Visualization of Social Web Data

300958 Social Web Analytics

WESTERN SYDNEY
UNIVERSITY

School of Computing, Engineering and Mathematics

Week 5

Outline

High dimensional data sets

A data set usually consists of a set of records, where each record has measurements over a set of dimensions. So the number of measurements is the dimensionality of the data set.

For example, if the records are:

- people, the measurements may be height, weight, age, hair colour, etc. of each individuals.
- Expression of a gene in a biological sample. (>10,000 measurements)
- tweets, each measurement is the frequency of each known word. (>1,000,000 “measurements”)

It is difficult to visualise the similarity between each record when records are represented in a high dimensional space.

We will examine two ways to project these high dimensional records into a 2D space using *Principal Components Analysis* and *Multidimensional Scaling*

Outline

Dimension Reduction

When we take a photo, we project a three dimensional space (the scene) onto a two dimensional plane (a screen or a piece of paper). We want the two dimensional plane to capture the information from the 3D scene, but since we are projecting into a lower dimensional space, we lose information (e.g. what was behind the tree?).

Principal components analysis (PCA) computes where we should position our camera so that the photo contains the most information from the scene (the least information is lost from the projection).

Dimension Reduction

When we take a photo, we project a three dimensional space (the scene) onto a two dimensional plane (a screen or a piece of paper). We want the two dimensional plane to capture the information from the 3D scene, but since we are projecting into a lower dimensional space, we lose information (e.g. what was behind the tree?).

Principal components analysis (PCA) computes where we should position our camera so that the photo contains the most information from the scene (the least information is lost from the projection).

Principal Components Analysis

Given any data set with many dimensions, PCA computes the projection plane that preserves the most variance in the data.
We can then plot this projection to visualise the similarity between records.

2D Example

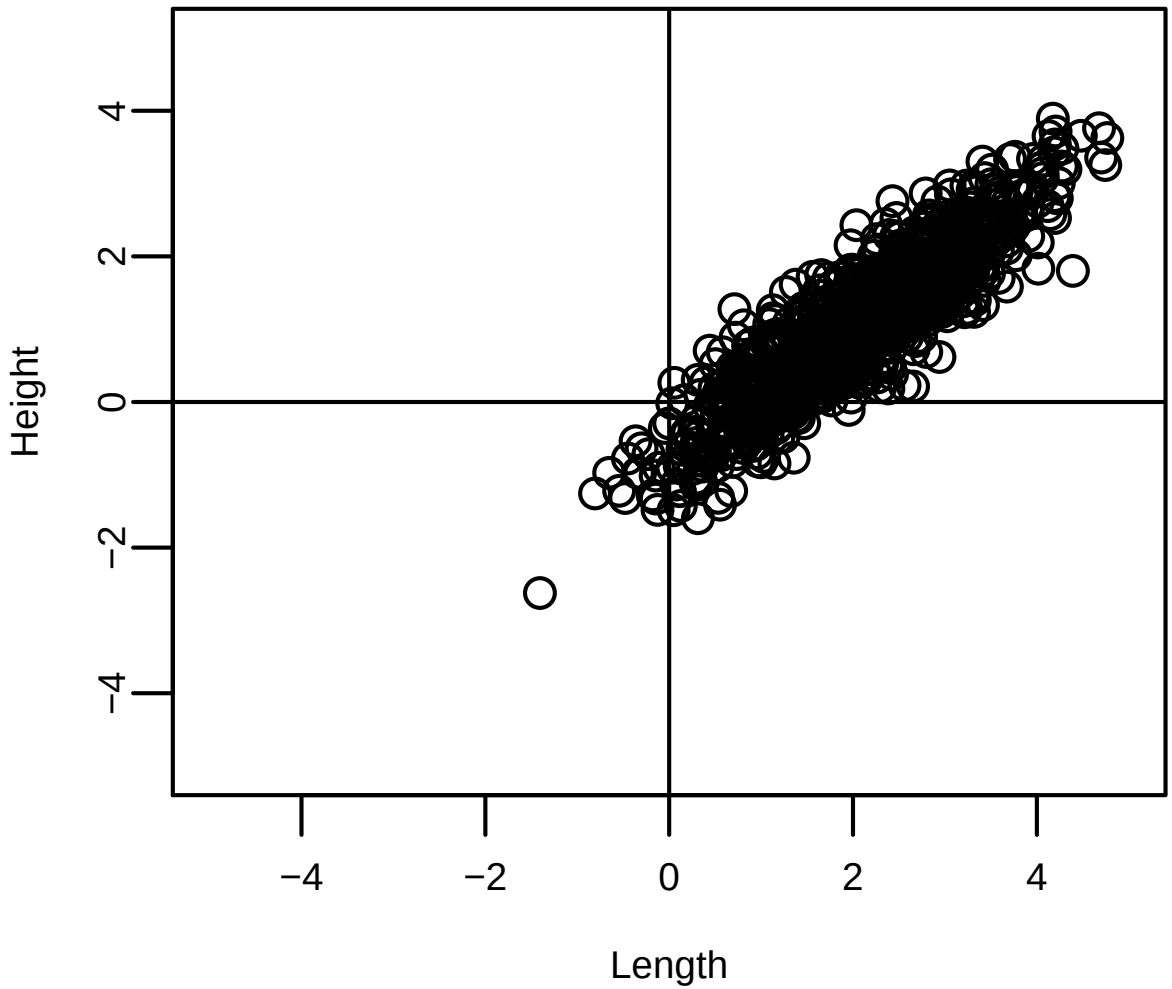


Figure: A plot of the height obtained and the length of a long jump.

2D Example: Axis of greatest variance

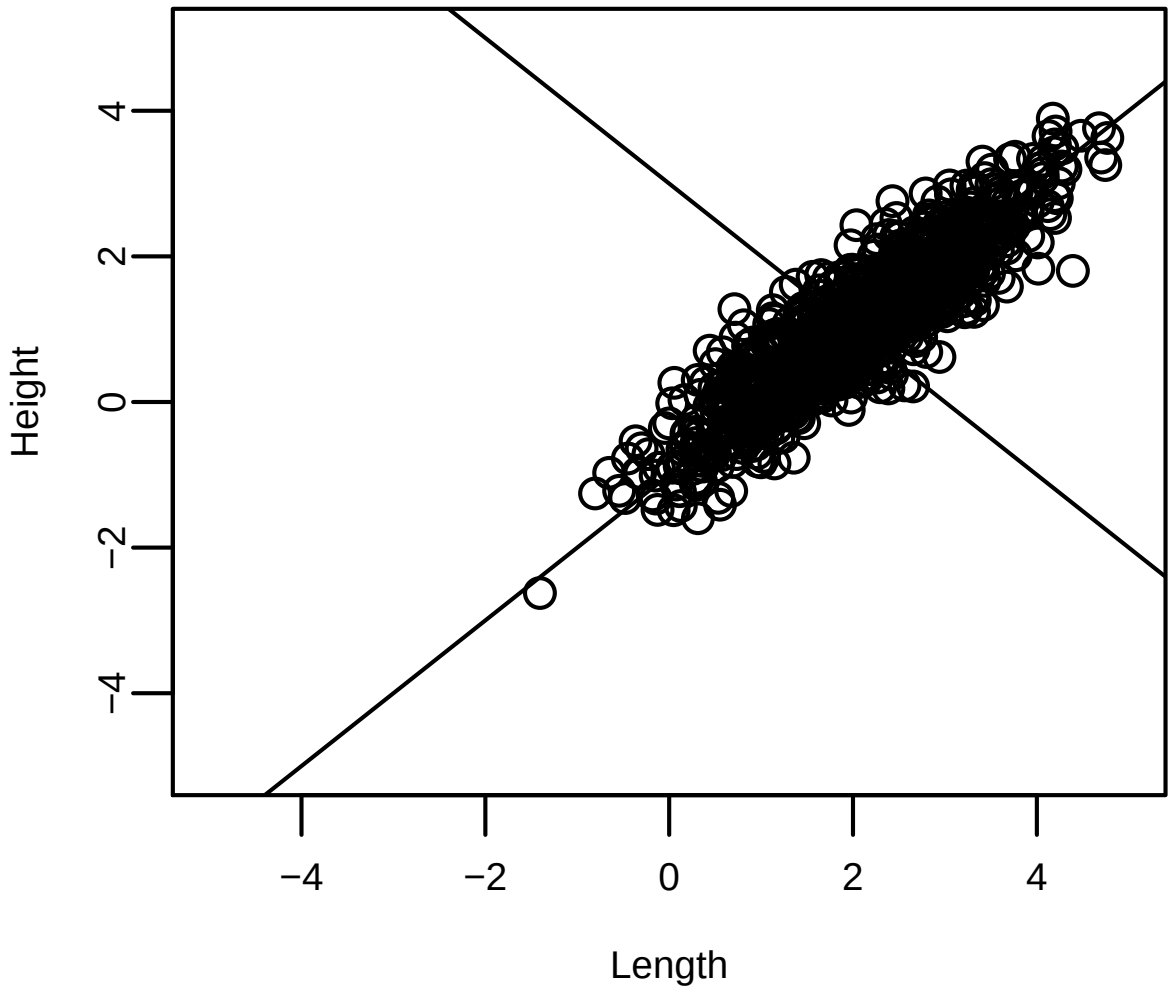


Figure: A plot of the height obtained and the length of a long jump, with alternate axis.

2D Example: Change of coordinates

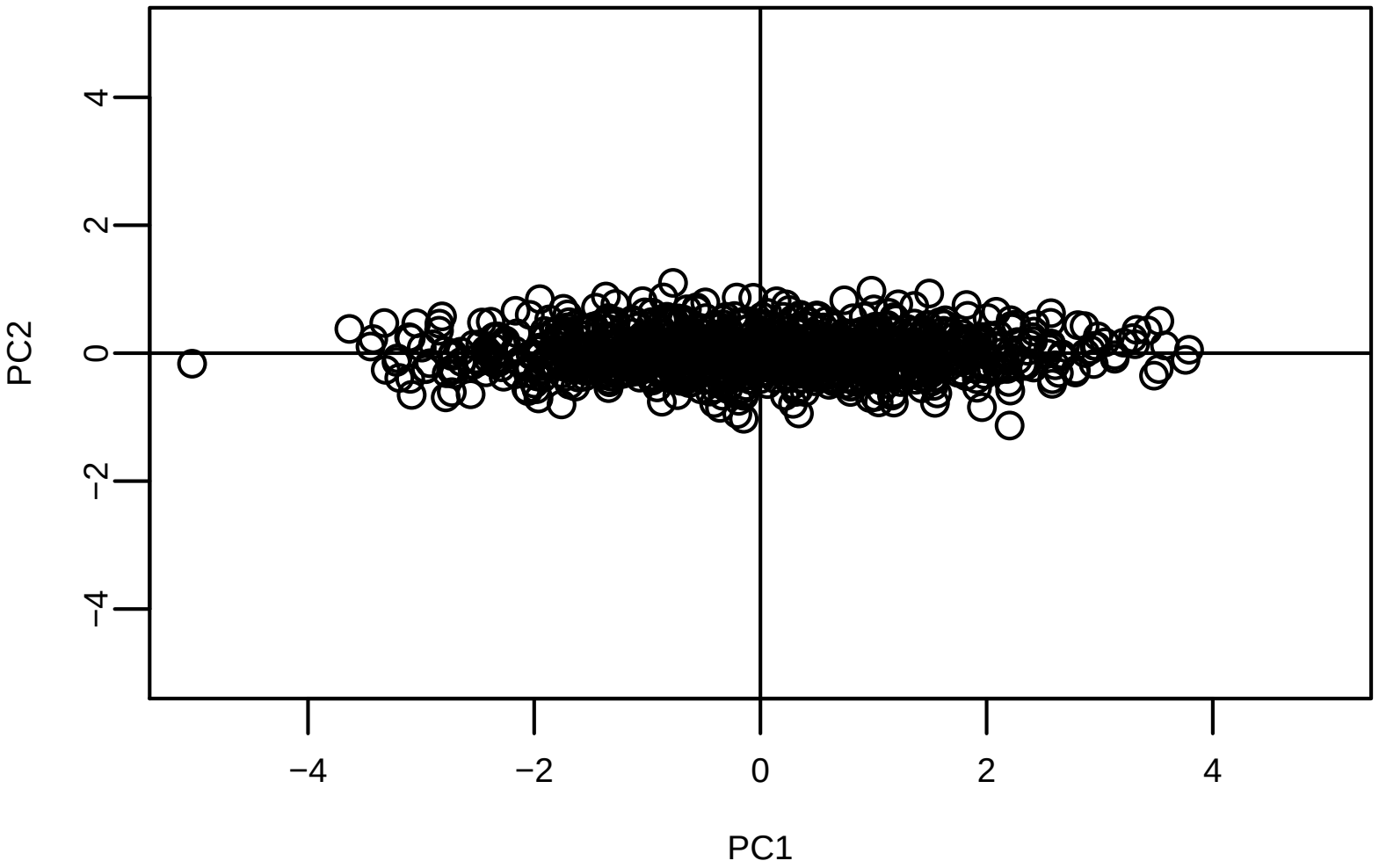


Figure: Length and height shown using Principal axis.

2D Example: Projection onto PC1

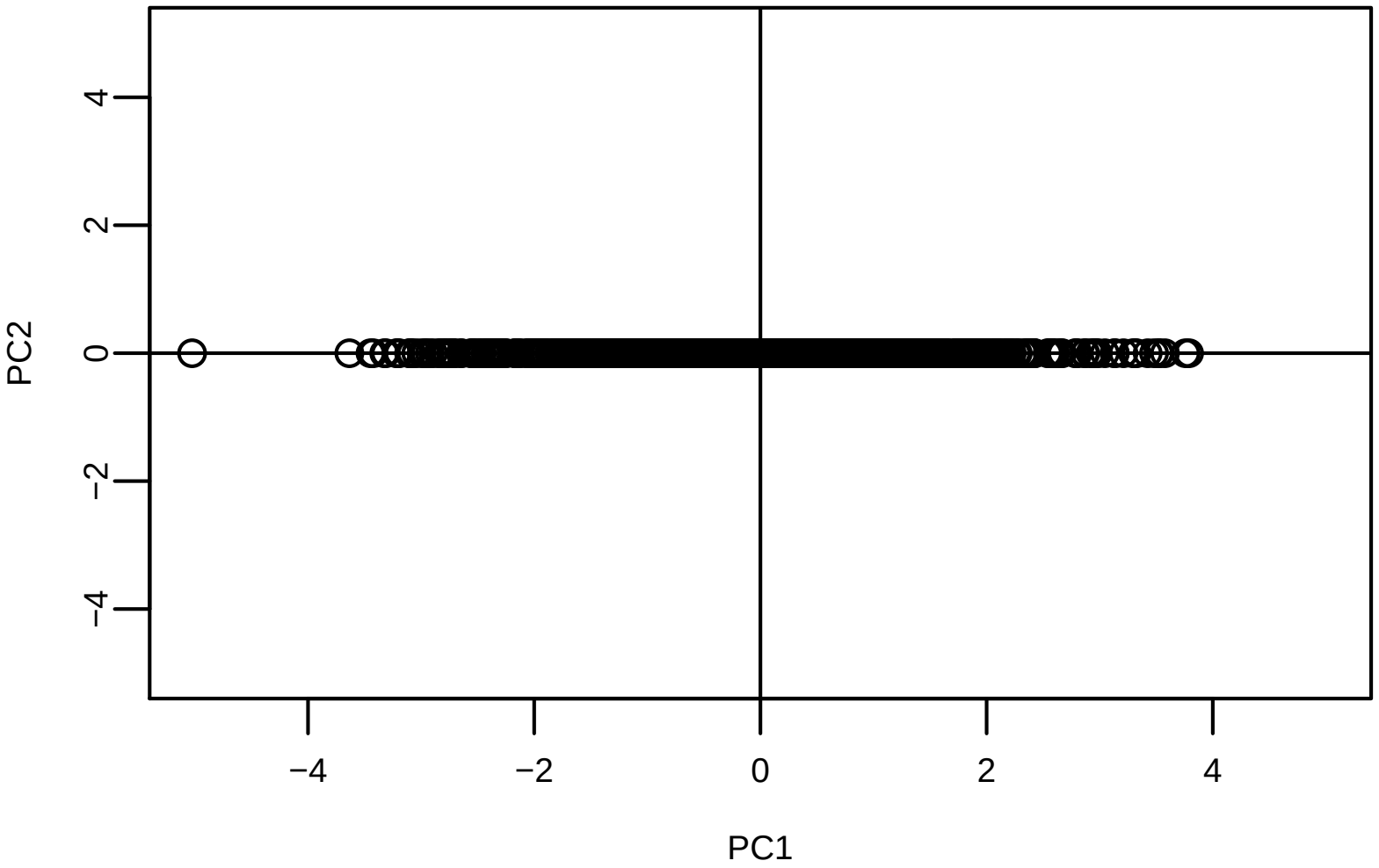


Figure: We retain most of the information using only the first principal axis.

Outline

Iris Data over four dimensions

In the previous example, we can visualise the original data and the projection. But if the data has more that three dimensions, we are unable to easily visualise the record relationships. Fisher's Iris data is four dimensional.

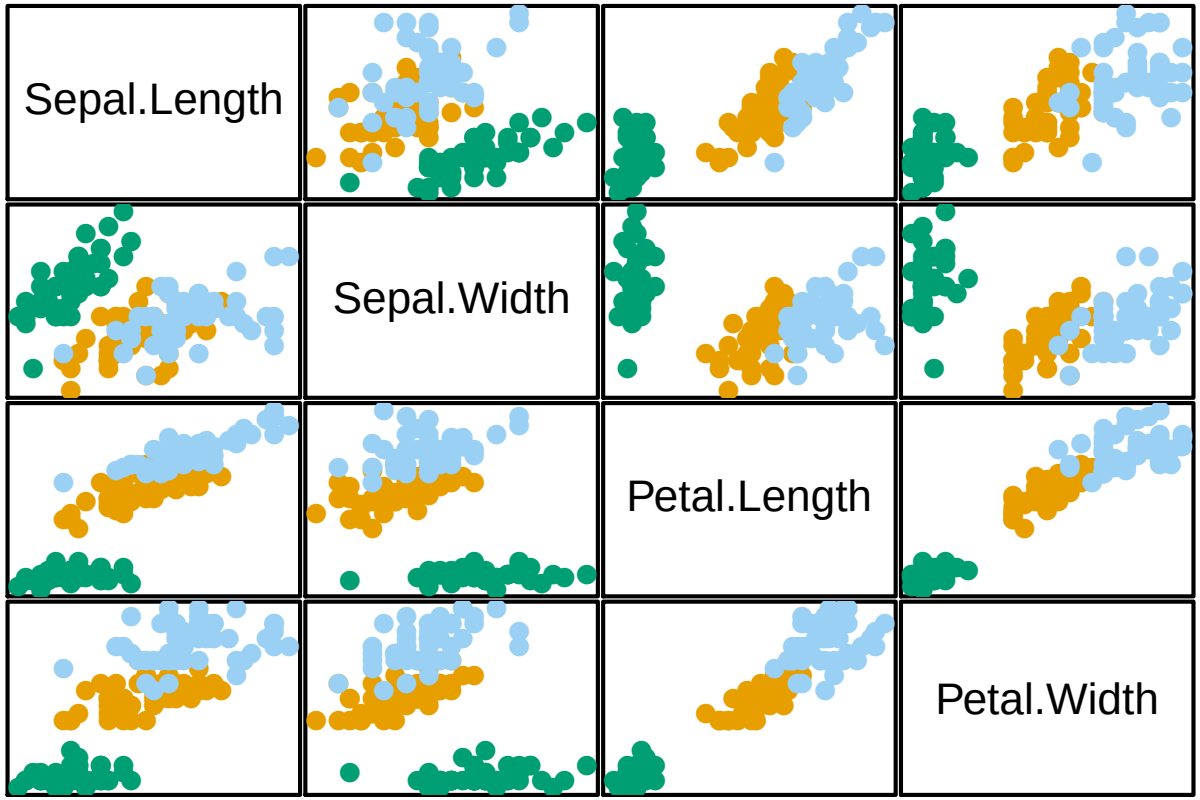


Figure: Pairwise plots of each dimension of the Iris data.

Principal Components Analysis

Each of the p variables has a variance and pairs of the variables may be correlated.

Principal components seeks a new variable Y_1 that is a linear combination of the original variables and has *maximum* variance.

$$Y_1 = a_1 X_1 + a_2 X_2 + \cdots + a_p X_p$$

Since $Var(aZ) = a^2 Var(Z)$ we must also restrict the a_i's so they are not arbitrarily big. Usually,

$$a_1^2 + a_2^2 + \cdots + a_p^2 = 1$$

Computing the Principal Components

The a_i's are actually computed using something called an Eigenvalue or Singular Value Decomposition.

- The a_i's above give the first principal component. But this is only one dimension.
- The *second* principal component is defined as the linear combination of the X_i's that maximises the variance

$$Y_2 = b_1 X_1 + b_2 X_2 + \cdots + b_p X_p$$

AND is *orthogonal* to the first principal component.

$$a_1 b_1 + a_2 b_2 + \cdots + a_p b_p = 0$$

The selected b_i's maximise the variance subject to the constraints that,

$$b_1^2 + b_2^2 + \cdots + b_p^2 = 1$$

Each further principal component is orthogonal to all previous principal components and maximise the remaining variance.

Principal Components in R


```
pca <- prcomp(X)
summary(pca)
```

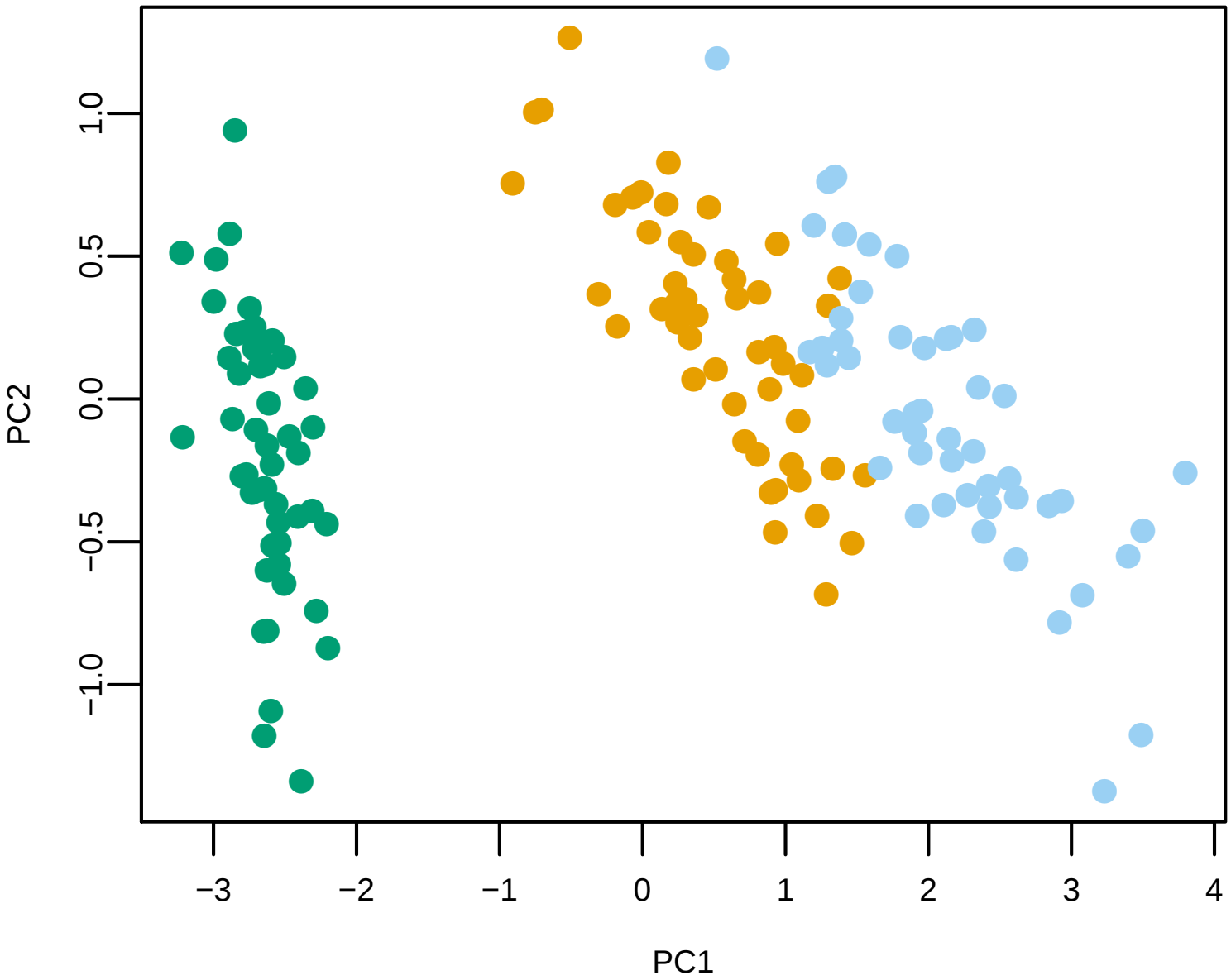
```
## Importance of components:
##                           PC1     PC2    PC3     PC4
## Standard deviation     2.0563 0.49262 0.2797 0.15439
## Proportion of Variance 0.9246 0.05307 0.0171 0.00521
## Cumulative Proportion  0.9246 0.97769 0.9948 1.00000
```

The summary shows the standard deviation of each of the 4 PCs for the iris data.

It also shows the percentage of *total variation* explained and the cumulative percentage.

The first two principal components here explain 97.8% of the total variation.

Principal Components projection of Iris



Outline

Creating the term-document matrix


```
library(twitteR)
### Search Twitter for some "relevant" tweets
tweets1 = searchTwitter("Anthony Albanese", n=500, lang="en")
tweets2 = searchTwitter("Scott Morrison", n=500, lang="en")
tweets3 = searchTwitter("Richard Di Natale", n=500, lang="en")
tweets = c(tweets1, tweets2, tweets3)
### Extract the tweet text
stext = sapply(tweets, function(x) x$getText())
library(tm)
### create a corpus and create document term matrix
corpus = Corpus(VectorSource(stext))
stoplist =   stopwords =c("anthony","albanese", "scott", "morrison",
                          "richard", "dinatale",
                          "ausvotes",  "auspol",
                          stopwords("english"))
tdm = TermDocumentMatrix(corpus,
        control = list(removePunctuation = TRUE,stopwords = stoplist,
                       removeNumbers = TRUE, tolower = TRUE))
### convert tdm to matrix
M = as.matrix(tdm)
```

Preparing term frequencies for PCA

This generates around 3000 terms on 1500 tweets.

The term-document matrix as above is terms (rows) by tweets (columns). R expects the *variables* to be in columns so we transpose.

Also, PCA works best for approximately Normally distributed data The term-document matrix is counts of each term (word) in each document (tweet). These frequencies are unlikely to be Normally distributed.

So we make a transformation; one possibility is TF-IDF but we find a square root works better here.

2D projection of Tweets


```
#cols = as.numeric(factor(cols))
```

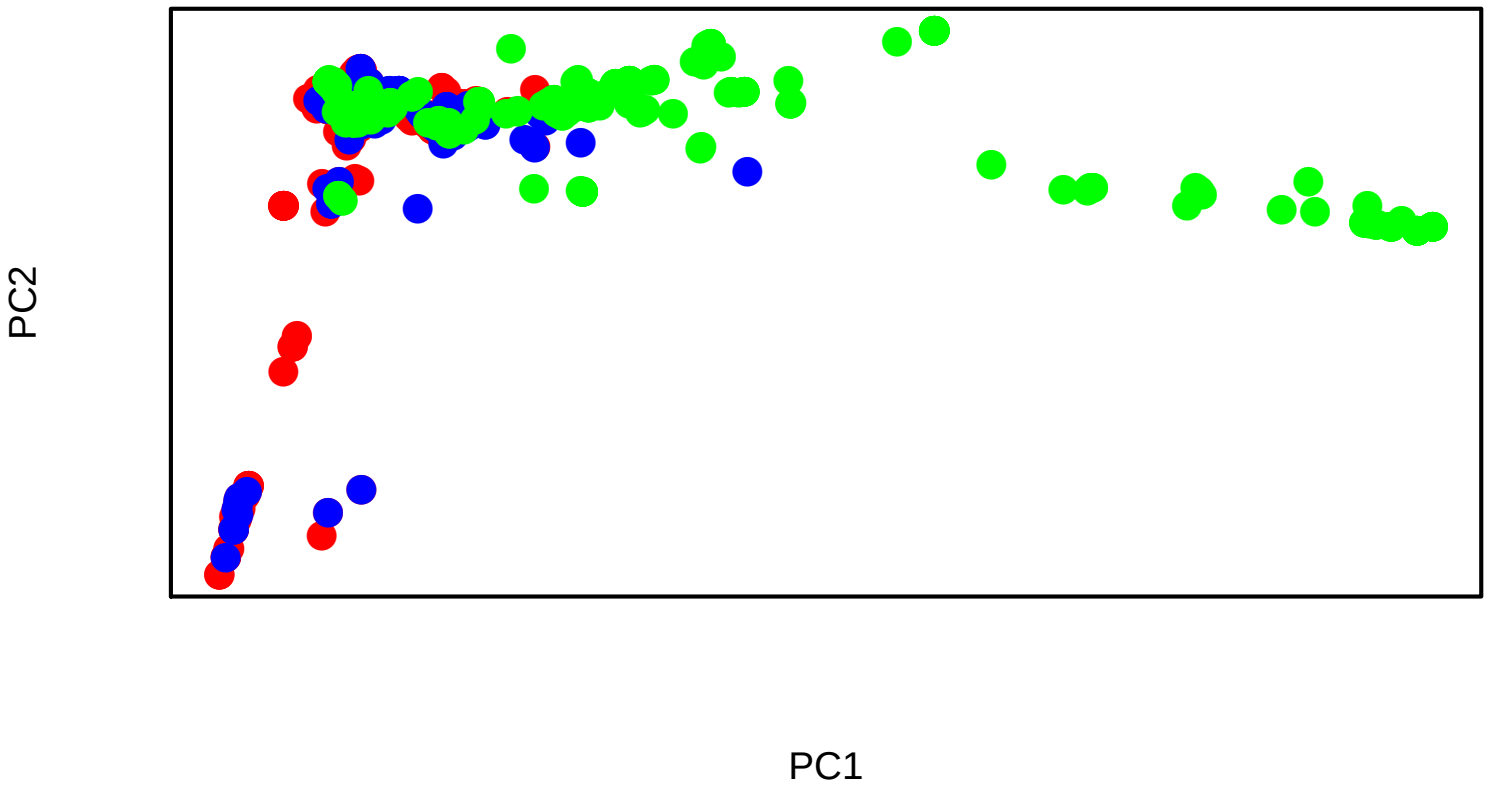


Figure: 2D projection of Tweets. Colours represent authors.

PCA summary of Tweet projection


```
summary(pca.tweet)$importance[,1:5]
```

```
##                             PC1       PC2       PC3       PC4       PC5
## Standard deviation     0.5763666 0.4668454 0.4462819 0.3863688 0.3620616
## Proportion of Variance 0.0355400 0.0233100 0.0213100 0.0159700 0.0140200
## Cumulative Proportion  0.0355400 0.0588500 0.0801600 0.0961300 0.1101500
```

Notice that even the first 5 principal components only explain 11% of the variation. In fact, 20 components only explain 25%.

Outline

Multidimensional scaling

Multidimensional scaling is a alternate way to view the dimension reduction problem.

If the data exists in p dimensions, and there is a distance defined between all pairs of observations/objects in the p dimensions. Can we construct a low dimensional (2D?) that has as near as possible the same distances in the low dimensional space?

Multidimensional scaling

Multidimensional scaling is a alternate way to view the dimension reduction problem.

If the data exists in p dimensions, and there is a distance defined between all pairs of observations/objects in the p dimensions. Can we construct a low dimensional (2D?) that has as near as possible the same distances in the low dimensional space?

Before we can do this, we will examine the properties of a distance function, and define:

- Euclidean Distance
- Manhattan Distance
- Maximum Distance
- Minkowski Distance
- Binary Distance
- Cosine Distance

Outline

Distance functions

A distance function (a metric) for vectors is a function that satisfies the following properties.

Metric Properties

A distance function for a set of vectors (vector space) satisfies:

1. $d(X, Y) \geq 0$: the distance is non-negative.
2. $d(X, Y) = 0 \Leftrightarrow X = Y$: if two vectors have zero distance, they must be the same vector.
3. $d(X, Y) = d(Y, X)$: the distance between X and Y is the same as between Y and X.
4. $d(X, Z) \leq d(X, Y) + d(Y, Z)$: the direct path is never longer than other paths.

Euclidean Distance

Euclidean distance (named after Euclid) is the distance of the straight line between two points (the distance we all know of). Useful when vectors represent coordinates.

$$d_{ij} = \sqrt{\sum_k (Y_{ik} - Y_{jk})^2}$$

Example

Given the two vectors $\vec{x} = [0\,1\,3\,0\,2], \vec{y} = [5\,0\,0\,0\,1]$

$$\begin{aligned} d_{ij} &= \sqrt{(0-5)^2 + (1-0)^2 + (3-0)^2 + (0-0)^2 + (2-1)^2} \\ &= \sqrt{25+1+9+0+1} = \sqrt{36} \\ &= 6 \end{aligned}$$

Manhattan Distance

Manhattan (Taxicab) distance provides the distance between two points when we are constrained to travel on the lines of grid (just as when driving through a city, we must drive on the roads).

$$d_{ij} = \sum_k |Y_{ik} - Y_{jk}|$$

Example

Given the two vectors $\vec{x} = [0\,1\,3\,0\,2], \vec{y} = [5\,0\,0\,0\,1]$

$$\begin{aligned} d_{ij} &= |0-5| + |1-0| + |3-0| + |0-0| + |2-1| \\ &= 5 + 1 + 3 + 0 + 1 \\ &= 10 \end{aligned}$$

Maximum Distance

Maximum (Chebyshev) distance returns the maximum distance between each pair of vector elements. Also known as the chessboard distance, since it provides the number of moves required to move a King between two points on a chessboard.

$$d_{ij} = \max_k |Y_{ik} - Y_{jk}|$$

Example

Given the two vectors $\vec{x} = [0\,1\,3\,0\,2], \vec{y} = [5\,0\,0\,0\,1]$

$$\begin{aligned} d_{ij} &= \max\left(|0-5|, |1-0|, |3-0|, |0-0|, |2-1|\right) \\ &= \max\left(5, 1, 3, 0, 1\right) \\ &= 5 \end{aligned}$$

Minkowski Distance

Minkowski distance is a generalisation of the previous distance functions.

$$d_{ij} = \left(\sum_k |Y_{ik} - Y_{jk}|^p \right)^{1/p}$$

Note that:

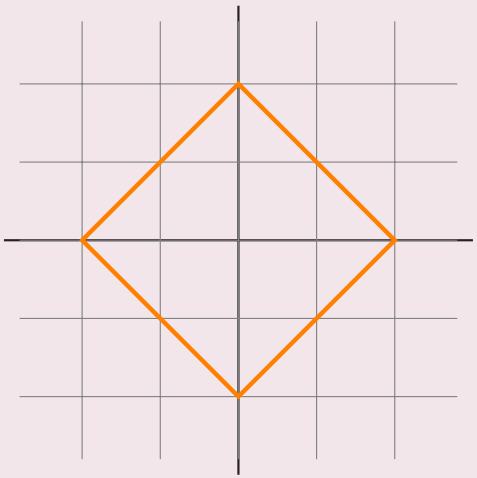
- If $p = 2$ we get Euclidean distance
- If $p = 1$ we get Manhattan distance
- If $p = \infty$ we get Maximum distance

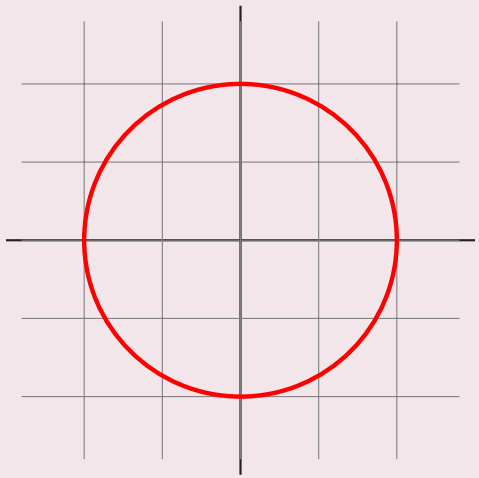
Visualising distances

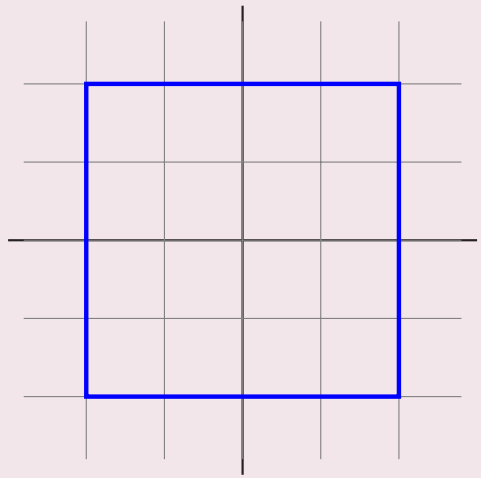
A circle consists of the set of all points that are distance r from the centre.

Problem

Which of the following are circles when using Euclidean distance, Manhattan distance, and Maximum distance?







Binary Distance

The binary distance treats elements of a vector as either present or absent and ignores the weight of the element.

$$d_{ij} = 1 - \frac{\sum_k (Y_{ik} > 0) \cap (Y_{jk} > 0)}{\sum_k (Y_{ik} > 0) \cup (Y_{jk} > 0)}$$

The distance is the number of elements that are non-zero in only one of the vectors divided by the number of elements that are non-zero in at least one vector.

Binary Distance

The binary distance treats elements of a vector as either present or absent and ignores the weight of the element.

$$d_{ij} = 1 - \frac{\sum_k (Y_{ik} > 0) \cap (Y_{jk} > 0)}{\sum_k (Y_{ik} > 0) \cup (Y_{jk} > 0)}$$

The distance is the number of elements that are non-zero in only one of the vectors divided by the number of elements that are non-zero in at least one vector.

Example

Given the two vectors $\vec{x} = [0\,1\,3\,0\,2], \vec{y} = [5\,0\,0\,0\,1]$

$$\begin{aligned} d_{ij} &= 1 - \frac{0+0+0+0+1}{1+1+1+0+1} \\ &= 1 - \frac{1}{4} = 0.75 \end{aligned}$$

Cosine distance

When examining document vectors (in a previous lecture), we showed that we compare them using the Cosine of weighted vectors. Cosine similarity is a similarity function, not a distance function.

- A high Cosine similarity score between documents implies that the two documents are similar.
- But small distances imply similarity.

We know that Cosine similarity has a maximum score of 1 when two documents vectors have the same direction. To convert Cosine similarity to a distance:

$$d(\text{doc}_i, \text{doc}_j) = 1 - \cos(\text{doc}_i, \text{doc}_j)$$

- if $\cos(\text{doc}_i, \text{doc}_j) = 1, d(\text{doc}_i, \text{doc}_j) = 0$
- if $\cos(\text{doc}_i, \text{doc}_j) < 1, d(\text{doc}_i, \text{doc}_j) > 0$

Cosine similarity and Euclidean distance

We can show the relationship between Euclidean distance and Cosine similarity:

$$
\begin{aligned}
d(X, Y)^2 &= \sum_k (X_k - Y_k)^2 \\
&= \sum_k X_k^2 + Y_k^2 - 2X_k Y_k \\
&= \sum_k X_k^2 + \sum_k Y_k^2 - 2\sum_k X_k Y_k \\
&= 1 + 1 - 2\cos(X, Y) \\
&= 2(1 - \cos(X, Y))
\end{aligned}
$$

if the vectors X and Y have length 1 ($\|X\| = 1$ and $\|Y\| = 1$).

Unit length document vectors

To adjust the length of each vector so that it satisfies $\|X\| = 1$, we divide by its length to obtain the scaled vector:

$$X = \frac{\vec{\text{doc}}_i}{\|\vec{\text{doc}}_i\|}$$

Note also that scaling a document vector changes the length but not the direction of the vector, so scaling a pair of vectors does not change their Cosine similarity (the angle between them).

Unit length document vectors

To adjust the length of each vector so that it satisfies $\|X\| = 1$, we divide by its length to obtain the scaled vector:

$$X = \frac{\vec{\text{doc}}_i}{\|\vec{\text{doc}}_i\|}$$

Note also that scaling a document vector changes the length but not the direction of the vector, so scaling a pair of vectors does not change their Cosine similarity (the angle between them).

Therefore we have the relationship between Cosine distance (1 - Cosine similarity) and Euclidean distance.

$$1 - \cos(X, Y) = d\left(\frac{X}{\|X\|}, \frac{Y}{\|Y\|}\right)^2 / 2$$

Outline

Distance Matrix

Once we have a distance function and our data (set of N vectors), MDS requires the $N \times N$ matrix of distances between all pairs of records in our data.

Problem

Given the three vectors $X_1 = [1\,2\,3], X_2 = [2\,0\,4], X_3 = [1\,0\,0]$, complete the distance matrix, where distance is measured using Manhattan distance:

$$D = \begin{bmatrix} & 4 & 5 \\ & & 5 \\ & & \end{bmatrix}$$

Multidimensional scaling

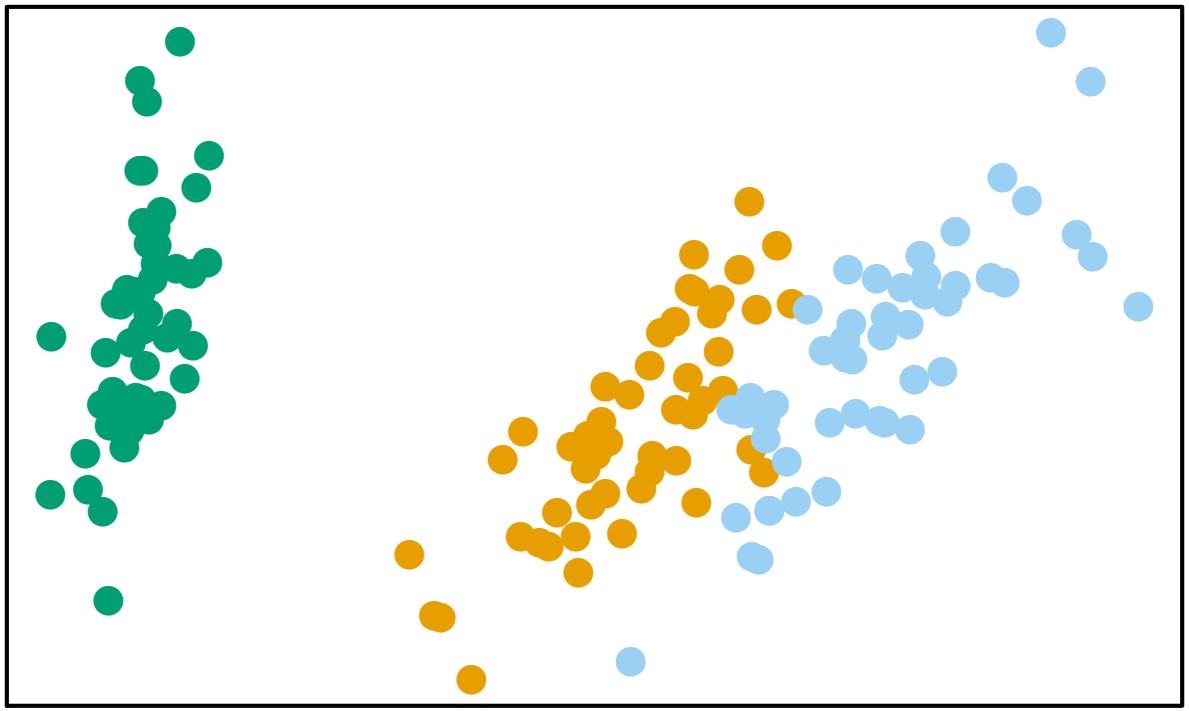
If we want to visualise the data in 2D, the process of Multidimensional scaling asks "which set of 2D vectors provide the matrix D (or most similar matrix D), when using Euclidean Distance?"

- MDS computes the set of vectors in the lower dimensional space that best preserve the distances of the original data.
- MDS computes distances of the lower dimensional vectors using Euclidean distance to provide the best visualisation of distances.

Multidimensional scaling - Iris Data

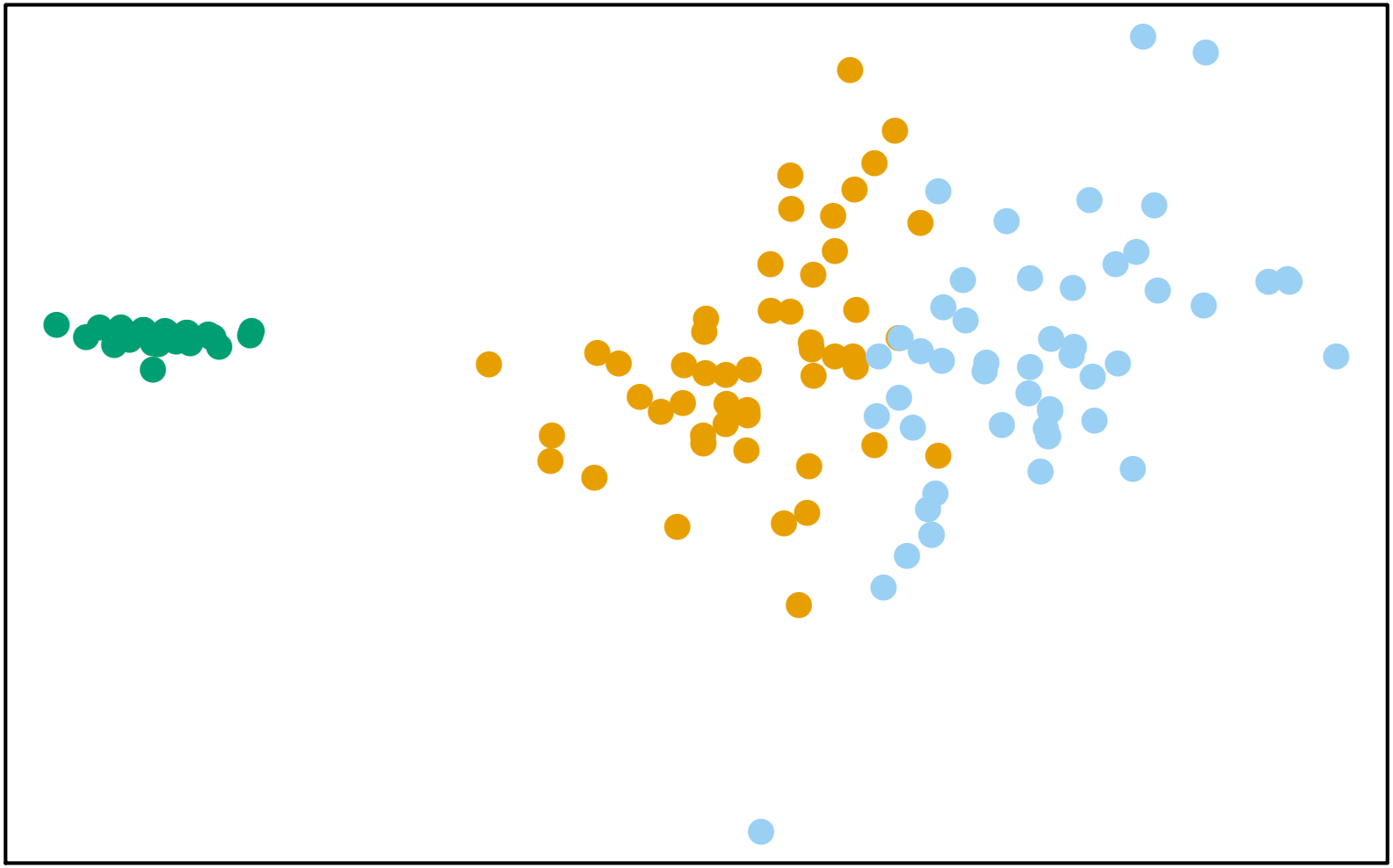
The following plot looks like the PCA plot for the Iris data. When MDS is used on data that measures distances with Euclidean distance, PCA and MDS are equivalent.

```
D <- dist(X) # compute distance matrix on X using Euclidean distance.
mds <- cmdscale(D, k=2) # MDS given distances D, projected to 2D
```



Multidimensional scaling - Iris Data


```
D <- dist(X, method="maximum")
mds <- cmdscale(D, k=2)
```



Outline

Examining the effect of each distance function

In this section we will apply MDS to the set of 1498 tweets from:

- Anthony Albanese (ALP, shown in red)
- Scott Morrison (Liberal Party, shown in blue)
- Richard Di Natale (Greens, shown in green)

After preprocessing, we are left with 3266 unique words appearing in the set of tweets, so each tweet is represented as a 3266 dimensional vector. We will use MDS to project the 3266 dimensional space into two dimensions.

TF-IDF is applied to each of the document vectors, then we use MDS with the chosen distance function.

MDS: TF-IDF with Euclidean Distance


```
M2 <- as.matrix(weightTfIdf(tdm))
D <- dist(t(M2), method="euclidean")
mds.tweet <- cmdscale(D, k=2)
```

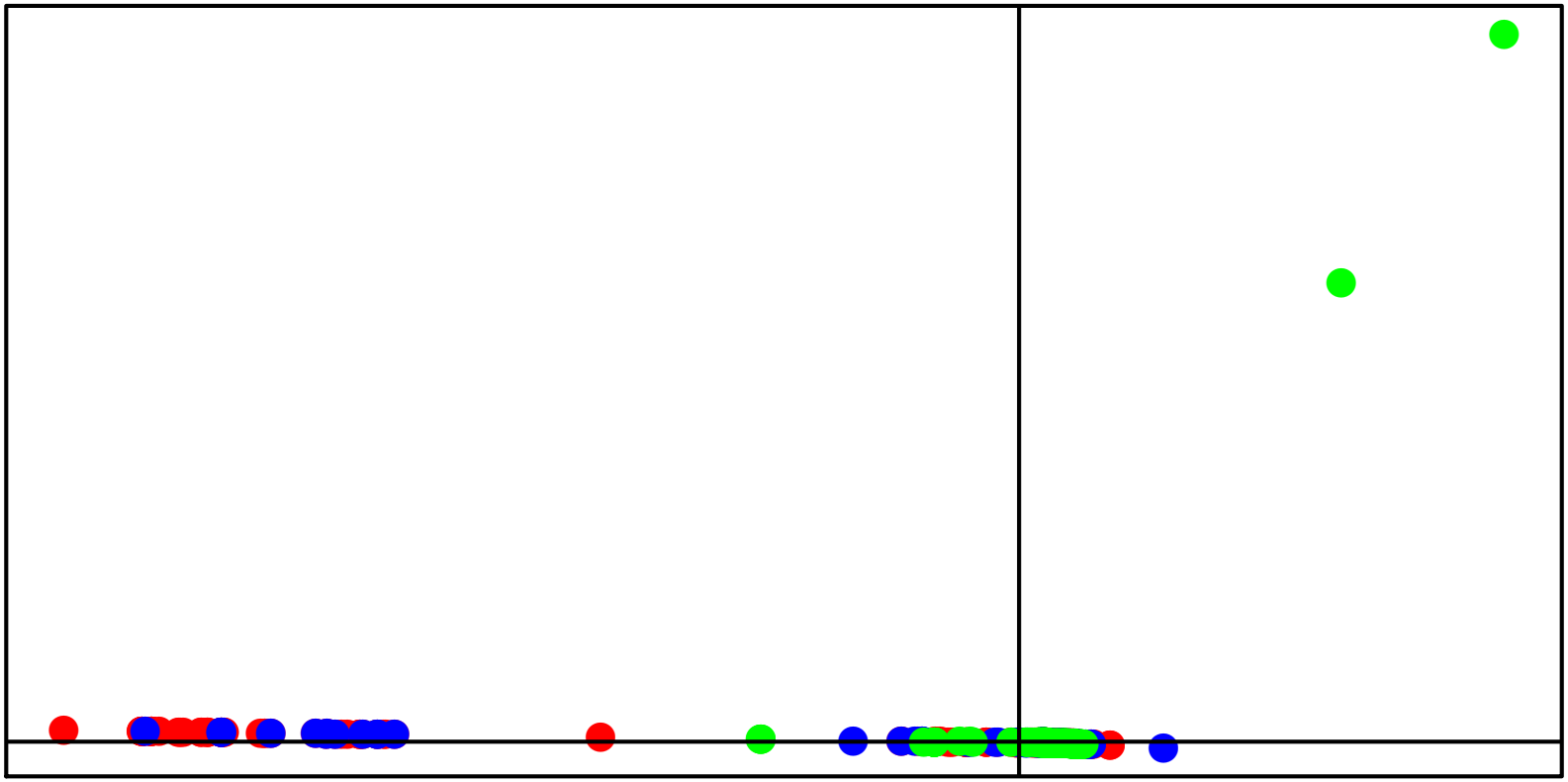


Figure: Similarity of tweets measured using Euclidean Distance

MDS: TF-IDF with Manhattan Distance


```
M2 <- as.matrix(weightTfIdf(tdm))
D <- dist(t(M2), method="manhattan")
mds.tweet <- cmdscale(D, k=2)
```

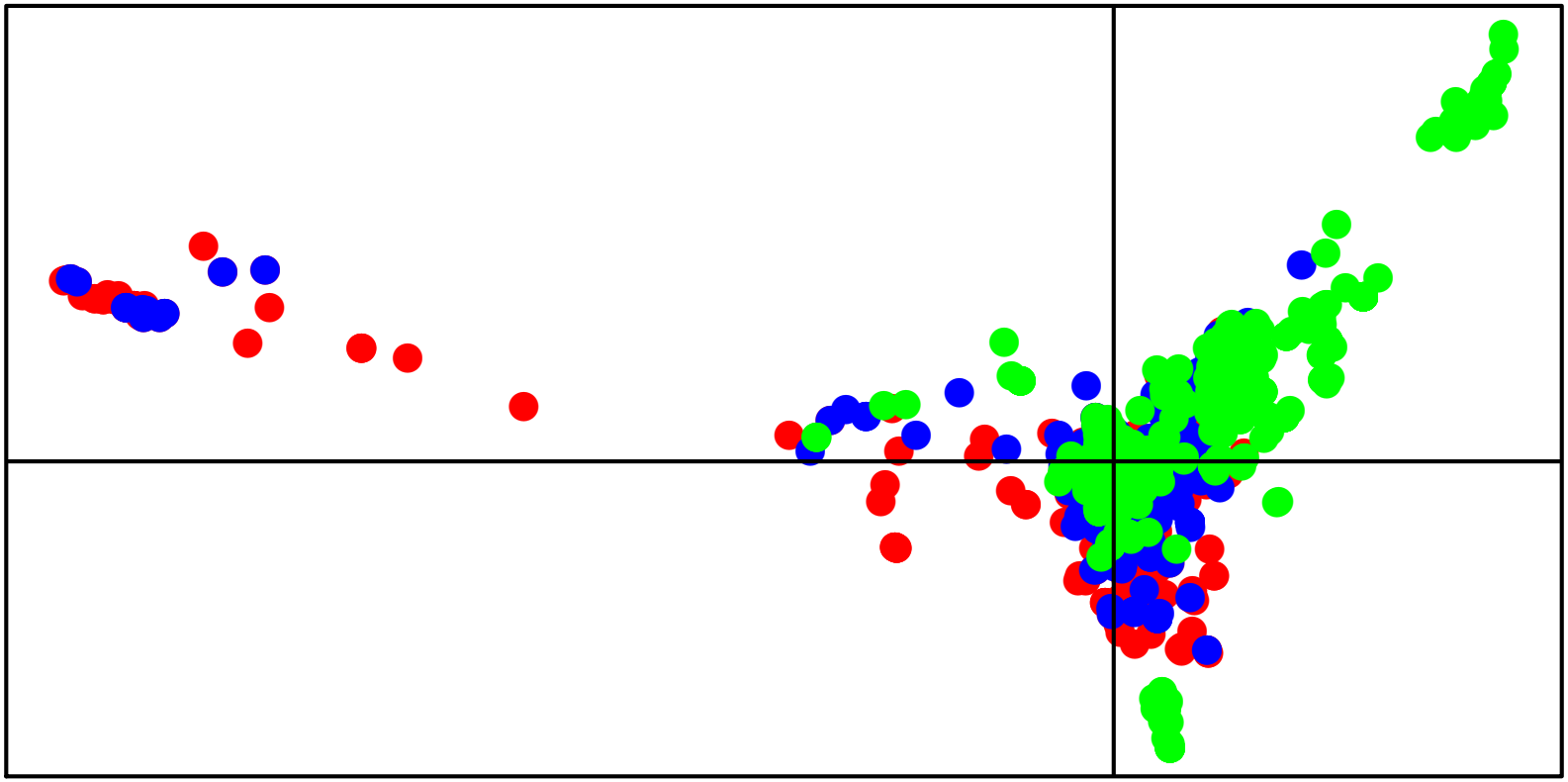


Figure: Similarity of tweets measured using Manhattan Distance

MDS: TF-IDF with Maximum Distance


```
M2 <- as.matrix(weightTfIdf(tdm))
D <- dist(t(M2), method="maximum")
mds.tweet <- cmdscale(D, k=2)
```

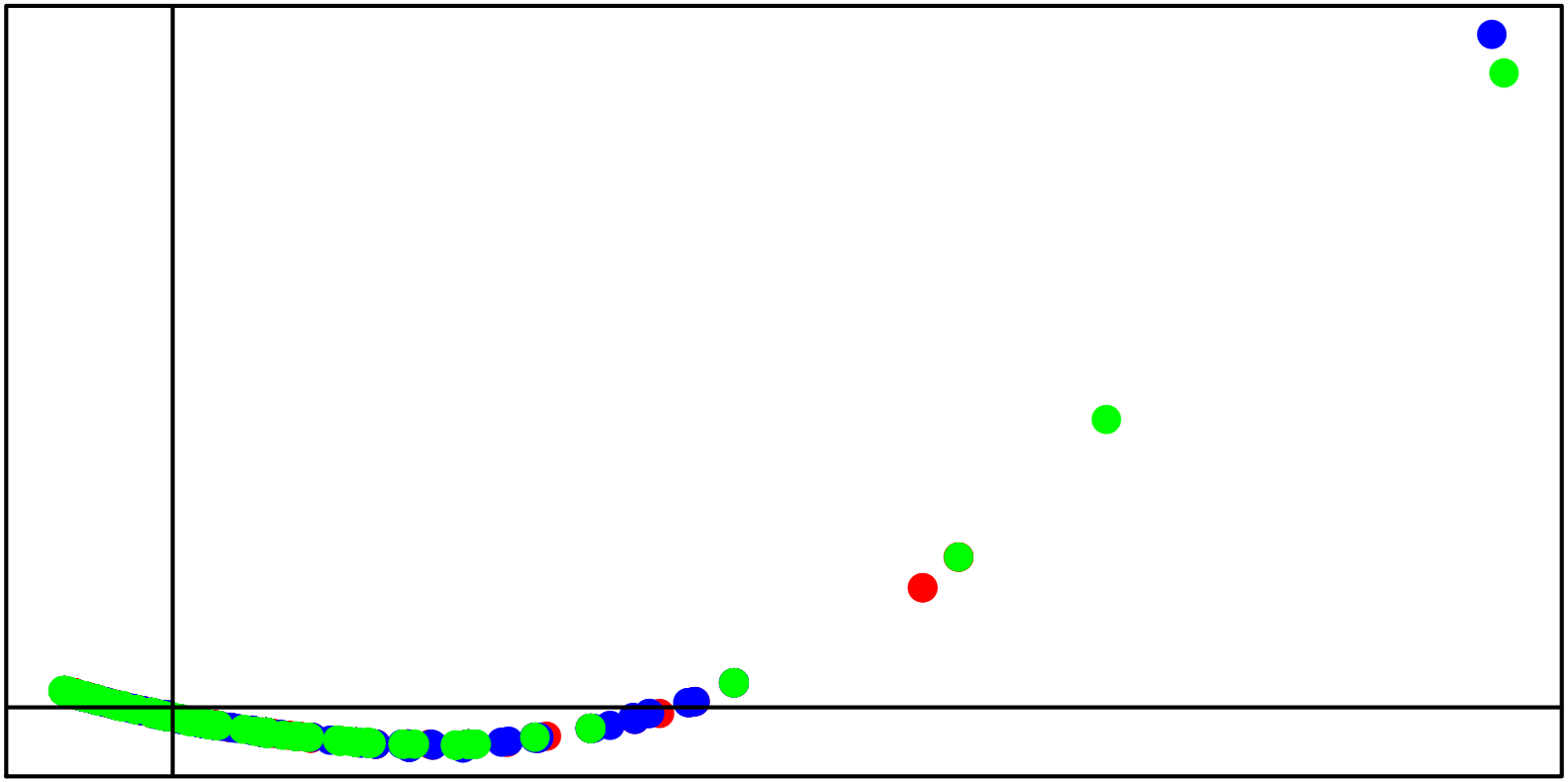


Figure: Similarity of tweets measured using Maximum Distance

MDS: TF-IDF with Binary Distance


```
M2 <- as.matrix(weightTfIdf(tdm))
D <- dist(t(M2), method="binary")
mds.tweet <- cmdscale(D, k=2)
```

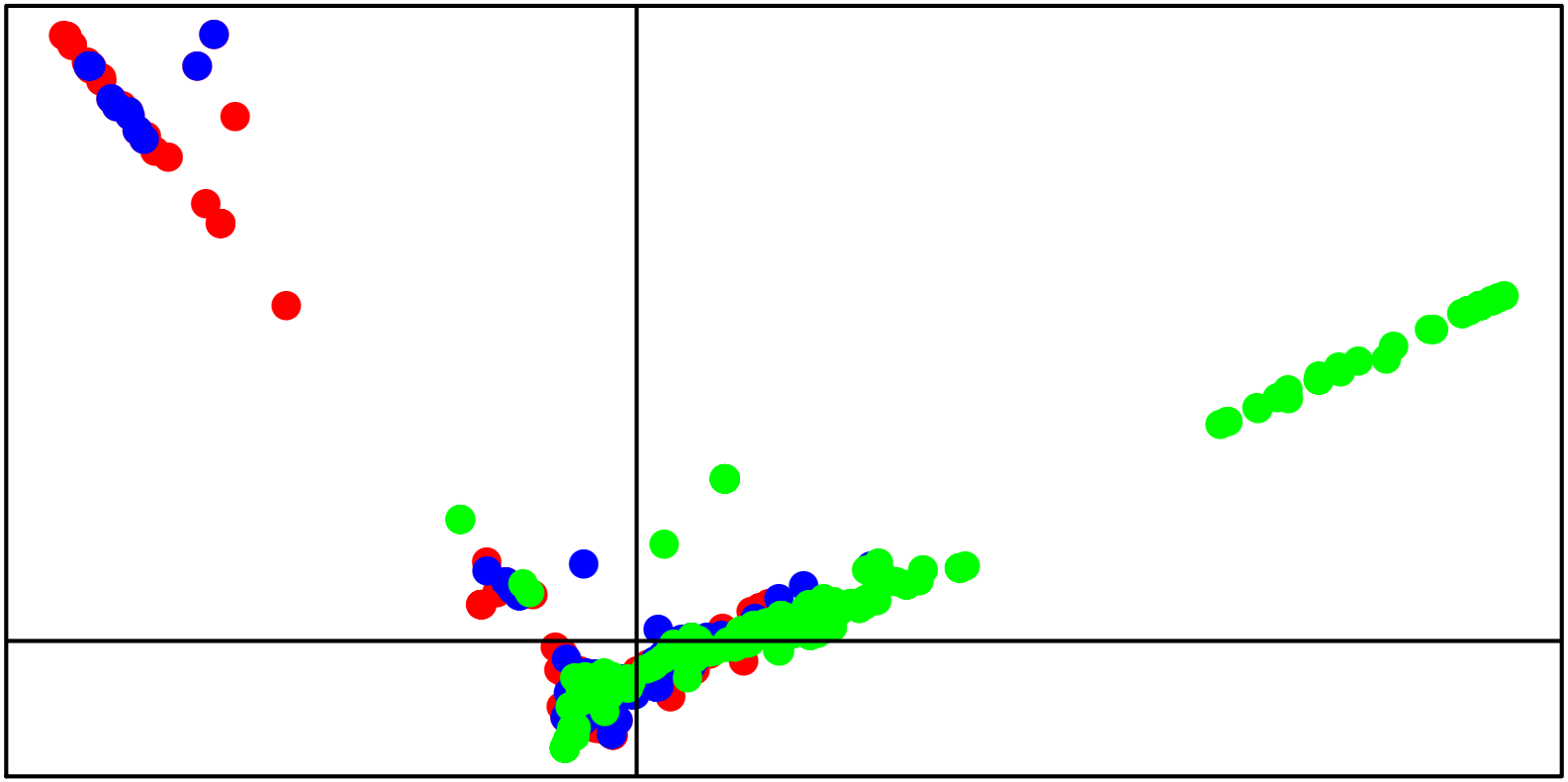


Figure: Similarity of tweets measured using Binary Distance

MDS: TF-IDF with Cosine Distance


```
M2 <- as.matrix(weightTfIdf(tdm))
M3 <- M2 %*% diag(1/sqrt(colSums(M2^2))) ## divide by vector norm
D <- dist(t(M3), method="euclidean")^2/2 ## square distance and divide by 2 to get
mds.tweet <- cmdscale(D, k=2)
```

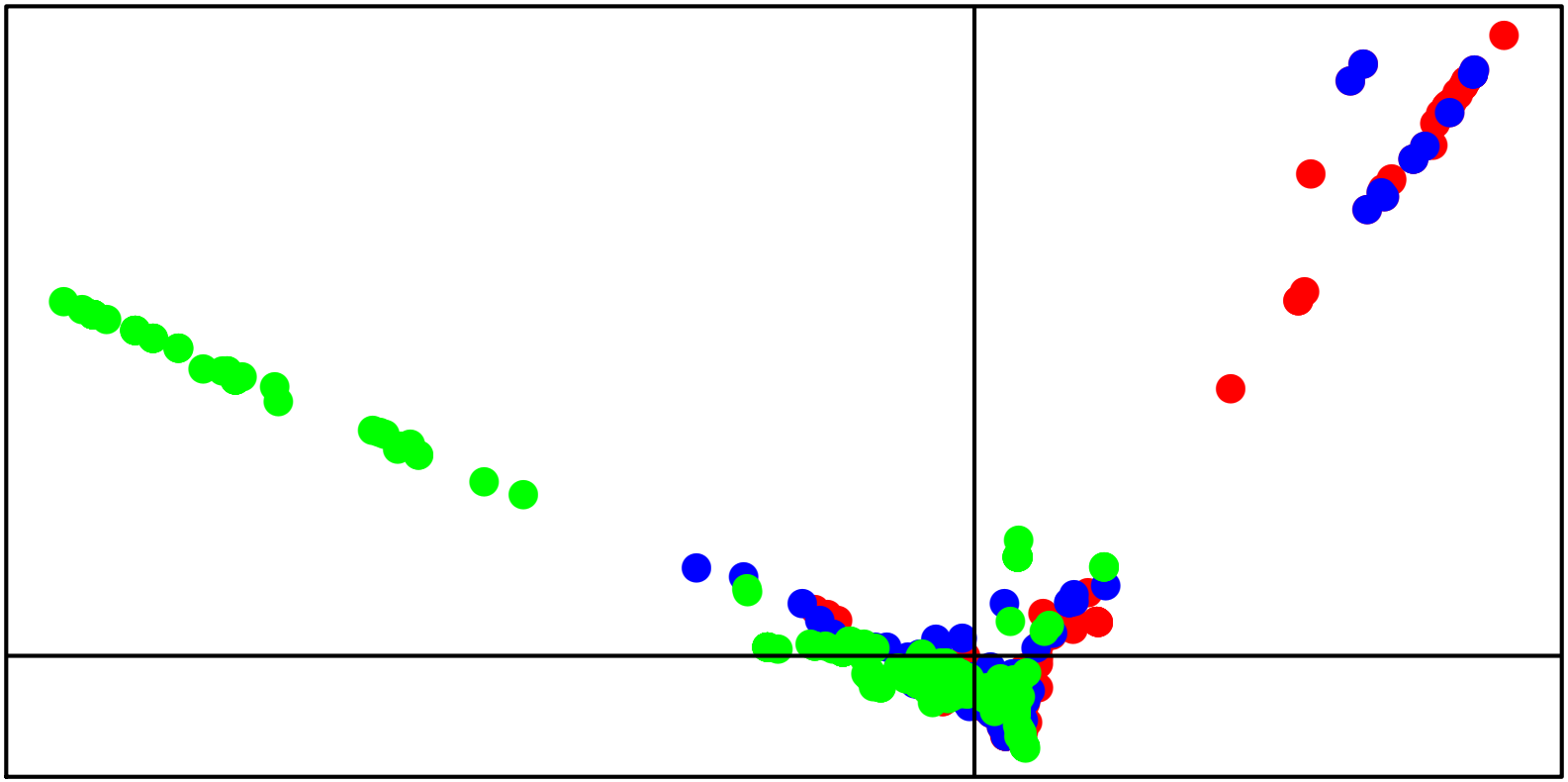


Figure: Similarity of tweets measured using Cosine Distance

Analysis of results

In the previous plots:

- Euclidean distance and Maximum distance did not provide good visualisation (all of the tweets are clumped together in a line.
- Cosine and Binary give very similar visualisation of the tweets. This is probably due to words only appearing either 0 or 1 time in a tweet (tweets are short).
- Manhattan distance gave similar but more spread version of the plots using Cosine and Binary distance.
- Cosine, Binary and Manhattan distances shows that all of the Bill Shorten and Malcolm Turnbull tweets were similar. There were a set from the Richard Di Natale that were similar to the others, but some that were different.

Outline

Crude Oil Texts

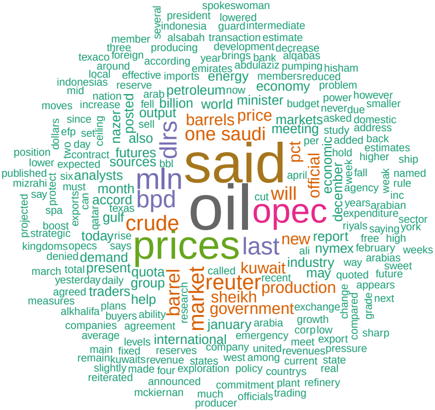


Figure: Word cloud from Crude Oil texts

Word Clouds

Word clouds are a simple visualisation of word frequencies in a corpus of texts.

Wordle is the most famous example.

The basic idea of a word cloud is that the size of the text should represent the frequency of the word, and words should be *placed* in some way.

A simple way to set the text size is using a linear scale.

$$s = s_{min} + (s_{max} - s_{min}) \frac{f - f_{min}}{f_{max} - f_{min}}$$

where here s is text size and f is frequency

Placing the words

The simple idea behind placing the words is this.

1. Place the first word (highest frequency) in the centre.
2. Assume some words are already placed.
3. Try placing the next word at the *current* position.
   - If it doesn't fit (ie. it overlaps with already placed words) move the *current* position out along a *spiral* and return to 3.
   - If it does fit place the word.
4. Repeat for any remaining words.

Placing the words

A spiral is used to place the words, it is easiest to express the equation for in polar coordinates.

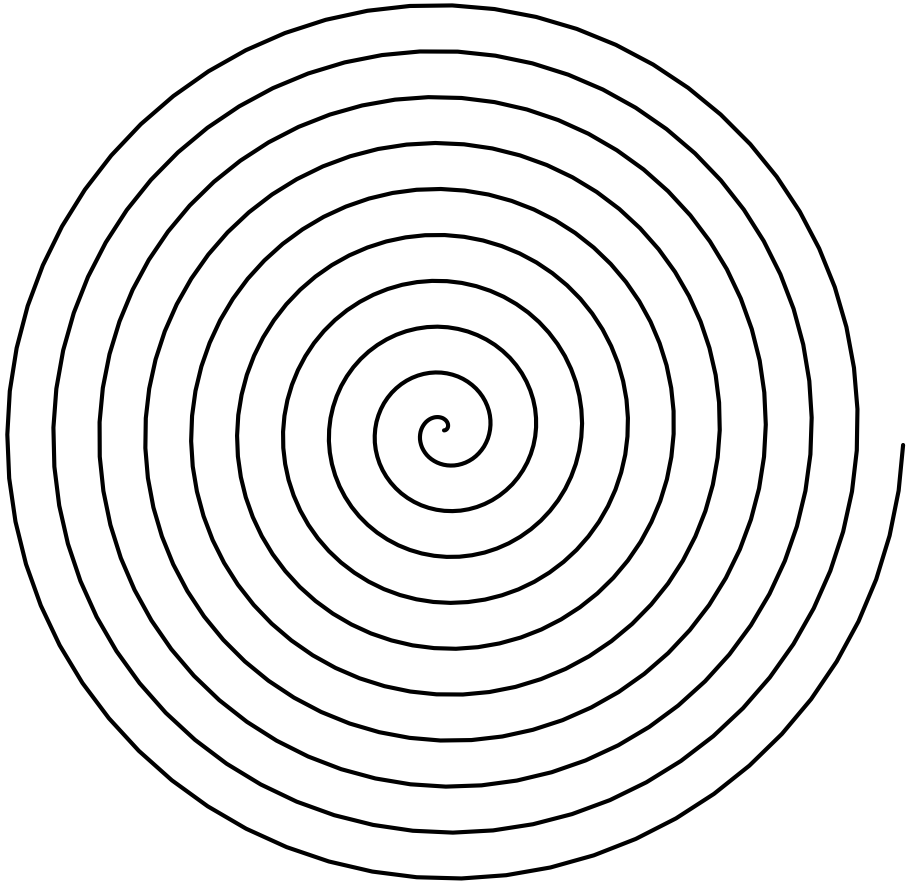
$$r = g(\theta)$$

where r is the radius (distance from the origin) and θ is the angle (can be greater than 360°)

The simplest form is a linear function of θ

$$r = \delta \times \theta$$

Placing the words



A Linear Spiral

Word Clouds

Animated Demo

```
library(animation)
source("mywordle.R")
mywordle(words,freq, pause=TRUE)
```

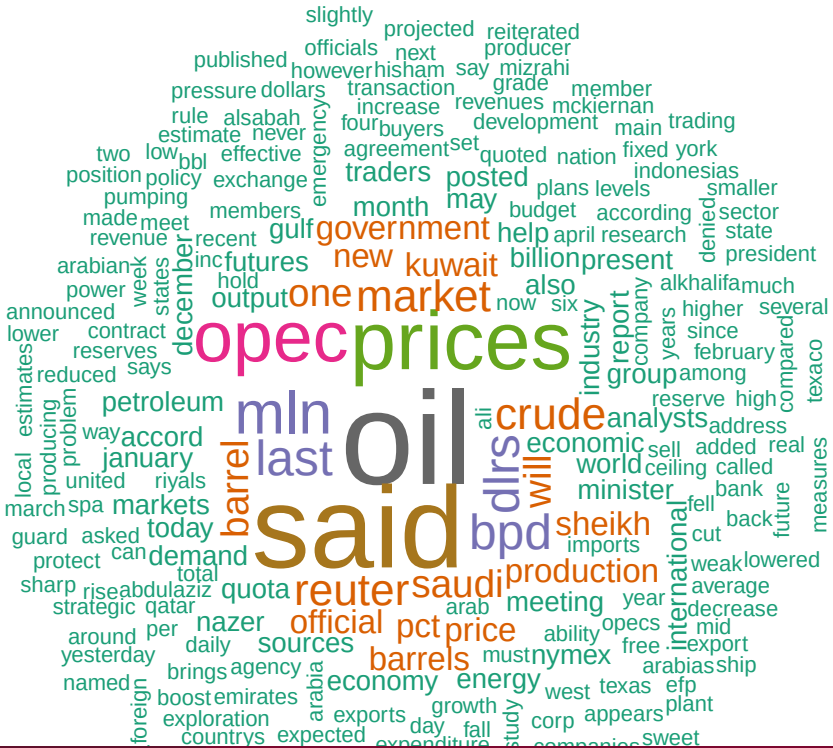
Word Clouds Code


```
boxes <- list()
r <- 0
theta <- 0
for(i in 1:length(words)) {
    repeat {
        b <- box(r, theta, words[i], sizes[i])
        if(!is.overlapped(b, boxes)) { break }
        theta <- theta + thetaStep
        r <- r + rStep
    }
    text(r*cos(theta),r*sin(theta), words[i], adj=c(0.5,0.5), cex=sizes[i])
    boxes <- c(list(b), boxes)
}
```

Word Clouds in R

Of course, R has prettier functions that allow colours and the rotation of some words (at random)

```
library(wordcloud)
wordcloud(words,freq, random.order=FALSE,
          colors=brewer.pal(8, "Dark2"))
```



Summary

- We can visualise word frequency using a word cloud.
- Principal Components Analysis can be used to visualise data where distances are measured using Euclidean distance.
- Multidimensional Scaling can be used to visualise data using any metric.
- Commonly used metrics are Euclidean distance, Manhattan distance, Maximum distance, Binary distance.
- Cosine distance is not a metric, but is useful for measuring document dissimilarity and can be used for MDS.

Next Week

Text Mining 2: Clustering