Exploratory Data Analysis with Principal Components Analysis

Laura Pikkupeura

May 28, 2019

Teachers and slides

- **Laura Pikkupeura**
- PhD studenst in Sandelin and Jensen Labs
- Mail: `laura.pikkupeura@bric.ku.dk`
- Background:
  - MSc Molecular Biomedicine
  - PhD in Stem Cell Biology, Genomics and Transcriptomics

Words of wisdom

"Plotting data can show you something unexpected"

- Hadley Wickham

Exploratory Data Analysis (EDA)

Central questions:

- Which samples are most similar?
- Are there any clusters in the data?
- Do patterns look like we expect them to?
- Are there outliers?
- Is there any unwanted variation - like batches?
- How do we answer these questions when we can't "see" the data?

Lecture Outline

The plan:

- Introduction to data dimensionality
- Dimensionality reduction
- Principal Component Analysis (PCA)
- Hands on experience with real data
- Discussion of outliers

Example datasets:

- `flowers` dataset
- `oil` dataset

Real datasets:

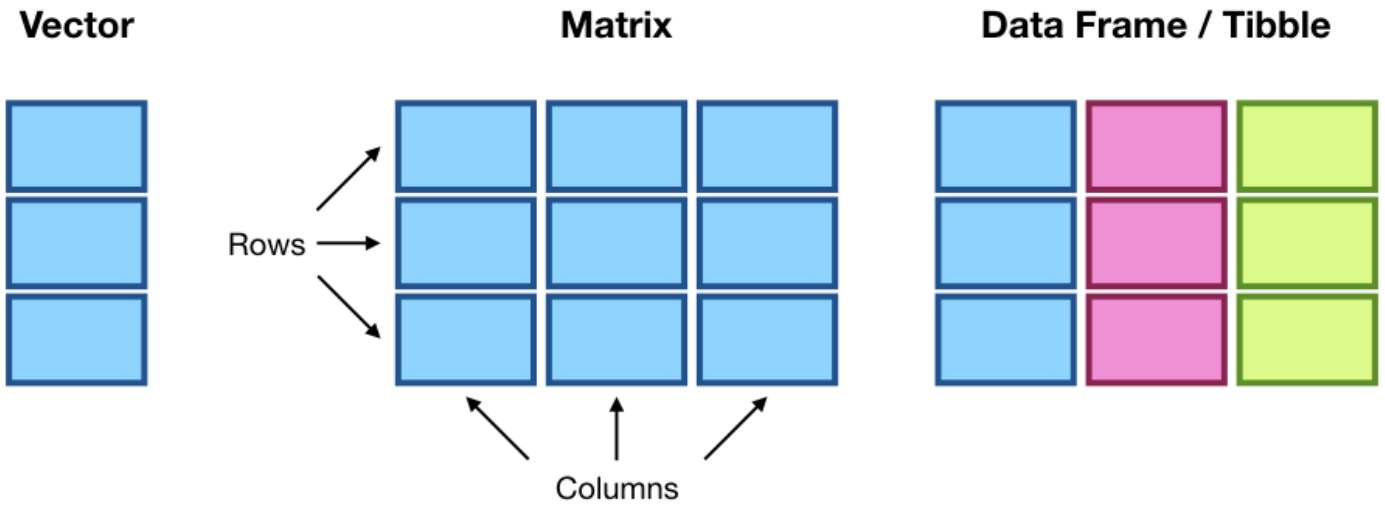
- `wang` dataset
- `crohns` dataset

General idea: First we play with some example data, and then you have to test you newly aquired expertise on real datasets!

Load the needed packages

For this lecture we will use both the tidyverse and `matrix`-objects

```
library(tidyverse)
```



Data Dimensionality

Data Dimensionality

An important concept in bioinformatics (and data science in general) is *data dimensionality*.

- **Formally**: The dimensionality of a dataset refers to the number of measurements for each sample/observation.
- **Informally**: The number of columns in a dataframe.

Let's look at some examples in R to get an intution about this!

Data Dimensionality

`Files/PCA/PCA_datasets.zip` in Absalon includes some example datasets as plain `.tab`-files. For each datasets there is a file with measurements, and a file with groupings for the data. We will use the grouping when we plot the data later in the class.

Data Dimensionality

`Files/PCA/PCA_datasets.zip` in Absalon includes some example datasets as plain `.tab`-files. For each datasets there is a file with measurements, and a file with groupings for the data. We will use the grouping when we plot the data later in the class.

They can be loaded into R using:

```
# Flowers dataset
flowers <- read_tsv("flowers_measurements.tab") %>%
    as.matrix
```

```
## Parsed with column specification:
## cols(
##   Sepal.Length = col_double(),
##   Sepal.Width = col_double(),
##   Petal.Length = col_double(),
##   Petal.Width = col_double()
## )
```

```
flowersGroups <- read_tsv("flowers_groups.tab")
```

```
## Parsed with column specification:
## cols(
##   Species = col_character()
## )
```

Data Dimensionality

Let's first look at the iris dataset, using methods that should be second nature to you at this point of the course:

```
# First few lines of the dataset
head(flowers)
```

```
##      Sepal.Length Sepal.Width Petal.Length Petal.Width
## [1,]          5.1         3.5          1.4         0.2
## [2,]          4.9         3.0          1.4         0.2
## [3,]          4.7         3.2          1.3         0.2
## [4,]          4.6         3.1          1.5         0.2
## [5,]          5.0         3.6          1.4         0.2
## [6,]          5.4         3.9          1.7         0.4
```

Data Dimensionality

```r
# Summary statistics
summary(flowers)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

Data Dimensionality

```r
# Dimensions of the dataframe
dim(flowers)
```

```
## [1] 150   4
```

The `flower` dataset consists of 150 flowers.

For each flower the Length and Width of Sepals and Petals have been measured.

This means that we have 4 measurements (2 length measures + 2 width measurements) for each observation (each flower)

The dimensionallity of the data is therefore **4**, or **4D**.

Data Dimensionality

Additional we have a known `grouping` or `class` for the flowers:

```
head(flowersGroups)
```

```
## # A tibble: 6 x 1
##   Species
##   <chr>
## 1 setosa
## 2 setosa
## 3 setosa
## 4 setosa
## 5 setosa
## 6 setosa
```

Data Dimensionality

Task 1:

Repeat the previous analysis for the `oil` dataset!

Data Dimensionality

Task 1:

Repeat the previous analysis for the `oil` dataset!

```
# Olive oil datasets
oil <- read_tsv("oil_measurements.tab") %>%
    as.matrix
```

```
## Parsed with column specification:
## cols(
##   palmitic = col_double(),
##   palmitoleic = col_double(),
##   stearic = col_double(),
##   oleic = col_double(),
##   linoleic = col_double(),
##   linolenic = col_double(),
##   arachidic = col_double(),
##   eicosenoic = col_double()
## )
```

```
oilGroups <- read_tsv("oil_groups.tab")
```

```
## Parsed with column specification:
## cols(
##   macro.area = col_character(),
```

Data Dimensionality

- Data dimensionality is important because it is inherently more difficult to intepret and analyse data with many dimensions, so-called *high-dimensional data*.
- This is because the human cognitive apparatus cannot visualize more than **3D** at a time, and written media (i.e. scientific publications) is limited to **2D**.
- Let's make some plots with R with the `flowers` dataset to illustrate this point...

```
# Now is the time to load your favourite plotting library!
library(GGally)
library(pheatmap)
```

Data Dimensionality

Setup the data for the examples:

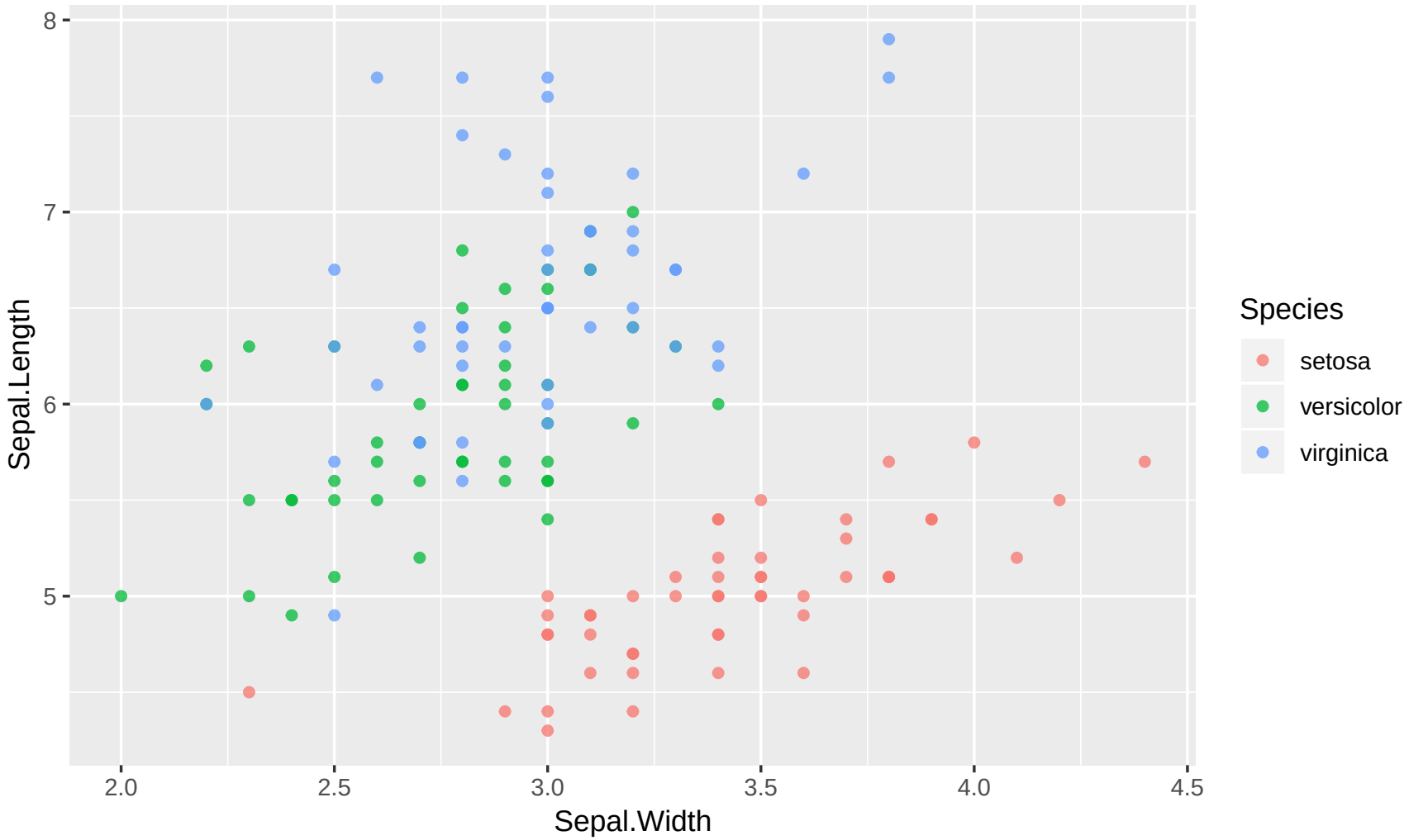
```
# Merge everything into one object
d <- as_tibble(flowers) %>%
    bind_cols(flowersGroups)

d
```

```
## # A tibble: 150 x 5
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##           <dbl>       <dbl>        <dbl>       <dbl> <chr>
##  1          5.1         3.5          1.4         0.2 setosa
##  2          4.9         3            1.4         0.2 setosa
##  3          4.7         3.2          1.3         0.2 setosa
##  4          4.6         3.1          1.5         0.2 setosa
##  5          5           3.6          1.4         0.2 setosa
##  6          5.4         3.9          1.7         0.4 setosa
##  7          4.6         3.4          1.4         0.3 setosa
##  8          5           3.4          1.5         0.2 setosa
##  9          4.4         2.9          1.4         0.2 setosa
## 10          4.9         3.1          1.5         0.1 setosa
## # ... with 140 more rows
```

Data Dimensionality

We can only plot **2D** at one time in a normal plot:

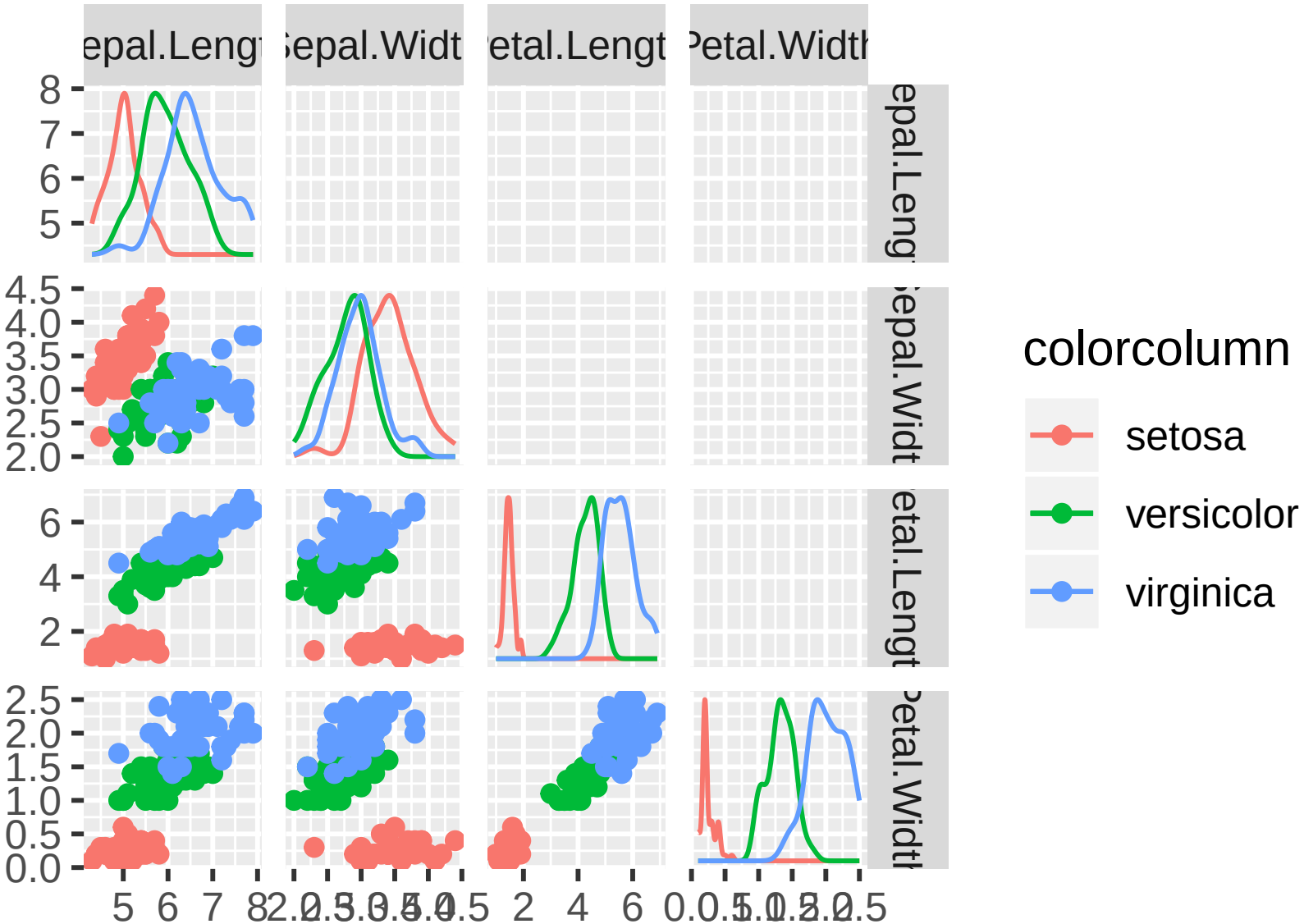
```
# Plot
ggplot(d, aes(x=Sepal.Width, y=Sepal.Length, color=Species)) +
    geom_point(alpha=0.75)
```



Data Dimensionality

Plotting all combinations of variables quickly becomes chaotic (This function is from the `GGally` package)

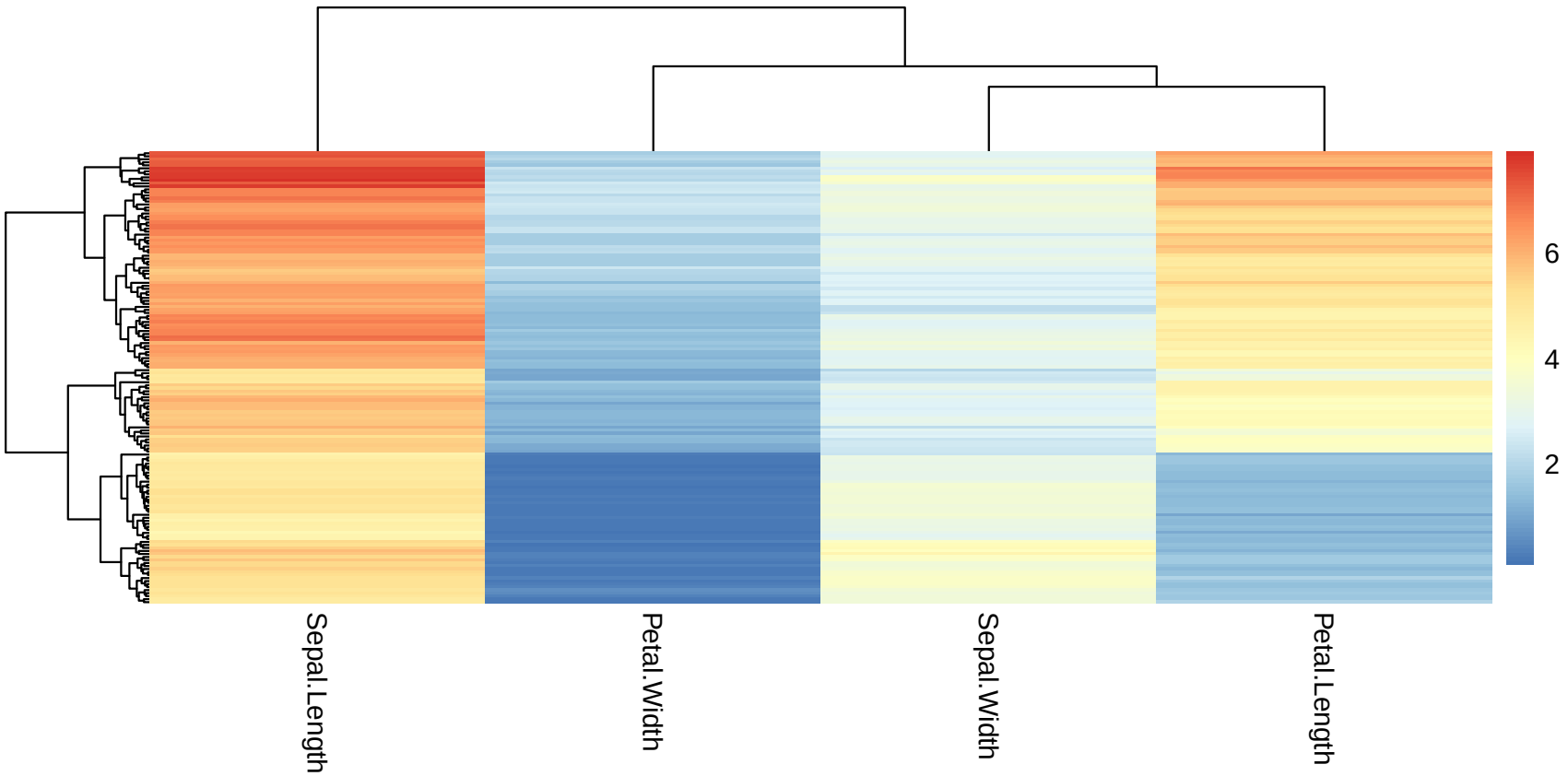
```
scatmat(d, color="Species")
```



Data Dimensionality

Heatmaps can visualize bigger datasets, but still become unpractical for very large datasets. (This function is from the `pheatmap` package)

```
pheatmap(flowers)
```



Data Dimensionality

- Already with only **4D**, visualizations quickly become messy.
- This finally brings us around to high throughput data:
  - **Microarrays**
  - **RNA-Seq**
  - **ChIP-Seq**
  - **CAGE**
  - etc...
- Tens of thousands of genes measured in each sample/library
- These datasets are *extremely* high dimensional $>$**1000D**!
- What can we do to solve this problem of dimensionality?

Dimensionality Reduction

Dimensionality Reduction

- We can use statistical methods to reduce the dimensionality of a dataset to allow for:
  - Low D plotting (which is what we will do)
  - Better performance of machine learning techinques ('The curse of dimensionality', not covered in this lecture)
- This reduction always comes at cost of loosing information, but is still tremendously useful.
- A wide range of methods, but we will focused on the most commonly used method called **Principal Component Analysis** or **PCA** for short.

Dimensionality Reduction

PCA in one line:

- *Transform a set of correlated variables into a lower number of uncorrelated variables, called Principal Components (PCs), preserving the maximum amount of variance.*

There are two part to this:

- Mathematical details of the method
- A good intuition about what is going on and how to use it effectively

Here will only deal with the second part (so no equations, hurray!)

Dimensionality Reduction

iPad demonstration:

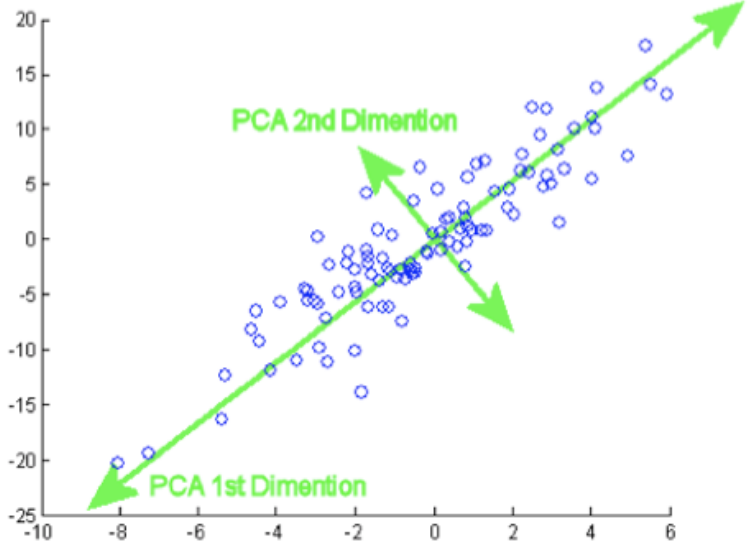
- You want sell your iPad on DBA (Den Blå Avis, the Danish version of eBay).
- You have to upload a picture of the iPad to lure in potential buyers. However, you can only upload single photo!
- Now you have a problem: How do you best depict the iPad, a 3D object, as a photo - a 2D representation?
- You actually have to perform dimensionality reduction on the iPad!

Dimensionality

How PCA would take the picture:

1) Find the longest axis through the iPad. This is the first PC!
2) Rotate the iPad around the first PC, until you again see most of the iPad. This is the second PC!

Dimensionality Reduction



Dimensionality Reduction

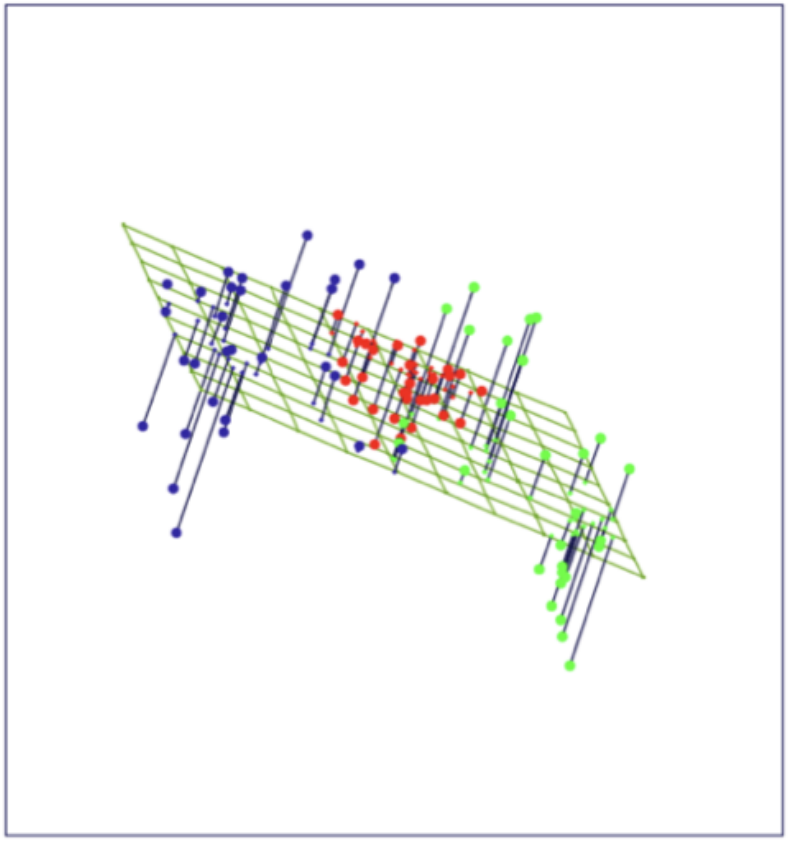
PCA in one line:

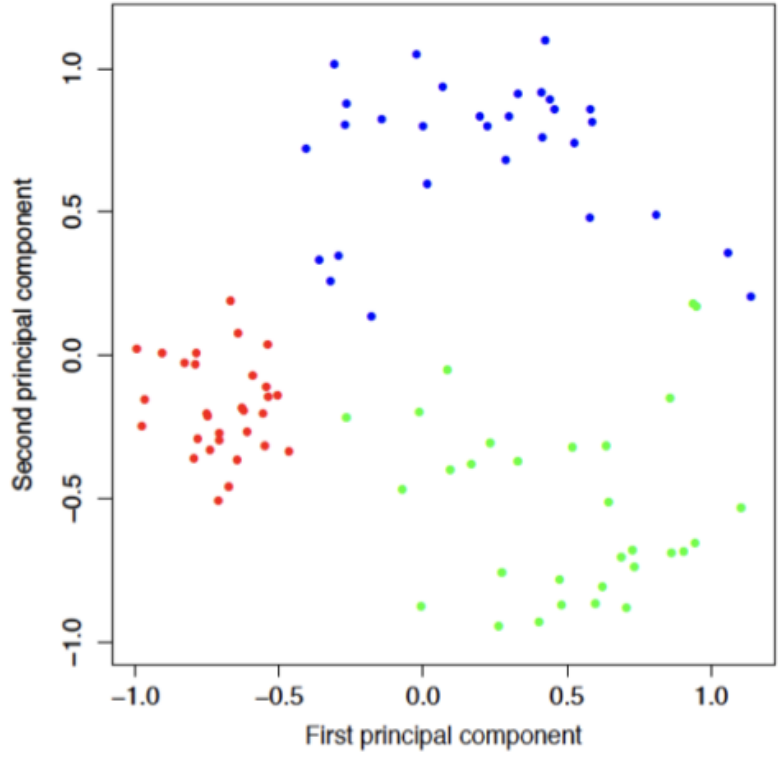
- *Transform a set of correlated variables into a lower number of uncorrelated variables, called Principal Components (PCs), preserving the maximum amount of variance.*

PCA is similar to taking a picture - except now what we want to "see" is the **variance** in the data:

1) Find the axis through the high-dimensional cloud of points covering the most variance: This is the first PC.
2) Perpendicularly to the first PC, find the axis covering the most amount of variance: This is the second PC.
3) Perpendicularly to the second PC, find the axis covering the most amount of variance: This is the third PC.
4) Continue until you run our of dimensions...

Dimensionality Reduction





Dimensionality Reduction

Let's try out PCA on the `flowers` dataset! As with everything in R, this is a one-liner:

```
pca_flowers <- prcomp(flowers)
```

That was easy! Let's look what's inside:

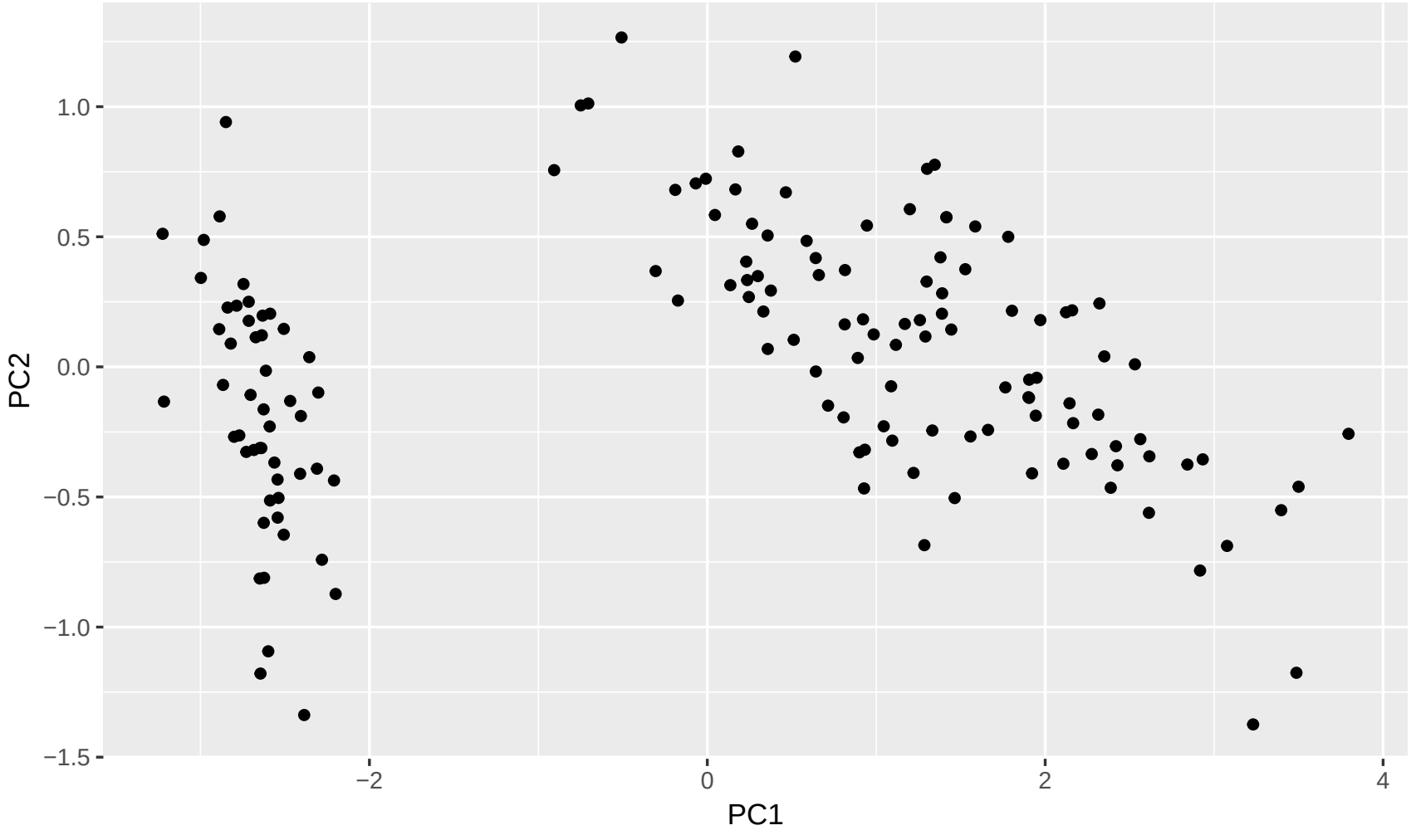
```
summary(pca_flowers)
```

```
## Importance of components:
##                           PC1     PC2    PC3     PC4
## Standard deviation     2.0563 0.49262 0.2797 0.15439
## Proportion of Variance 0.9246 0.05307 0.0171 0.00521
## Cumulative Proportion  0.9246 0.97769 0.9948 1.00000
```

Since the data is **4D** we get **4** PCs. The first PC captures **92%** of the variation in the dataset, while the second PC captures **5%**. Using just these two PCs, we can make a plot containing **97%** of the variation in the dataset! The positions of each observation along each PC (The position on the "picture") is stored in `pca_flowers$x` as a matrix object.

Dimensionality Reduction

```
ggplot(as_tibble(pca_flowers$x), aes(x=PC1, y=PC2)) + geom_point()
```



Dimensionality Reduction

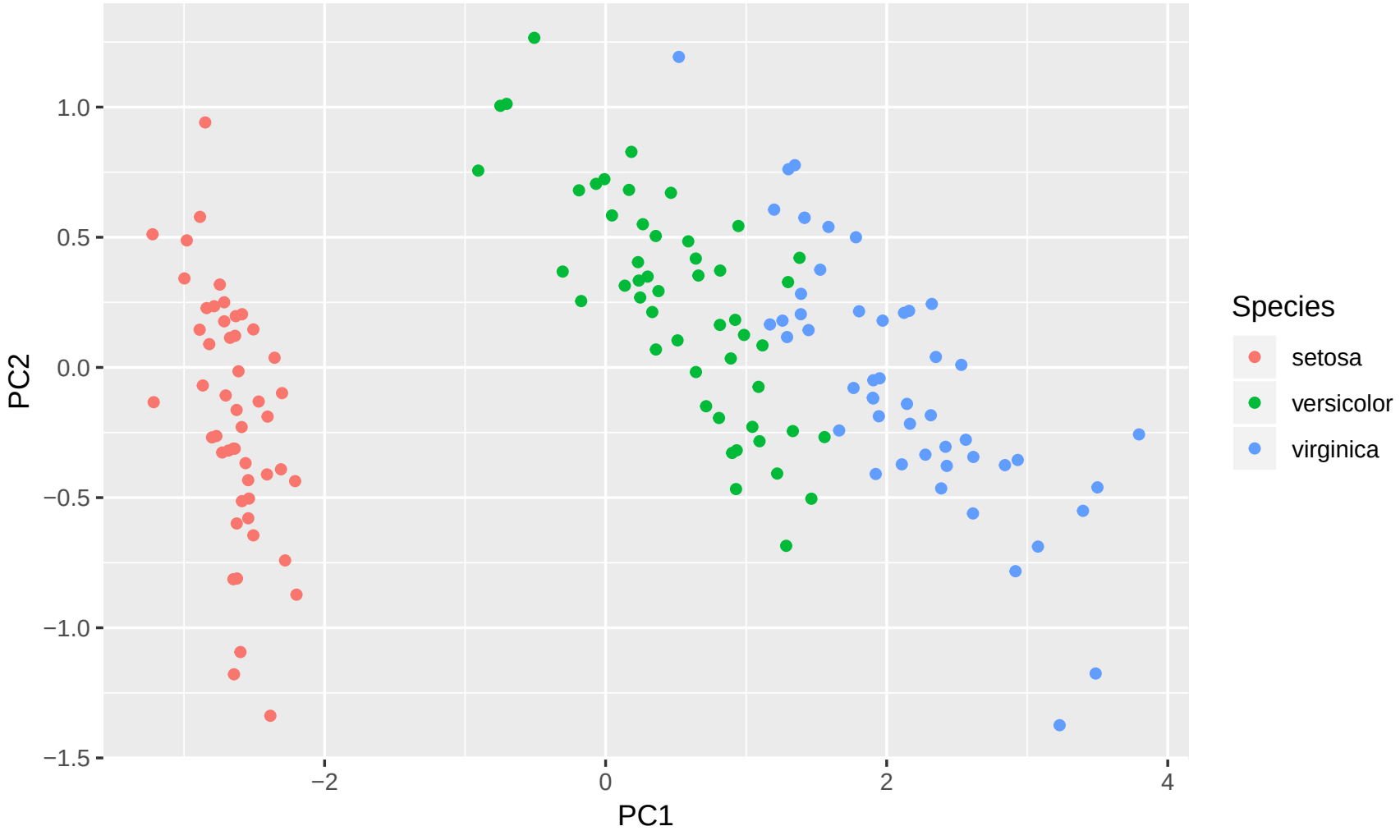
- Clearly there are some groups in the data. We have some additional information on samples groups, let's include this on the plot as colors.
- A small ggplot2 tip: Put all the data you need to plot in a single `tibble` - this makes plotting easier code-wise:

```
plot_flowers <- as_tibble(pca_flowers$x) %>%
    bind_cols(flowersGroups)

plot_flowers
```

```
## # A tibble: 150 x 5
##       PC1     PC2     PC3      PC4 Species
##     <dbl>   <dbl>   <dbl>    <dbl> <chr>
##  1 -2.68 -0.319   0.0279  0.00226 setosa
##  2 -2.71  0.177   0.210   0.0990  setosa
##  3 -2.89  0.145  -0.0179  0.0200  setosa
##  4 -2.75  0.318  -0.0316 -0.0756  setosa
##  5 -2.73 -0.327  -0.0901 -0.0613  setosa
##  6 -2.28 -0.741  -0.169  -0.0242  setosa
##  7 -2.82  0.0895 -0.258  -0.0481  setosa
##  8 -2.63 -0.163   0.0219 -0.0453  setosa
##  9 -2.89  0.578  -0.0208 -0.0267  setosa
## 10 -2.67  0.114   0.198  -0.0563  setosa
```

Dimensionality Reduction

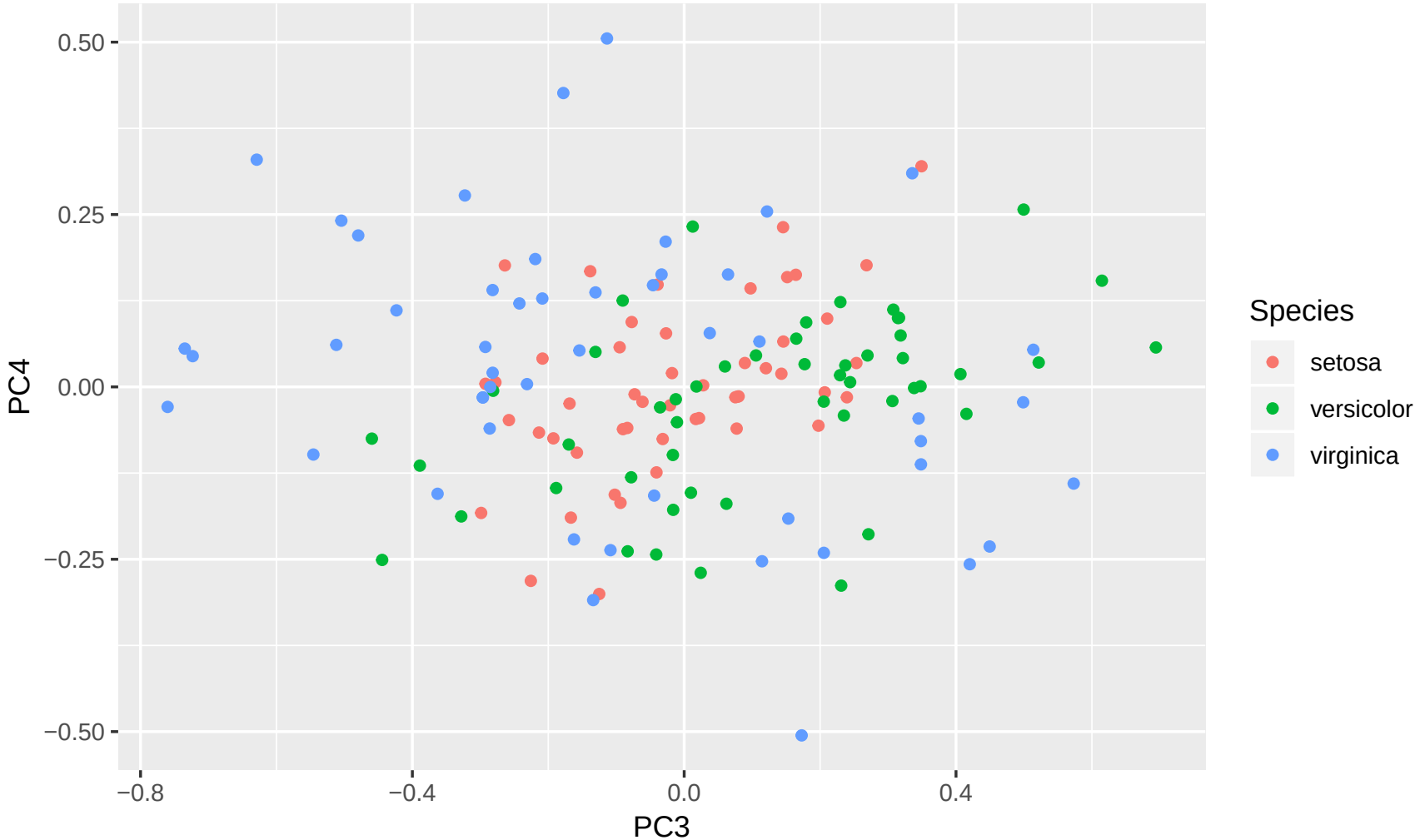
```
ggplot(plot_flowers, aes(x=PC1, y=PC2, color=Species)) + geom_point()
```



Dimensionality Reduction

We can also make a plot of the two remaining PCs. *Why does this look so bad?*

```
ggplot(plot_flowers, aes(x=PC3, y=PC4, color=Species)) + geom_point()
```



Task 2:

Repeat the previous analysis for the the `oil` dataset!

1. Perform the PCA
2. Inspect the amount of variance explained by each PC
3. Make a plot of the samples using `PC1` & `PC2` and `PC3` & `PC4`
4. Do you see any patterns?

Dimensionality Reduction

- An important part of PCA is **scaling** or **normalizing** the data
- PCA attempts to maximize variance in each PC, so if any input variables have higher variance than others, they will end up dominating the analysis.
- This is actually happening in the `oil` dataset:

```
apply(oil, 2, var)
```

```
##    palmitic palmitoleic     stearic       oleic    linoleic
##  28423.3515   2755.6583   1350.1903 164681.9365  58951.4616
##   linolenic   arachidic  eicosenoic
##    168.1871    485.3319    198.3392
```

- Due to the differences in measurements scales, `oleic` variance dominates.
- This can be corrected for by dividing each dimension by its standard deviation and substracting its mean:

```
apply(scale(oil), 2, var)
```

```
##    palmitic palmitoleic     stearic       oleic    linoleic
##           1           1           1           1           1
##   linolenic   arachidic  eicosenoic
##           1           1           1
```

Task 3:

Repeat the previous task with the `oil` dataset, but using the build in scaling feature with the `prcomp` function. Figure out how to do this by reading the `?prcomp` help file.

1. Compare the amount of variance explained by each PC for the scaled vs un-scaled data
2. Compare the pattern plot of between scaled vs un-scaled
3. Which PCA do you think is best - scaled or unscaled?

Applying PCA to real datasets

Applying PCA to real datasets

The `Wang` dataset:

- Dataset from the paper: *Alternative isoform regulation in human tissue transcriptomes* by Wang *et al*.
- URL to paper: http://www.ncbi.nlm.nih.gov/pubmed?term=18978772
- RNA-Seq data for **22** Human samples.
- Samples grouped by tissue:
  - Cell lines: 5 different subtypes.
  - Pure tissues: 9 different tissues.
  - Mixed tissues: 2 different mixes.
- The data contains log(CPM) expression for the most highly expressed genes.

Applying PCA to real datasets

Task 4:

- Load the dataset into R
- Determine the dimensionality of the dataset and the number of samples (Hint: What are the rows and what are the columns?)
- Perform PCA on the samples
- Determine how many PCs you need to explain at least 75% of the variance in the data
- Make the PCA plot and attempt to answer the following questions:
  - What is the general pattern observable in the data?
  - How many PCs are needed for capturing the important biological variation in the data?
  - Which type of `tissue` is most distinct? Which `cell.type`?
  - Which clusters do the mixed tissue samples belong to?
  - Biological question: Are cell lines a good model for tissues?

Applying PCA to real datasets

The `crohns` sdataset:

- Dataset from the paper: *Characterization of the enhancer and promoter landscape of inflammatory bowel disease from human colon biopsies* by Boyd *et al*.
- URL to paper: https://www.ncbi.nlm.nih.gov/pubmed/29695774
- Clinical Tissue Samples from Crohn's patients (Inflammatory Bowel Disease)
- CAGE data for 49 colon biopsies
- The tissue samples have been categorized by pathologists:
  - `crohns`: Crohn's disease with active inflammation
  - `con`: Healthy individuals
- The samples were prepared in four different groups, so-called batches, across a period of ~10 months.
- The data contains log(TPM-values) for the most highly expressed CAGE TSSs

Applying PCA to real datasets

Task 5:

- Load the dataset into R
- Determine the dimensionality of the dataset
- Perform PCA on the samples
- Determine how many PCs you need to explain at least 75% of the variance in the data
- Make the PCA plot and attempt to answer the following questions:
  - What is the general pattern observable in the data - does this correspond to anything you know about the labels?
  - Can you distinguish inflammed from non-inflammed samples?
  - Which patient group appears to have the largest intragroup variance?

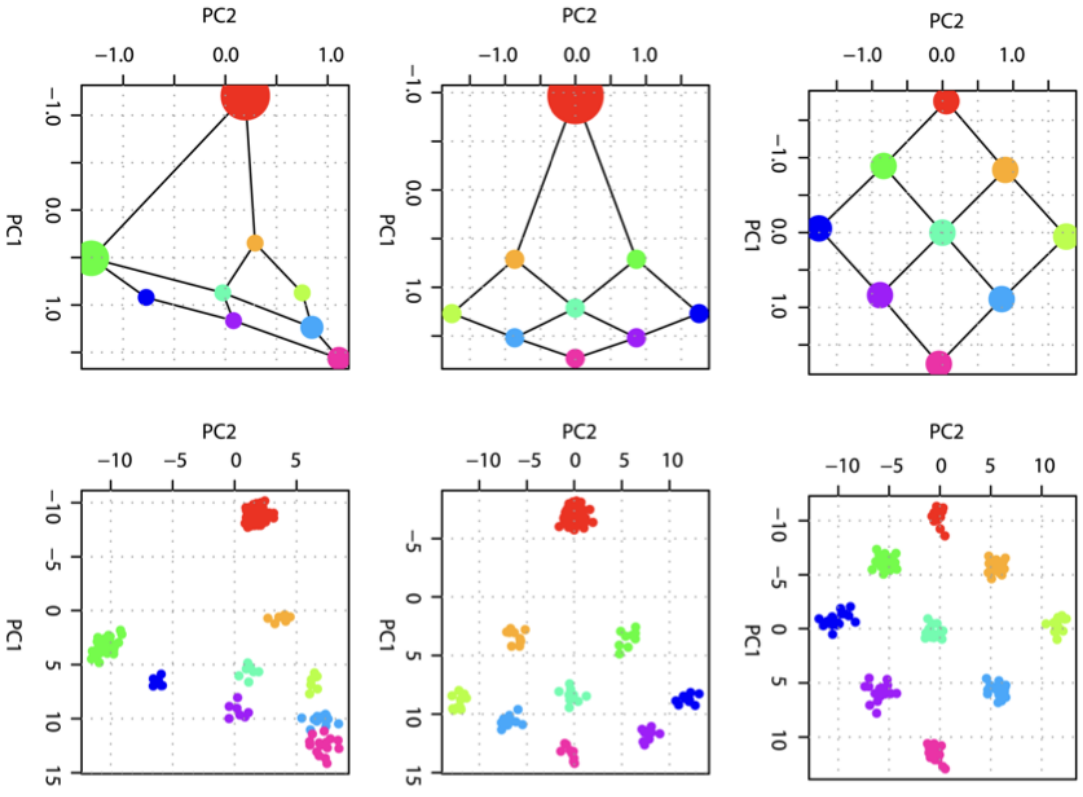
Perspectives

- PCA works because there is correlation in the original data - the total variance can be split into major patterns.
- Depending on the simplicity or complexity of the dataset PCA will be more or less effective.
- Returning the the photograph analogy:
  - What's the difference between taking pictures of these objects?
  - How does this relate to PCA?




Perspectives

- PCA is not a "natural" result from the data.
- The PCA plot will depend on how the data is prepared, subsetted, scaled, etc.
- As any other dimensionality reduction method, PCA has certain biases in the final visualization (i.e. sample size bias).



Relation to clustering

In many ways, dimensionality reduction techniques and clustering methods try to answer the same questions, i.e.:

- What are the major patterns in the data?
- What samples are similar/different?
- Are there any groupings?

Dimensionality reduction and clustering is not a matter of either/or, but should rather be seen as complimentary analyses!

Outliers - a field guide.

What happens when you see something unexpected?

- Multiple samples form a separate cluster: This is often called a *batch effect*
- A single sample appears distinct from all other: This is often called an *outlier*

Perhaps paradoxically, it is often easier to handle a batch effect than an outlier!

Outliers - how does an outlier arise?

A failed experiment:

- Failed RNA-extraction.
- Failed sequencing.
- etc.
- Sample will often look extremely different in ie. a PCA-plot.

True biological outlier:

- A single sample appears unique.
- At low sample sizes, might represent a rare subgroup.

Manual error:

- A sample has been mislabelled.
- The dreaded row-shift in excel due to copy-pasting.
- Mixup at the sequencing facility.
- Suprisingly **VERY** common!

Outliers - last year.

The question from last year:

- Should you never discard outliers?
- Should you always discard outliers?

The answer is as always:

- Statistics is a science, not a recipe.

Outliers - some guidelines.

How to handle outliers:

- If you very certain an experiment simply failed, the sample can be discarded.
- Re-check ALL manual steps, especially sample labelling.
- If the outlier still appears valid, consider the option that the experiment might be more complex than you expect - can you adjust your hypothesis?
- More advanced methods:
  - Perform a "weighted" or "robust" analysis
  - Outlier is included, but somehow "counts less" than the remaining samples.
  - Too advanced for this class.

Next lecture

See you on Friday for Differential Expression Analysis!