Centre for Biological Control - R Coding Club

Guy F. Sutton*

09 July, 2020

Tutorial #11 - Principal component analysis (PCA)

What is a Principal component analysis (PCA)?

Principal component analysis (PCA) is a statistical technique that allows us to reduce the information contained in many variables into a smaller number of summary variables (or indices). This is usually done to make a large amount of data more easily interpretable and analysable.

- *Example 1*: Say you collect a whole range of water chemistry data (e.g. C, N, K) during a study that tries to predict whether water chemistry can explain the geographical distribution of the invasive waterweed, *Egeria densa*. In doing so, you may have 20-odd water chemistry variables in the end. In this case, you cannot fit each variable to a model (e.g. GLM), for example, unless you had 1000's of data points (remeber: the more variables you have in a model, the more data you need). Instead, we can use PCA to reduce the water chemistry data down to maybe 2 or 3 variables representing gradients in water chemistry. For example, your first variable may be a C:N gradient, and your second variable may be a K/P gradient, and fit these 2 summary variables as explanatory variables in a GLM.
- *Example 2*: Another example may be where taxonomists take many morphological measurements for many specimens when trying to morphologically describe new species. In my field of wasp taxonomy, researchers may measure 100+ morphological characters per specimen. We can perform a PCA and reduce these 100+ measurements down to maybe 3-10 axes. The first variable may now be gradient of antennal characteristis (e.g. separating species based on how many segments can be counteted in their antennae, and the number of hairs on each antennal segment), while the second variable could separate species based on the distance between eyes and distance between their eyes and top of the head, and the third variable representing a gradient of ovipositor length and widths, and so on...

Aims of this tutorial:

1. Understand what the purpose of PCA
2. Run a simple PCA
3. Correctly interpret PCA output
4. Understand how PCA can be used in further analyses (NB: You are not expected to understand the additional modelling exercise *per se* - it is just there to illustrate how you would use the data obtained from PCA).

*Rhodes University, Grahamstown, South Africa, g.sutton@ru.ac.za

Raw data

Most of the code and ideas in this script have been adapted and gleaned from three amazing blogposts by Julia Silge and Allison Horst.

https://juliasilge.com/blog/multinomial-volcano-eruptions/

https://juliasilge.com/blog/cocktail-recipes-umap/

https://allisonhorst.github.io/palmerpenguins/articles/articles/pca.html

The data we will use comes from Allison Horst's `palmerpenguins` package which we must install remotely. The data is a set of body size and morphology measurements on a range of penguins belonging to different species, from different islands and genders.

```
# Take a look at the clean data
head(penguins_clean)
```

```
## # A tibble: 6 x 7
##   species island bill_length_mm bill_depth_mm flipper_length_~ body_mass_g sex
##   <chr>   <fct>           <dbl>         <dbl>             <dbl>       <dbl> <fct>
## 1 Adelie  Torge~           39.1          18.7               181        3750 MALE
## 2 Adelie  Torge~           39.5          17.4               186        3800 FEMA~
## 3 Adelie  Torge~           40.3          18                 195        3250 FEMA~
## 4 Adelie  Torge~           36.7          19.3               193        3450 FEMA~
## 5 Adelie  Torge~           39.3          20.6               190        3650 MALE
## 6 Adelie  Torge~           38.9          17.8               181        3625 FEMA~
```

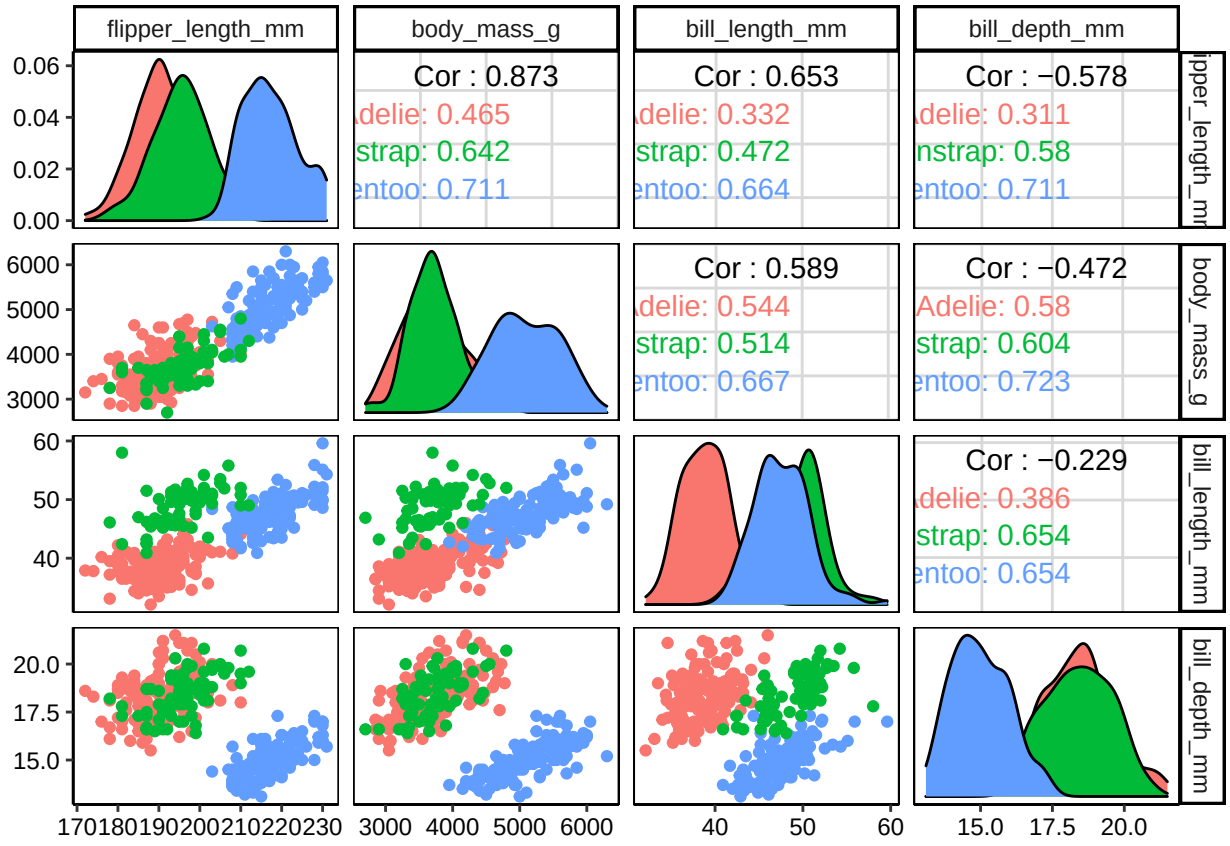
Exploratory data visualisation

Let's make some plots and visualisations to get a feel for the data.

```
# How many penguin individuals do we have for each
# species and island?
# Use .drop = FALSE to keep species x island combo with 0 penguins
penguins_clean %>%
  count(species, island, .drop = FALSE)
```

```
## # A tibble: 9 x 3
##   species   island        n
##   <chr>     <fct>     <int>
## 1 Adelie    Biscoe       44
## 2 Adelie    Dream        55
## 3 Adelie    Torgersen    47
## 4 Chinstrap Biscoe        0
## 5 Chinstrap Dream        68
## 6 Chinstrap Torgersen     0
## 7 Gentoo    Biscoe      119
## 8 Gentoo    Dream         0
## 9 Gentoo    Torgersen     0
```

```r
# Let's check to see whether the relationships between
# the variables are linear(-ish)
# e.g. is flipper_length_mm linearly correlated with body_mass_g?
penguins_clean %>%
  # Select the variables to plot
  dplyr::select(species,
                bill_length_mm:body_mass_g) %>%
  # Plot the variables
  # Numeric variables go within the columns(...) argument to plot
  # correlations between them
  GGally::ggpairs(., aes(color = species),
                  columns = c("flipper_length_mm",
                              "body_mass_g",
                              "bill_length_mm",
                              "bill_depth_mm"))
```



Looks pretty good. Linear relationships all over the joint. PCA requires relationships are approximately linear between numeric variables, although this assumption is usually relaxed, and hardly ever applied or checked for, in my experience.

Perform a PCA

Below, we are going to use `recipes` from the `tidymodels` package to perform a PCA. Beforehand, we need to pre-process the data:

- (1) remove any NA values - PCA cannot handle any NA values
- (2) center all predictors
- (3) scale all predictors
- Steps 2 and 3 make the data have equal variance (variance = 1), which removes the issue with scales of different variables having an influence on our final results. For example, if penguin body mass ranges from 10g to 30g, and flipper_length_mm ranges from 100mm to 500mm, PCA will undoubtedly say flipper length is NB, purely because the units are larger.

Step 1: Prepare our recipe for PCA

```
# Pre-process data for our PCA
pca_recipe <-
  # Provide the data frame name
  recipe(~., data = penguins_clean) %>%
  # Any columms that aren't numeric (i.e. charatcer variables)
  # need to be defined here
  update_role(species, island, sex, new_role = "id") %>%
  # Center and scale variables
  step_normalize(all_predictors()) %>%
  # Starts the PCA process - doesn't actually do anything yet
  step_pca(all_predictors())

# Take a look at what we have done so far
pca_recipe
```

```
## Data Recipe
##
## Inputs:
##
##       role #variables
##         id          3
##  predictor          4
##
## Operations:
##
## Centering and scaling for all_predictors
## No PCA components were extracted.
```

Step 2: Run the PCA

```
# Run the PCA
pca_prep <- prep(pca_recipe)
pca_prep
```

```
## Data Recipe
##
## Inputs:
##
##       role #variables
##         id          3
##  predictor          4
##
## Training data contained 333 data points and no missing data.
##
## Operations:
##
## Centering and scaling for bill_length_mm, ... [trained]
## PCA extraction with bill_length_mm, bill_depth_mm, ... [trained]
```

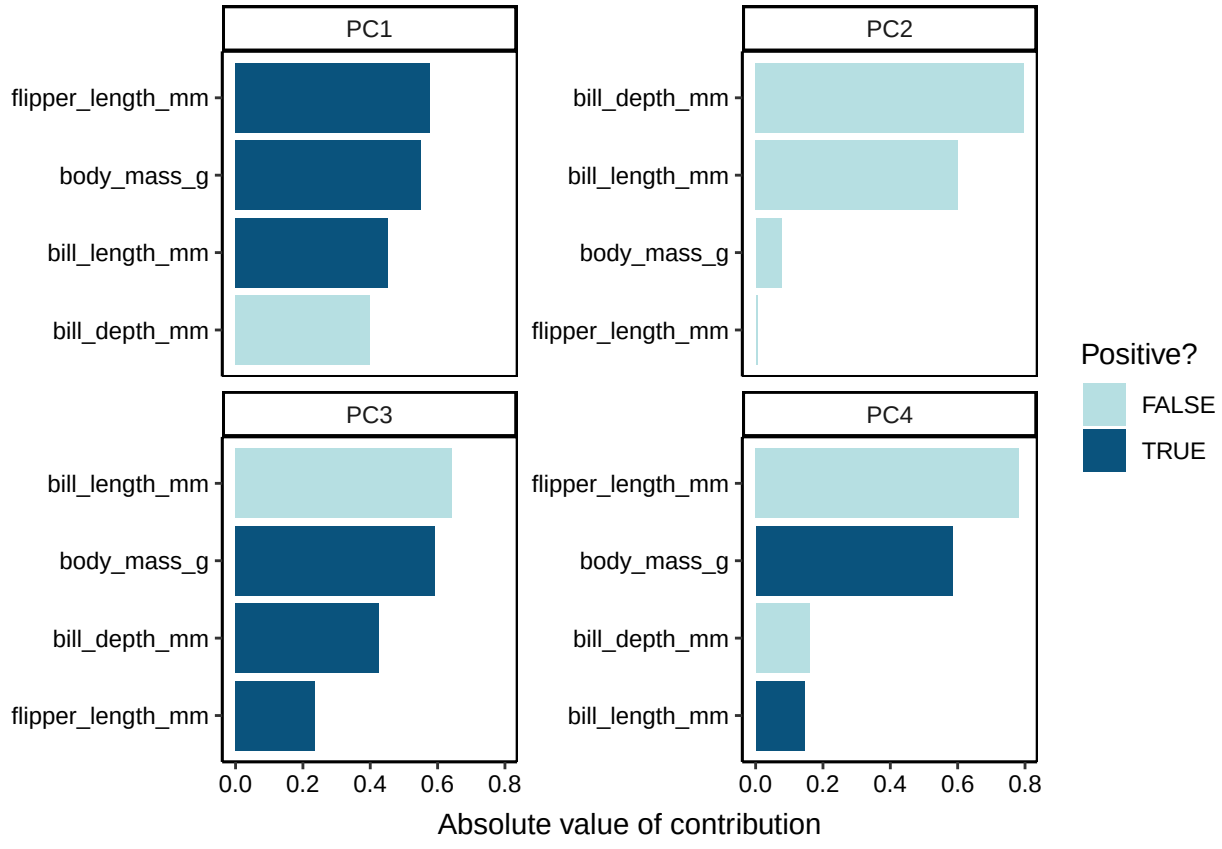
```
# Extract the results from the PCA in tidy-format
tidied_pca <- tidy(pca_prep, 2)
tidied_pca
```

```
## # A tibble: 16 x 4
##    terms              value component id
##    <chr>              <dbl> <chr>     <chr>
##  1 bill_length_mm     0.454   PC1       pca_8TZAZ
##  2 bill_depth_mm     -0.399   PC1       pca_8TZAZ
##  3 flipper_length_mm  0.577   PC1       pca_8TZAZ
##  4 body_mass_g        0.550   PC1       pca_8TZAZ
##  5 bill_length_mm    -0.600   PC2       pca_8TZAZ
##  6 bill_depth_mm     -0.796   PC2       pca_8TZAZ
##  7 flipper_length_mm -0.00579 PC2       pca_8TZAZ
##  8 body_mass_g       -0.0765  PC2       pca_8TZAZ
##  9 bill_length_mm    -0.642   PC3       pca_8TZAZ
## 10 bill_depth_mm      0.426   PC3       pca_8TZAZ
## 11 flipper_length_mm  0.236   PC3       pca_8TZAZ
## 12 body_mass_g        0.592   PC3       pca_8TZAZ
## 13 bill_length_mm     0.145   PC4       pca_8TZAZ
## 14 bill_depth_mm     -0.160   PC4       pca_8TZAZ
## 15 flipper_length_mm -0.782   PC4       pca_8TZAZ
## 16 body_mass_g        0.585   PC4       pca_8TZAZ
```

Step 3: Visualise the PCA fit

Now we need to visualise which variables are correlated with which axes? These values represent factor loadings - the greater the value for the factor loading, the more strongly that variable is correlated with that PC axis, and the sign gives the direction.

```
# Which variables are correlated with which axes?
tidied_pca %>%
  filter(component %in% paste0("PC", 1:4)) %>%
  group_by(component) %>%
  top_n(8, abs(value)) %>%
  ungroup() %>%
  mutate(terms = reorder_within(terms, abs(value), component)) %>%
  ggplot(aes(abs(value), terms, fill = value > 0)) +
  geom_col() +
  facet_wrap(~component, scales = "free_y") +
  scale_fill_manual(values = c("#b6dfe2", "#0A537D")) +
  scale_y_reordered() +
  labs(x = "Absolute value of contribution",
       y = NULL,
       fill = "Positive?") +
  theme(legend.position = "right")
```



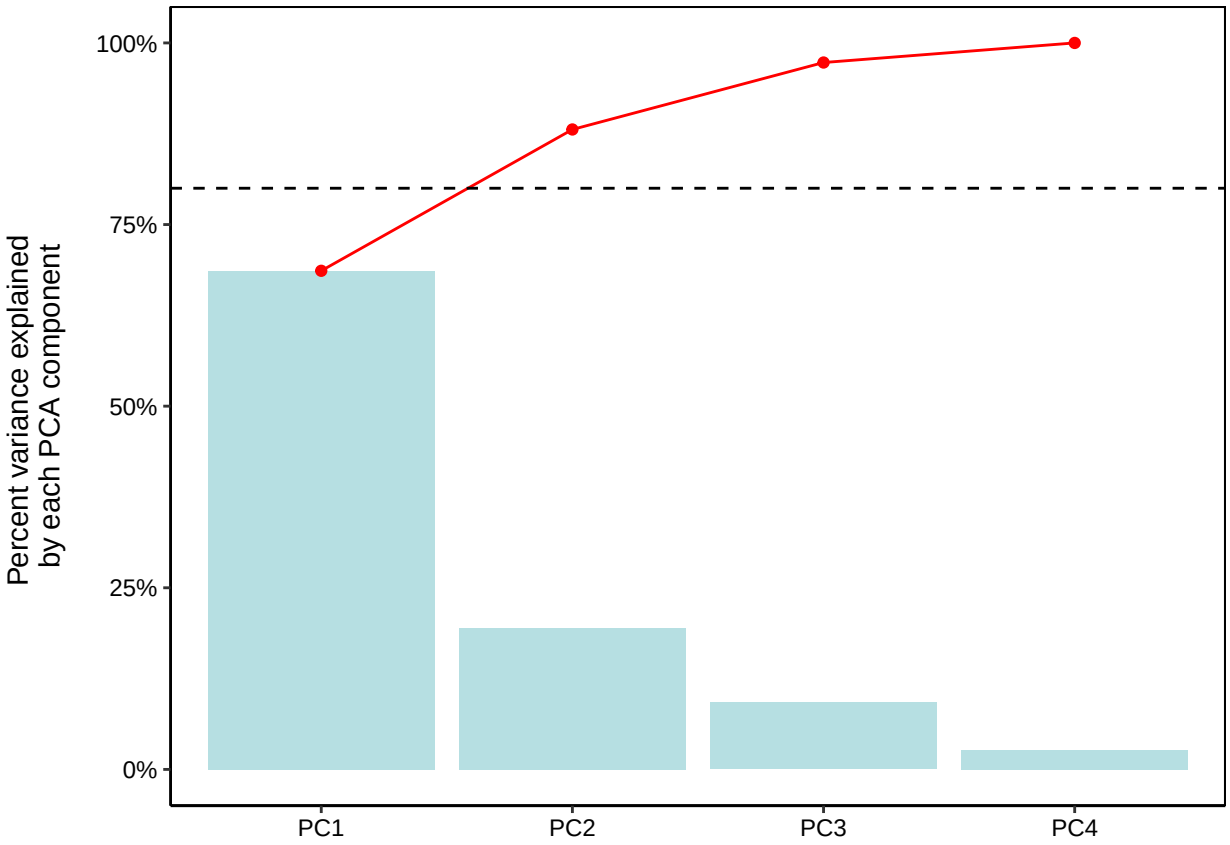
We must look at these loadings to decide on which variables differentiate each PC axis. For example, PC1 is differentiated by flipper_length vs bill_depth - i.e. species with longer flippers, have smaller bill widths.

Step 4: How many PCA axes should we retain?

We have to make a decision as to how many PCA axes should we retain to capture as much variation in the variables we are interested in, without specifying too many variables (otherwise we may as well just fit all the individual variables). A general rule of thumb is to include as many PC axes required to explain 80% cumulative variation, although this will often not be possible (often 4-5 PCA axes may only explain maybe 30-70%). This is a subjective process, and care needs to be taken when making this decision. Evaluating how your results change when adding more axes should guide your decision.

```
# Extract variance explained by each PC axis
sdev <- pca_prep$steps[[2]]$res$sdev
percent_variation <- sdev^2 / sum(sdev^2)

# Convert values into a df
perc_df <- tibble(
  component = unique(tidied_pca$component),
  percent_var = percent_variation,
  percent_cum = cumsum(percent_variation)) %>%
  dplyr::mutate(component = fct_inorder(component)) %>%
  # Make graph
  ggplot(aes(component, percent_var)) +
  # Plot columns for each PC axis
  geom_col(fill = "#b6dfe2") +
  # Plot a line of cumulative percentage
  geom_line(aes(y = percent_cum, group = 1),
            colour = "red") +
  # Plot the points for cumulative percentage too
  geom_point(aes(y = percent_cum, group = 1),
             colour = "red") +
  # Add an 80% cumulative percent mark
  # Usually, most people include as many PC axes required to explain
  # 80% cumulative variation
  geom_hline(yintercept = 0.8, linetype = "dashed") +
  scale_y_continuous(labels = scales::percent_format()) +
  labs(x = NULL,
       y = "Percent variance explained \nby each PCA component")
perc_df
```

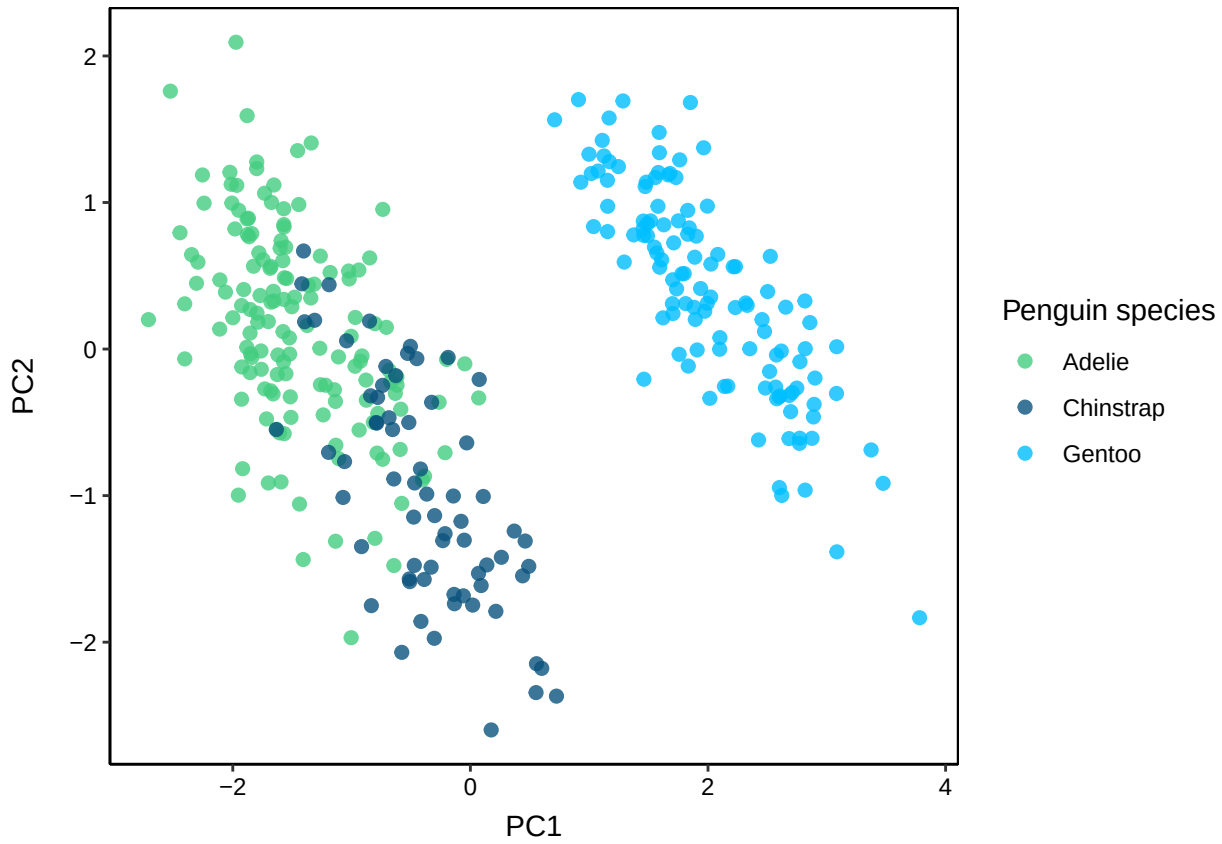


In this example, two PC axes is sufficient (PC1 and PC2). PC1 explained 68.6% and PC2 explained 19.5% of the variation in penguin morphology (88.1% in total).

Step 5: Plot the PCA

Here, we are going to plot only the first two axes PC1 and PC2, as they explained more than 80% of the variation in morphological measurements between penguin species.

```
# Plot PCA (here we plot only the first two axes PC1 and PC2)
pca_plot <-
  # Extract the PC axes scores
  juice(pca_prep) %>%
  # Make a plot
  ggplot(aes(PC1, PC2)) +
  # Add sample points to PCA
  geom_point(aes(color = species),
             alpha = 0.8,
             size = 2) +
  # Specify three colours for the three different penguin species
  scale_colour_manual(values = c("seagreen3", "#0A537D", "deepskyblue1")) +
  # Add a legend for the colours
  theme(legend.position = "right") +
  # Change the title of the legend
  labs(colour = "Penguin species")
pca_plot
```

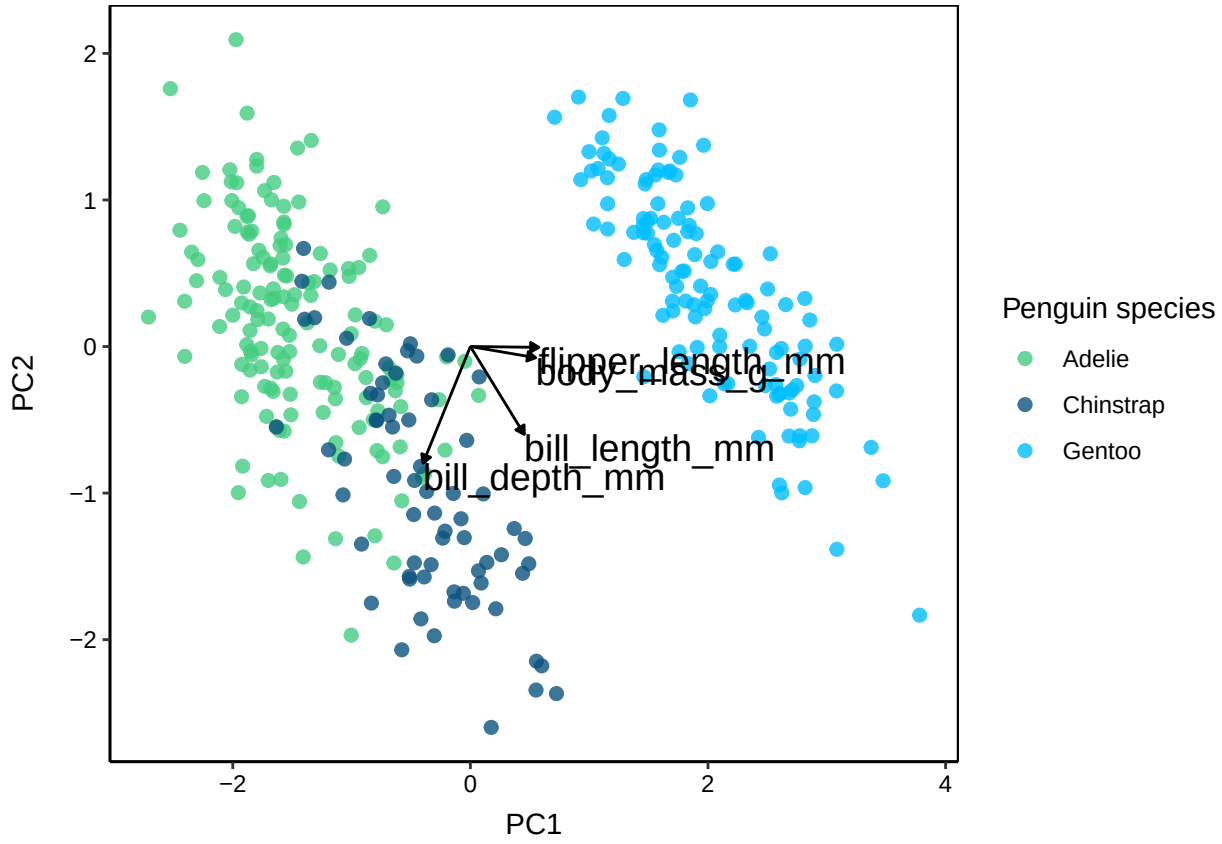


Clearly, the PCA axes can differentiate between Gentoo and the other two species, as indicated by the distinct groupings. Biologically, this means that Gentoo penguins are morphologically distinguishable from the other two species. But, what morphological measurements drive these differences?

Step 5b: Plot the PCA biplot

Below, we are going to plot a biplot, over the PCA that we made on the previous page. A biplot indicates which variables are associated with which PCA axes. This will be more clear with an example.

```
# Now we plot the biplot over the PCA plot
pca_biplot <- pca_plot +
  # Add arrows
  geom_segment(data = pca_wider,
               aes(xend = PC1, yend = PC2),
               x = 0,
               y = 0,
               arrow = arrow_style) +
  # Add variable names
  geom_text(data = pca_wider,
            aes(x = PC1,
                y = PC2,
                label = terms),
            hjust = 0,
            vjust = 1,
            size = 5)
pca_biplot
```



Interpretation:

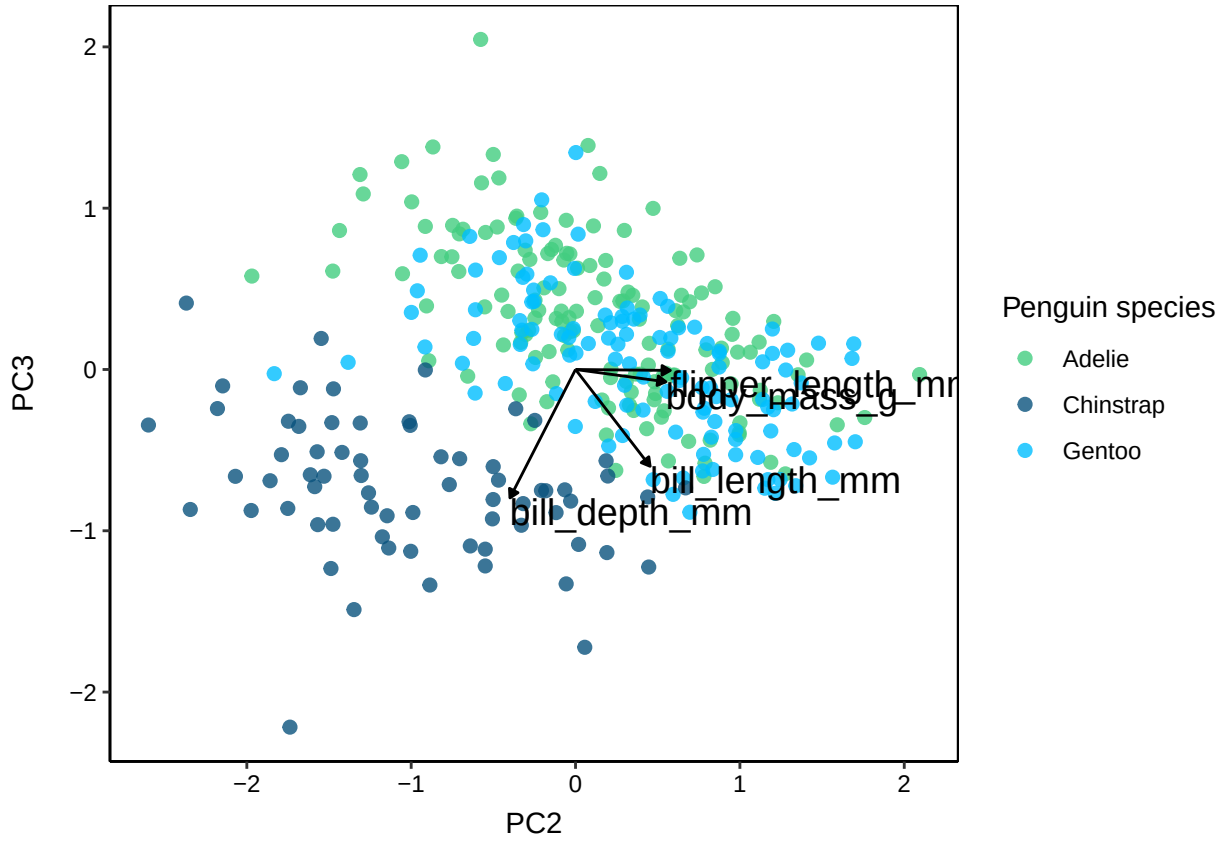
(1) The first axis represents an axis of penguin body mass.

- This axis separates our species into larger (Gentoo) and smaller penguin species (Chinstap and Adelie), based on their respective body masses and flipper lengths.

(2) The second PC axis represents an axis of bill characteristis.

- This axis separates our two remaining penguin species (albeit not very well) into species with larger (Adelie) and smaller (Chinstrap) bills.

While we don't need to look at any additional PC axes, let's add PC3 into the mix, and see if we can differentiate Chinstrap vs Adelie penguins. *Code not shown*



By adding PC3, we can clearly see that bill_depth can help us distinguish between Chinstrap and Adelie penguins.

So what?

So you have now performed your PCA, but what now?

- Well, as I said above, PCA is usually performed to reduce the number of variables in play down to a manageable amount of variables. This dataset was not ideal because there were very few morphological variables to start with.
- In the next session, we will use a machine learning statistical model to determine whether we can accurately determine which species of penguin we have based on their morphological measurements. To do this, we will extract the first three PCA axes and then run a model to see whether these axes (representative of morphological body traits) can accurately delineate penguin species.

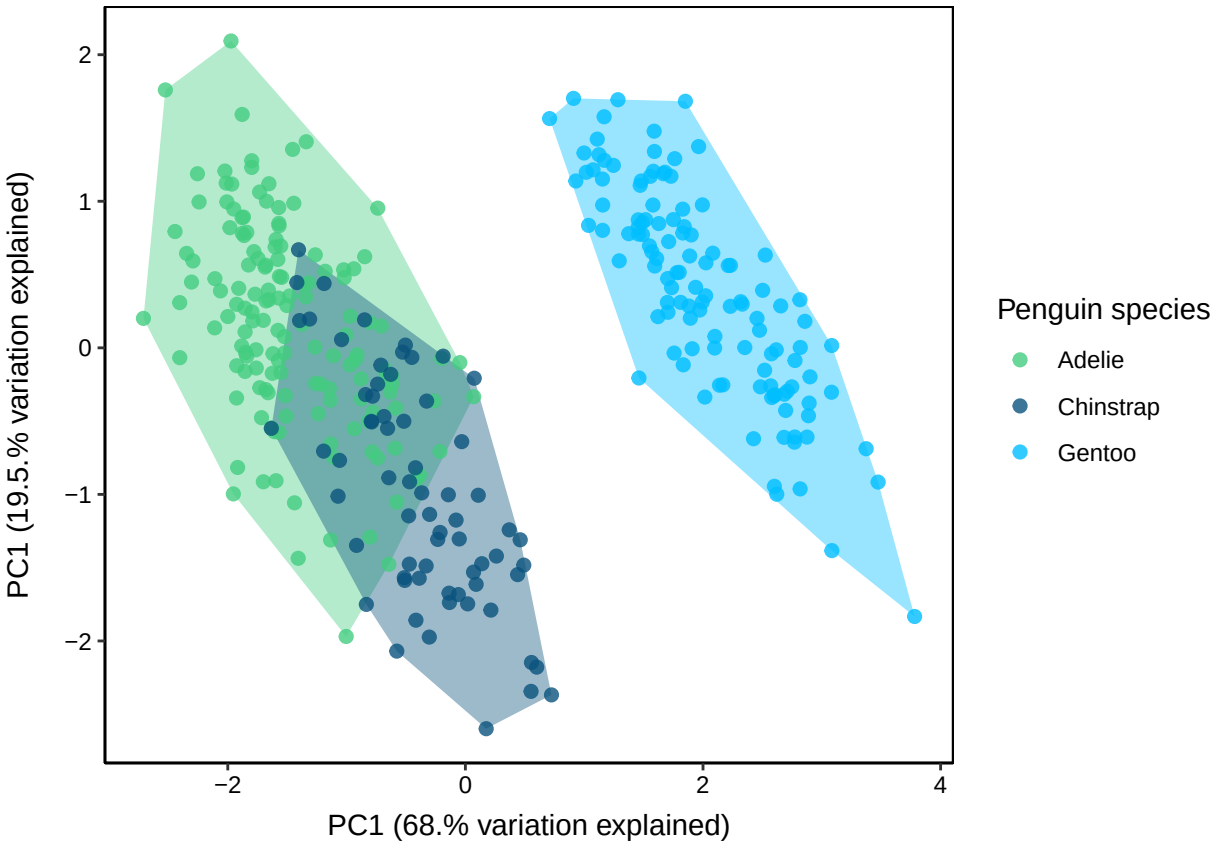
Bonus points

What if we want to add convex hulls per group?

```
# First, extract PC co-ords
pc_points <- juice(pca_prep)

# Second, calculate convex polygons for each species
hulls <- pc_points %>%
  group_by(species) %>%
  slice(chull(PC1, PC2))

# Lastly, make the new plot
pca_hulls <-
  # Extract the PC axes scores
  juice(pca_prep) %>%
  # Make a plot
  ggplot(aes(PC1, PC2)) +
  # Add convex hulls
  # Hulls must come before points!!!
  geom_polygon(data = hulls,
               aes(fill = species),
               alpha = 0.4) +
  # Add sample points to PCA
  geom_point(aes(color = species),
             alpha = 0.8,
             size = 2) +
  # Specify three colours for the three different penguin species
  scale_colour_manual(values = c("seagreen3", "#0A537D", "deepskyblue1")) +
  scale_fill_manual(values = c("seagreen3", "#0A537D", "deepskyblue1")) +
  # Remove the second legend (unnecessary)
  guides(fill = "none") +
  # Add a legend for the colours
  theme(legend.position = "right") +
  # Change the title of the legend
  labs(colour = "Penguin species",
       x = "PC1 (68.% variation explained)",
       y = "PC1 (19.5.% variation explained)")
pca_hulls
```


PC1 (19.5.% variation explained)
2
1
0
−1
−2
−2
0
2
4
PC1 (68.% variation explained)
Penguin species
Adelie
Chinstrap
Gentoo