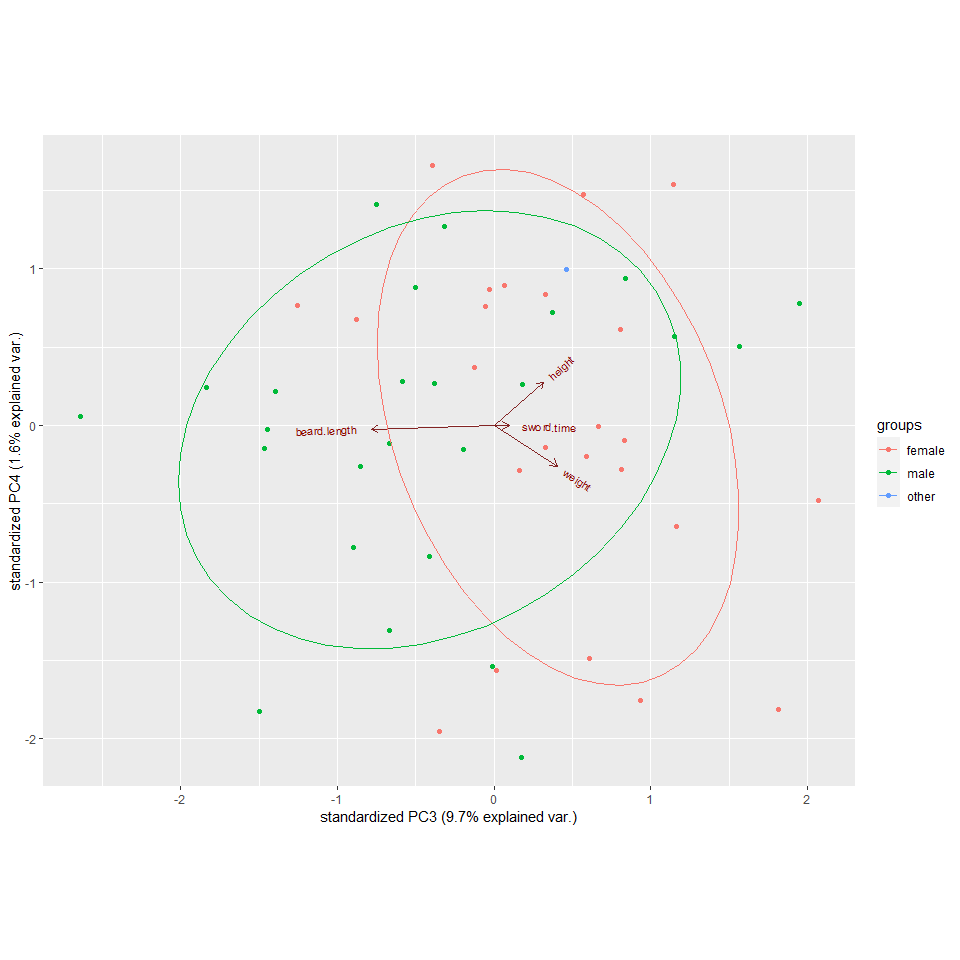
# Biplot mit Ellipsen erstellen  
ggbiplot(pirates.pca, groups = pir$sex, ellipse = TRUE)



Klasse. Klar trennbare Gruppen entlang von PC1, würde ich sagen - die beiden Gruppen liegen entlang dieser Gruppe fast ohne Überlappung “nebeneinander”.

Wenn wir jetzt uns ncoh einmal die “Ansicht” auf den 3. und 4. Eigenvektor anschauen, werden wir sehen, warum es sinnvoll ist:

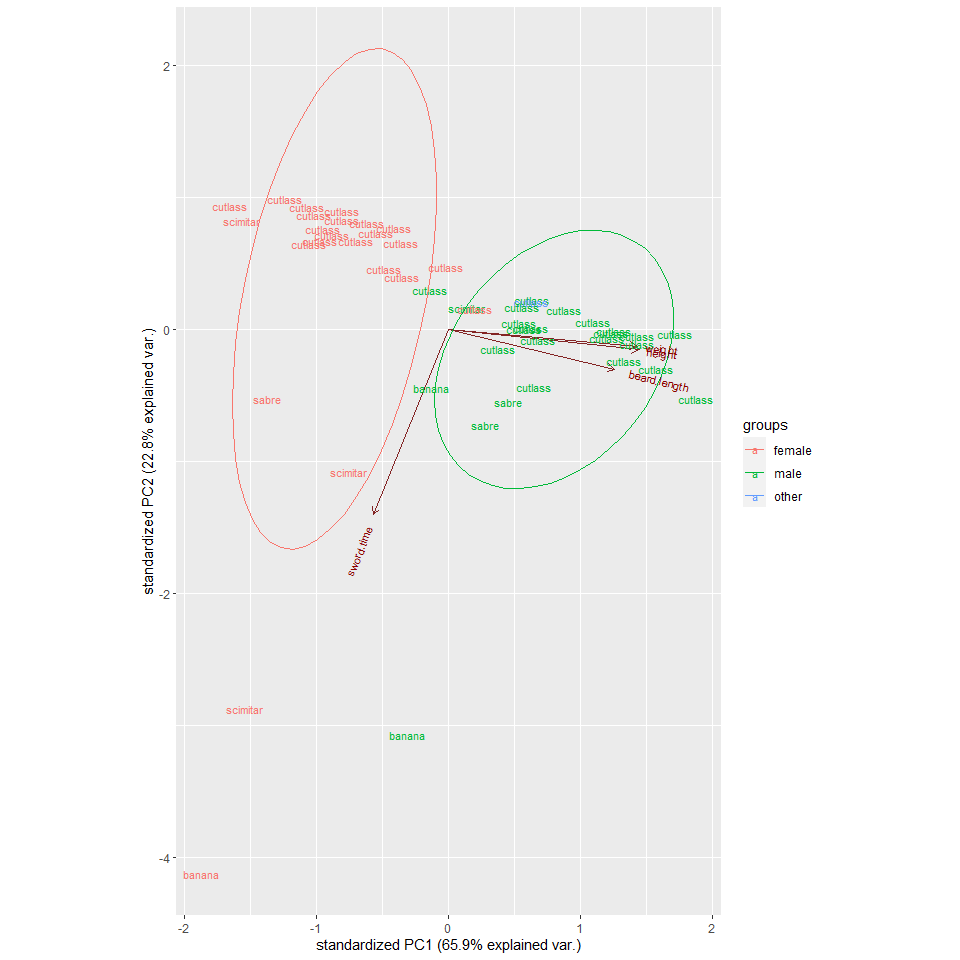
# Biplot für den 3. und 4. Eigenvektor erstellen  
ggbiplot(pirates.pca, choices=c(3,4), groups = pir$sex, ellipse = TRUE) # choices hilft einem, die Hauptkomponenten auszusuchen, die dargestellt werden sollen



Wir sehen: Die Daten streuen viel mehr, die beiden Gruppen sind nicht mehr gut voneinander zu trennen. Das liegt daran, dass die gewählten Eigenvektoren die Varianz nicht gut erklären. PC3 und PC4 hängen nicht mit dem Geschlecht zusammen.

Wenn man kleinere Datensätze hat, kann man sich anstelle der Punkte zB die Namen oder andere Label anzeigen lassen. Das führe ich hier einmal vor, aber es ergibt nicht viel Sinn bei den Piraten. Nehmen wir den Schwerttyp.

# Biplot mit Labels erstellen  
ggbiplot(pirates.pca, groups = pir$sex, ellipse = TRUE, labels = pir$sword.type)



Ach, nagut. Vielleicht erklärt der Schwerttyp ein bisschen die Variable sword.time. Schaut mal,ob das bei euch auch rauskommt. ;-)

Sooo. Super! Wir haben mithilfe der PCA herausbekommen, dass sich unser Datensatz von Piraten anhand von zwei Hauptkomponenten gut in zwei Gruppen (Geschlecht) unterteilen lässt. Schaut man in dem plot, welche der Variablen für die Verteilung auf den zwei Komponenten besonders wichtig sind, so sind das Größe und Gewicht auf der ersten Komponente und Bartlänge etwas “schräg” dazu zum Teil auf der zweiten Komponente. Der Wert Schwertzeit scheint dabei keine so große Rolle gespielt zu haben. Wir können also annehmen, dass das Geschlecht einen Einfluss auf Größe, Gewicht und Bartlänge hat.

Das ist jetzt vielleicht nicht allzu überraschend… Aber in anderen Fällen mag man das nicht so wissen.

Probieren wir doch noch einmal etwas ähnliches mit den Pinguinen!

# Pakete und Daten laden  
library(palmerpenguins)  
data(penguins)

**Aufgabe** Im Pinuguindatensatz gibt es vier metrische Variablen. Berechnet für sie eine Hauptkomponentenanalyse, färbt die Punkte nach einer der nominalen Faktoren ein, ersetzt die Punkte mit einem Label und schaut so, welcher Variable einen Einfluss auf die Verteilung haben könnten.

Außerdem: ggbiplot lässt im Hintergrund ggplot laufen. Fügt noch einen sinnvollen Titel hinzu (ggtitel()) und ändert den Hintergrund mit theme\_bw()!

ACHTUNG. Im Pinguin-Datensatz stecken NA-Werte. Man kann bei PCA um den Datensatz na.omit() legen, dann werden die Reihen mit NA-Werten nicht in die Berechnung einbezogen. Dann hat man aber das Problem, dass, wenn man die Punkte in der Visualisierung einfärben will, mehr Zeilen im originalen “beschreibenden” Datensatz hat als in der Berechnung. Damit kann R sie nicht mehr übereinander legen und wirft eine Fehlermeldung aus.

Deswegen sollten wir vorher einen Datensatz konstruieren, der keine NAs mehr enthält. Versucht das einmal selber herauszubekommen!