

# Cluster analysis

# Cluster Analysis

- One side-effect of discriminant analysis: could draw picture of data (if 1st 2s LDs told most of story) and see which individuals “close” to each other.
- Discriminant analysis requires knowledge of groups.
- Without knowledge of groups, use *cluster analysis*: see which individuals close together, which groups suggested by data.
- Idea: see how individuals group into “clusters” of nearby individuals.
- Base on “dissimilarities” between individuals.
- Or base on standard deviations and correlations between variables (assesses dissimilarity behind scenes).

# Packages

```
library(MASS) # for lda later  
library(tidyverse)  
library(spatstat) # for crossdist later  
library(ggrepel)  
library(conflicted)  
conflict_prefer("select", "dplyr")  
conflict_prefer("filter", "dplyr")
```

## One to ten in 11 languages

|    | English | Norwegian | Danish | Dutch | German |
|----|---------|-----------|--------|-------|--------|
| 1  | one     | en        | en     | een   | eins   |
| 2  | two     | to        | to     | twee  | zwei   |
| 3  | three   | tre       | tre    | drie  | drei   |
| 4  | four    | fire      | fire   | vier  | vier   |
| 5  | five    | fem       | fem    | vijf  | funf   |
| 6  | six     | seks      | seks   | zes   | sechs  |
| 7  | seven   | sju       | syv    | zeven | sieben |
| 8  | eight   | atte      | otte   | acht  | acht   |
| 9  | nine    | ni        | ni     | negen | neun   |
| 10 | ten     | ti        | ti     | tien  | zehn   |

# One to ten

|    | French | Spanish | Italian | Polish   | Hungarian | Finnish   |
|----|--------|---------|---------|----------|-----------|-----------|
| 1  | un     | uno     | uno     | jeden    | egy       | yksi      |
| 2  | deux   | dos     | due     | dwa      | ketto     | kaksi     |
| 3  | trois  | tres    | tre     | trzy     | harom     | kolme     |
| 4  | quatre | cuatro  | quattro | cztery   | negy      | nelja     |
| 5  | cinq   | cinco   | cinque  | piec     | ot        | viisi     |
| 6  | six    | seis    | sei     | szesc    | hat       | kuusi     |
| 7  | sept   | siete   | sette   | siedem   | het       | seitseman |
| 8  | huit   | ocho    | otto    | osiem    | nyolc     | kahdeksan |
| 9  | neuf   | nueve   | nove    | dziewiec | kilenc    | yhdeksan  |
| 10 | dix    | diez    | dieci   | dziesiec | tiz       | kymmenen  |

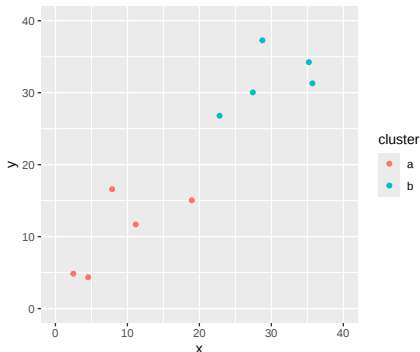
## Dissimilarities and languages example

- Can define dissimilarities how you like (whatever makes sense in application).
- Sometimes defining “similarity” makes more sense; can turn this into dissimilarity by subtracting from some maximum.
- Example: numbers 1–10 in various European languages. Define similarity between two languages by counting how often the same number has a name starting with the same letter (and dissimilarity by how often number has names starting with different letter).
- Crude (doesn't even look at most of the words), but see how effective.

## Two kinds of cluster analysis

- Looking at process of forming clusters (of similar languages): **hierarchical cluster analysis** (`hclust`).
- Start with each individual in cluster by itself.
- Join “closest” clusters one by one until all individuals in one cluster.
- How to define closeness of two *clusters*? Not obvious, investigate in a moment.
- Know how many clusters: which division into that many clusters is “best” for individuals? **K-means clustering** (`kmeans`).

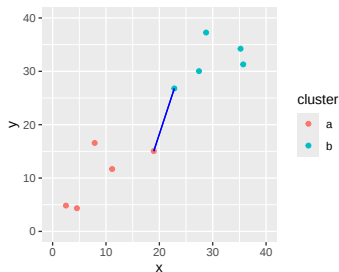
## Two made-up clusters



How to measure distance between set of red points and set of blue ones?

# Single-linkage distance

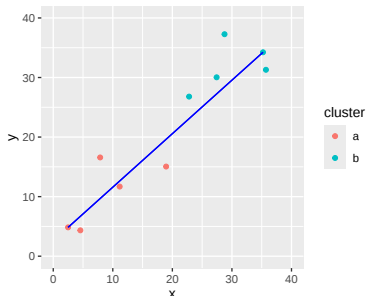
Find the red point and the blue point that are closest together:



Single-linkage distance between 2 clusters is distance between their closest points.

# Complete linkage

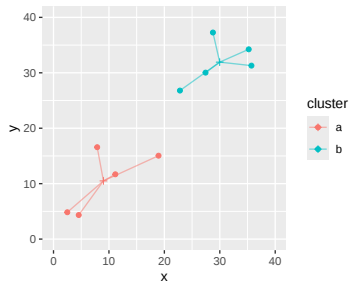
Find the red and blue points that are farthest apart:



Complete-linkage distance is distance between farthest points.

# Ward's method

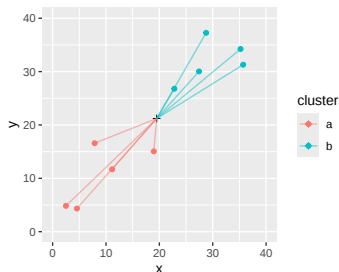
Work out mean of each cluster and join point to its mean:



Work out (i) sum of squared distances of points from means.

## Ward's method part 2

Now imagine combining the two clusters and working out overall mean.  
Join each point to this mean:



Calc sum of squared distances (ii) of points to combined mean.

## Ward's method part 3

- Sum of squares (ii) will be bigger than (i) (points closer to own cluster mean than combined mean).
- Ward's distance is (ii) minus (i).
- Think of as “cost” of combining clusters:
- if clusters close together, (ii) only a little larger than (i)
- if clusters far apart, (ii) a lot larger than (i) (as in example).

# Hierarchical clustering revisited

- Single linkage, complete linkage, Ward are ways of measuring closeness of clusters.
- Use them, starting with each observation in own cluster, to repeatedly combine two closest clusters until all points in one cluster.
- They will give different answers (clustering stories).
- Single linkage tends to make “stringy” clusters because clusters can be very different apart from two closest points.
- Complete linkage insists on whole clusters being similar.
- Ward tends to form many small clusters first.

## Dissimilarity data in R

Dissimilarities for language data were how many number names had *different* first letter:

```
my_url <- "http://ritsokiguess.site/datafiles/languages.txt"
(number.d <- read_table(my_url))
```

```
# A tibble: 11 x 12
```

|    | la    | en    | no    | dk    | nl    | de    | fr    | es    | it    |
|----|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|    | <chr> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> |
| 1  | en    | 0     | 2     | 2     | 7     | 6     | 6     | 6     | 6     |
| 2  | no    | 2     | 0     | 1     | 5     | 4     | 6     | 6     | 6     |
| 3  | dk    | 2     | 1     | 0     | 6     | 5     | 6     | 5     | 5     |
| 4  | nl    | 7     | 5     | 6     | 0     | 5     | 9     | 9     | 9     |
| 5  | de    | 6     | 4     | 5     | 5     | 0     | 7     | 7     | 7     |
| 6  | fr    | 6     | 6     | 6     | 9     | 7     | 0     | 2     | 1     |
| 7  | es    | 6     | 6     | 5     | 9     | 7     | 2     | 0     | 1     |
| 8  | it    | 6     | 6     | 5     | 9     | 7     | 1     | 1     | 0     |
| 9  | pl    | 7     | 7     | 6     | 10    | 8     | 5     | 3     | 4     |
| 10 | hu    | 9     | 8     | 8     | 8     | 9     | 10    | 10    | 10    |
| 11 | fi    | 9     | 9     | 9     | 9     | 9     | 9     | 9     | 8     |

```
# i 3 more variables: pl <dbl>, hu <dbl>, fi <dbl>
```

# Making a distance object

```
number.d %>%  
  select(-la) %>%  
  as.dist() -> d  
d
```

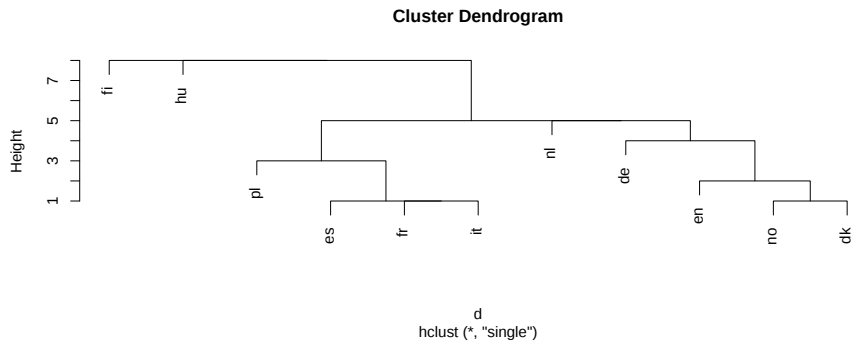
|    | en | no | dk | nl | de | fr | es | it | pl | hu |
|----|----|----|----|----|----|----|----|----|----|----|
| no |    | 2  |    |    |    |    |    |    |    |    |
| dk | 2  |    | 1  |    |    |    |    |    |    |    |
| nl | 7  | 5  | 6  |    |    |    |    |    |    |    |
| de | 6  | 4  | 5  | 5  |    |    |    |    |    |    |
| fr | 6  | 6  | 6  | 9  | 7  |    |    |    |    |    |
| es | 6  | 6  | 5  | 9  | 7  | 2  |    |    |    |    |
| it | 6  | 6  | 5  | 9  | 7  | 1  | 1  |    |    |    |
| pl | 7  | 7  | 6  | 10 | 8  | 5  | 3  | 4  |    |    |
| hu | 9  | 8  | 8  | 8  | 9  | 10 | 10 | 10 | 10 |    |
| fi | 9  | 9  | 9  | 9  | 9  | 9  | 9  | 8  | 9  | 8  |

```
class(d)
```

```
[1] "dist"
```

# Cluster analysis and dendrogram

```
d.hc <- hclust(d, method = "single")  
plot(d.hc)
```



# Comments

- Tree shows how languages combined into clusters.
- First (bottom), Spanish, French, Italian joined into one cluster, Norwegian and Danish into another.
- Later, English joined to Norse languages, Polish to Romance group.
- Then German, Dutch make a Germanic group.
- Finally, Hungarian and Finnish joined to each other and everything else.

# Clustering process 1/2

```
enframe(d.hc$labels)
```

```
# A tibble: 11 x 2
```

|    | name  | value |
|----|-------|-------|
|    | <int> | <chr> |
| 1  | 1     | en    |
| 2  | 2     | no    |
| 3  | 3     | dk    |
| 4  | 4     | nl    |
| 5  | 5     | de    |
| 6  | 6     | fr    |
| 7  | 7     | es    |
| 8  | 8     | it    |
| 9  | 9     | pl    |
| 10 | 10    | hu    |
| 11 | 11    | fi    |

## Clustering process 2/2

```
d.hc$merge
```

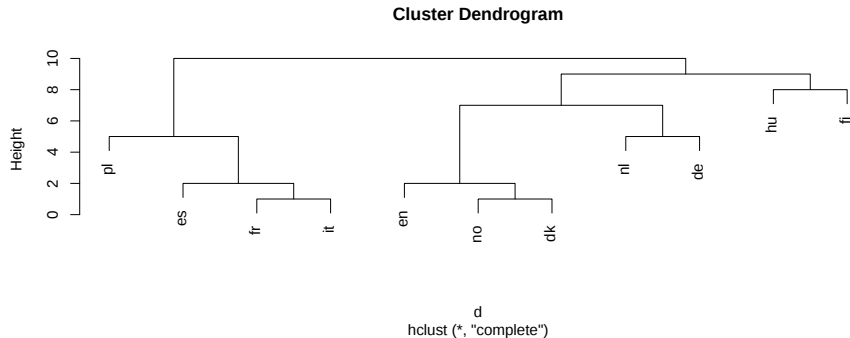
|       | [,1] | [,2] |
|-------|------|------|
| [1,]  | -2   | -3   |
| [2,]  | -6   | -8   |
| [3,]  | -7   | 2    |
| [4,]  | -1   | 1    |
| [5,]  | -9   | 3    |
| [6,]  | -5   | 4    |
| [7,]  | -4   | 6    |
| [8,]  | 5    | 7    |
| [9,]  | -10  | 8    |
| [10,] | -11  | 9    |

# Comments

- Lines of merge show what was combined
  - ▶ First, languages 2 and 3 (no and dk)
  - ▶ Then languages 6 and 8 (fr and it)
  - ▶ Then #7 combined with cluster formed at step 2 (es joined to fr and it).
  - ▶ Then en joined to no and dk ...
  - ▶ Finally fi joined to all others.

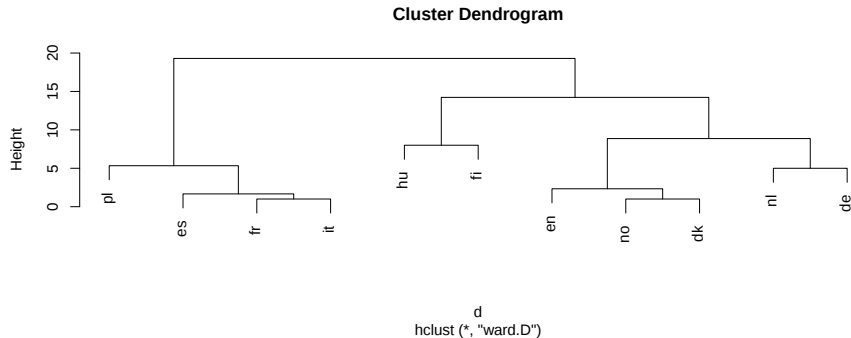
# Complete linkage

```
d.hc <- hclust(d, method = "complete")  
plot(d.hc)
```



# Ward

```
d.hc <- hclust(d, method = "ward.D")  
plot(d.hc)
```



# Chopping the tree

- Three clusters (from Ward) looks good:

```
cutree(d.hc, 3)
```

|    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|
| en | no | dk | nl | de | fr | es | it | pl | hu | fi |
| 1  | 1  | 1  | 1  | 1  | 2  | 2  | 2  | 2  | 3  | 3  |

## Turning the “named vector” into a data frame

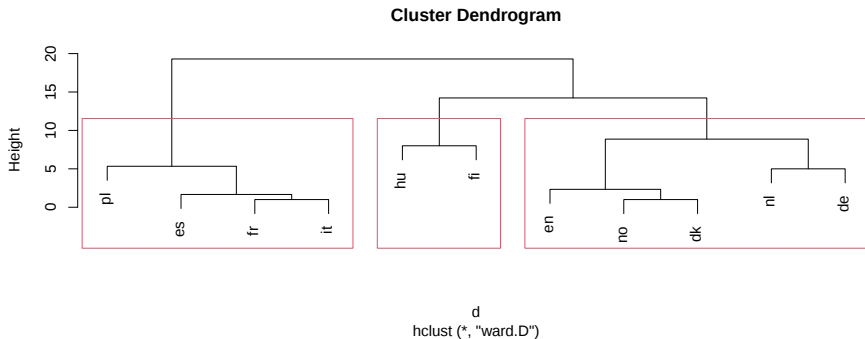
```
cutree(d.hc, 3) %>% enframe(name="country", value="cluster")
```

```
# A tibble: 11 x 2
```

|    | country | cluster |
|----|---------|---------|
|    | <chr>   | <int>   |
| 1  | en      | 1       |
| 2  | no      | 1       |
| 3  | dk      | 1       |
| 4  | nl      | 1       |
| 5  | de      | 1       |
| 6  | fr      | 2       |
| 7  | es      | 2       |
| 8  | it      | 2       |
| 9  | pl      | 2       |
| 10 | hu      | 3       |
| 11 | fi      | 3       |

# Drawing those clusters on the tree

```
plot(d.hc)  
rect.hclust(d.hc, 3)
```



# Comparing single-linkage and Ward

- In Ward, Dutch and German get joined earlier (before joining to Germanic cluster).
- Also Hungarian and Finnish get combined earlier.

# Making those dissimilarities

Original data:

```
my_url <- "http://ritsokiguess.site/datafiles/one-ten.txt"
lang <- read_delim(my_url, " ")
lang
```

```
# A tibble: 10 x 11
  en    no    dk    nl    de    fr    es    it    pl
  <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
1 one   en     en     een  eins  un    uno   uno   jeden
2 two   to     to     twee zwei  deux  dos   due   dwa
3 three tre   tre     drie drei  trois tres  tre   trzy
4 four  fire  fire    vier vier  quatre cuatro quatt~ czte~
5 five  fem   fem     vijf funf  cinq  cinco cinque piec
6 six   seks  seks    zes  sechs six   seis  sei   szesc
7 seven sju    syv     zeven sieben sept  siete sette sied~
8 eight atte otte    acht acht  huit  ocho  otto  osiem
9 nine  ni     ni     negen neun  neuf  nueve nove  dzie~
10 ten  ti     ti     tien zehn  dix   diez  dieci dzie~
# i 2 more variables: hu <chr>, fi <chr>
```

It would be a lot easier to extract the first letter if the number names were all in one column.

## Tidy, and extract first letter

```
lang %>% mutate(number=row_number()) %>%  
  pivot_longer(-number, names_to="language", values_to="name") %>%  
  mutate(first=str_sub(name, 1, 1)) -> lang.long  
lang.long
```

```
# A tibble: 110 x 4
```

|    | number | language | name  | first |
|----|--------|----------|-------|-------|
|    | <int>  | <chr>    | <chr> | <chr> |
| 1  | 1      | en       | one   | o     |
| 2  | 1      | no       | en    | e     |
| 3  | 1      | dk       | en    | e     |
| 4  | 1      | nl       | een   | e     |
| 5  | 1      | de       | eins  | e     |
| 6  | 1      | fr       | un    | u     |
| 7  | 1      | es       | uno   | u     |
| 8  | 1      | it       | uno   | u     |
| 9  | 1      | pl       | jeden | j     |
| 10 | 1      | hu       | egy   | e     |

```
# i 100 more rows
```

# Calculating dissimilarity

- Suppose we wanted dissimilarity between English and Norwegian. It's the number of first letters that are different.
- First get the lines for English:

```
english <- lang.long %>% filter(language == "en")  
english
```

```
# A tibble: 10 x 4  
  number language name first  
  <int> <chr>    <chr> <chr>  
1       1 en      one   o  
2       2 en      two   t  
3       3 en     three t  
4       4 en     four  f  
5       5 en     five  f  
6       6 en     six   s  
7       7 en    seven s  
8       8 en    eight e  
9       9 en     nine  n  
10      10 en     ten   t
```

## And then the lines for Norwegian

```
norwegian <- lang.long %>% filter(language == "no")  
norwegian
```

```
# A tibble: 10 x 4  
  number language name first  
  <int> <chr>    <chr> <chr>  
1      1 no      en     e  
2      2 no      to     t  
3      3 no      tre    t  
4      4 no      fire   f  
5      5 no      fem    f  
6      6 no      seks   s  
7      7 no      sju    s  
8      8 no      atte   a  
9      9 no      ni     n  
10     10 no      ti     t
```

And now we want to put them side by side, matched by number. This is what `left_join` does. (A “join” is a lookup of values in one table using another.)

# The join

```
english %>% left_join(norwegian, join_by(number))
```

```
# A tibble: 10 x 7
```

|    | number | language.x | name.x | first.x | language.y | name.y | first.y |
|----|--------|------------|--------|---------|------------|--------|---------|
|    | <int>  | <chr>      | <chr>  | <chr>   | <chr>      | <chr>  | <chr>   |
| 1  | 1      | en         | one    | o       | no         | en     | e       |
| 2  | 2      | en         | two    | t       | no         | to     | t       |
| 3  | 3      | en         | three  | t       | no         | tre    | t       |
| 4  | 4      | en         | four   | f       | no         | fire   | f       |
| 5  | 5      | en         | five   | f       | no         | fem    | f       |
| 6  | 6      | en         | six    | s       | no         | seks   | s       |
| 7  | 7      | en         | seven  | s       | no         | sju    | s       |
| 8  | 8      | en         | eight  | e       | no         | atte   | a       |
| 9  | 9      | en         | nine   | n       | no         | ni     | n       |
| 10 | 10     | en         | ten    | t       | no         | ti     | t       |

`first.x` is 1st letter of English word, `first.y` 1st letter of Norwegian word.

# Counting the different ones

```
english %>% left_join(norwegian, join_by(number)) %>%  
  count(different=(first.x != first.y))
```

```
# A tibble: 2 x 2  
  different      n  
  <lgl>      <int>  
1 FALSE         8  
2 TRUE          2
```

or

```
english %>% left_join(norwegian, join_by(number)) %>%  
  count(different=(first.x != first.y)) %>%  
  filter(different) %>% pull(n) -> ans  
ans
```

```
[1] 2
```

Words for 1 and 8 start with different letter; rest are same.

# A language with itself

The answer should be zero:

```
english %>% left_join(english, join_by(number)) %>%  
  count(different=(first.x != first.y)) %>%  
  filter(different) %>% pull(n) -> ans  
ans
```

integer(0)

- but this is “an integer vector of length zero”.
- so we have to allow for this possibility when we write a function to do it.

## Function to do this for any two languages

```
countdiff <- function(lang.1, lang.2, d) {  
  d %>% filter(language == lang.1) -> lang1d  
  d %>% filter(language == lang.2) -> lang2d  
  lang1d %>%  
    left_join(lang2d, join_by(number)) %>%  
    count(different = (first.x != first.y)) %>%  
    filter(different) %>% pull(n) -> ans  
  # if ans has length zero, set answer to (integer) zero.  
  ifelse(length(ans)==0, 0L, ans)  
}
```

# Testing

```
countdiff("en", "no", lang.long)
```

```
[1] 2
```

```
countdiff("en", "en", lang.long)
```

```
[1] 0
```

English and Norwegian have two different; English and English have none different.

Check.

## For all pairs of languages?

- First need all the languages:

```
languages <- names(lang)
languages
```

```
[1] "en" "no" "dk" "nl" "de" "fr" "es" "it" "pl"
[10] "hu" "fi"
```

- and then all *pairs* of languages:

```
pairs <- crossing(lang = languages, lang2 = languages)
```

# The pairs

```
pairs
```

```
# A tibble: 121 x 2
  lang lang2
  <chr> <chr>
1 de    de
2 de    dk
3 de    en
4 de    es
5 de    fi
6 de    fr
7 de    hu
8 de    it
9 de    nl
10 de   no
# i 111 more rows
```

## Run countdiff for all those language pairs

```
pairs %>% rowwise() %>%  
  mutate(diff = countdiff(lang, lang2, lang.long)) -> thediff  
thediff
```

```
# A tibble: 121 x 3
```

```
# Rowwise:
```

|    | lang  | lang2 | diff  |
|----|-------|-------|-------|
|    | <chr> | <chr> | <int> |
| 1  | de    | de    | 0     |
| 2  | de    | dk    | 5     |
| 3  | de    | en    | 6     |
| 4  | de    | es    | 7     |
| 5  | de    | fi    | 9     |
| 6  | de    | fr    | 7     |
| 7  | de    | hu    | 9     |
| 8  | de    | it    | 7     |
| 9  | de    | nl    | 5     |
| 10 | de    | no    | 4     |

```
# i 111 more rows
```

# Make square table of these

```
thediff %>% pivot_wider(names_from=lang2, values_from=diff)
```

```
# A tibble: 11 x 12
  lang      de      dk      en      es      fi      fr      hu      it
  <chr> <int> <int> <int> <int> <int> <int> <int> <int>
1 de         0       5       6       7       9       7       9       7
2 dk         5       0       2       5       9       6       8       5
3 en         6       2       0       6       9       6       9       6
4 es         7       5       6       0       9       2      10       1
5 fi         9       9       9       9       0       9       8       9
6 fr         7       6       6       2       9       0      10       1
7 hu         9       8       9      10       8      10       0      10
8 it         7       5       6       1       9       1      10       0
9 nl         5       6       7       9       9       9       8       9
10 no         4       1       2       6       9       6       8       6
11 pl         8       6       7       3       9       5      10       4

# i 3 more variables: nl <int>, no <int>, pl <int>
```

and that was where we began.

## Another example

Birth, death and infant mortality rates for 97 countries (variables not dissimilarities):

|      |      |      |                |      |      |      |                |
|------|------|------|----------------|------|------|------|----------------|
| 24.7 | 5.7  | 30.8 | Albania        | 12.5 | 11.9 | 14.4 | Bulgaria       |
| 13.4 | 11.7 | 11.3 | Czechoslovakia | 12   | 12.4 | 7.6  | Former_E._Germ |
| 11.6 | 13.4 | 14.8 | Hungary        | 14.3 | 10.2 | 16   | Poland         |
| 13.6 | 10.7 | 26.9 | Romania        | 14   | 9    | 20.2 | Yugoslavia     |
| 17.7 | 10   | 23   | USSR           | 15.2 | 9.5  | 13.1 | Byelorussia_SS |
| 13.4 | 11.6 | 13   | Ukrainian_SSR  | 20.7 | 8.4  | 25.7 | Argentina      |
| 46.6 | 18   | 111  | Bolivia        | 28.6 | 7.9  | 63   | Brazil         |
| 23.4 | 5.8  | 17.1 | Chile          | 27.4 | 6.1  | 40   | Columbia       |
| 32.9 | 7.4  | 63   | Ecuador        | 28.3 | 7.3  | 56   | Guyana         |
| ...  |      |      |                |      |      |      |                |

- Want to find groups of similar countries (and how many groups, which countries in each group).
- Tree would be unwieldy with 97 countries.
- More automatic way of finding given number of clusters?

## Reading in

```
url <- "http://ritsokiguess.site/datafiles/birthrate.txt"
vital <- read_table(url)
vital
```

```
# A tibble: 97 x 4
```

|    | birth | death | infant | country        |
|----|-------|-------|--------|----------------|
|    | <dbl> | <dbl> | <dbl>  | <chr>          |
| 1  | 24.7  | 5.7   | 30.8   | Albania        |
| 2  | 13.4  | 11.7  | 11.3   | Czechoslovakia |
| 3  | 11.6  | 13.4  | 14.8   | Hungary        |
| 4  | 13.6  | 10.7  | 26.9   | Romania        |
| 5  | 17.7  | 10    | 23     | USSR           |
| 6  | 13.4  | 11.6  | 13     | Ukrainian_SSR  |
| 7  | 46.6  | 18    | 111    | Bolivia        |
| 8  | 23.4  | 5.8   | 17.1   | Chile          |
| 9  | 32.9  | 7.4   | 63     | Ecuador        |
| 10 | 34.8  | 6.6   | 42     | Paraguay       |

# Standardizing

- Infant mortality rate numbers bigger than others, consequence of measurement scale (arbitrary).
- Standardize (numerical) columns of data frame to have mean 0, SD 1, done by scale.

```
vital %>%  
  mutate(across(where(is.numeric), \(x) scale(x))) -> vital.s  
vital.s
```

```
# A tibble: 97 x 4
```

|   | birth[,1] | death[,1] | infant[,1] | country        |
|---|-----------|-----------|------------|----------------|
|   | <dbl>     | <dbl>     | <dbl>      | <chr>          |
| 1 | -0.334    | -1.11     | -0.524     | Albania        |
| 2 | -1.17     | 0.186     | -0.948     | Czechoslovakia |
| 3 | -1.30     | 0.552     | -0.872     | Hungary        |
| 4 | -1.15     | -0.0293   | -0.609     | Romania        |
| 5 | -0.851    | -0.180    | -0.694     | USSR           |
| 6 | -1.17     | 0.164     | -0.911     | Ukrainian_SSR  |

## Three clusters

Pretend we know 3 clusters is good. Take off the column of countries, and run `kmeans` on the resulting data frame, asking for 3 clusters:

```
vital.s %>% select(-country) %>%  
  kmeans(3) -> vital.km3  
vital.km3
```

K-means clustering with 3 clusters of sizes 40, 25, 32

Cluster means:

|   | birth      | death      | infant      |
|---|------------|------------|-------------|
| 1 | -1.0376994 | -0.3289046 | -0.90669032 |
| 2 | 1.1780071  | 1.3323130  | 1.32732200  |
| 3 | 0.3768062  | -0.6297388 | 0.09639258  |

Clustering vector:

```
[1] 3 1 1 1 1 1 2 1 3 3 1 2 1 1 1 1 1 1 1 1 2 2 1 3 3 3 2  
[29] 1 3 1 3 3 1 1 3 3 3 2 2 3 3 2 2 3 2 2 2 3 1 1 1 1 1 1 3
```

# What's in the output?

- Cluster sizes:

```
vital.km3$size
```

```
[1] 40 25 32
```

- Cluster centres:

```
vital.km3$centers
```

|   | birth      | death      | infant      |
|---|------------|------------|-------------|
| 1 | -1.0376994 | -0.3289046 | -0.90669032 |
| 2 | 1.1780071  | 1.3323130  | 1.32732200  |
| 3 | 0.3768062  | -0.6297388 | 0.09639258  |

- Cluster 1 has lower than average rates on everything; cluster 2 has much higher than average.

## Cluster sums of squares and membership

```
vital.km3$withinss
```

```
[1] 17.21617 28.32560 21.53020
```

Cluster 1 compact relative to others (countries in cluster 1 more similar).

```
vital.km3$cluster
```

```
[1] 3 1 1 1 1 1 2 1 3 3 1 2 1 1 1 1 1 1 1 2 2 1 3 3 3 2  
[29] 1 3 1 3 3 1 1 3 3 3 2 2 3 3 2 2 3 2 2 3 1 1 1 1 1 1 3  
[57] 3 3 3 3 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 1 2 1 3 3 2 3 1 3  
[85] 2 2 2 2 3 2 2 2 2 2 3 2 2
```

The cluster membership for each of the 97 countries.

## Store countries and clusters to which they belong

```
vital.3 <- tibble(  
  country = vital.s$country,  
  cluster = vital.km3$cluster  
)  
vital.3
```

# A tibble: 97 x 2

|   | country        | cluster |
|---|----------------|---------|
|   | <chr>          | <int>   |
| 1 | Albania        | 3       |
| 2 | Czechoslovakia | 1       |
| 3 | Hungary        | 1       |
| 4 | Romania        | 1       |
| 5 | USSR           | 1       |
| 6 | Ukrainian_SSR  | 1       |
| 7 | Bolivia        | 2       |
| 8 | Chile          | 1       |

## Which countries in which cluster?

Write function to extract cluster number `i` from dataframe `d` with a column called `country`:

```
get_countries <- function(i, d) {  
  d %>% filter(cluster == i) %>% pull(country)  
}
```

## Cluster membership: cluster 2

```
get_countries(2, vital.3)
```

|      |             |              |                |
|------|-------------|--------------|----------------|
| [1]  | "Bolivia"   | "Mexico"     | "Afghanistan"  |
| [4]  | "Iran"      | "Bangladesh" | "Gabon"        |
| [7]  | "Ghana"     | "Namibia"    | "Sierra_Leone" |
| [10] | "Swaziland" | "Uganda"     | "Zaire"        |
| [13] | "Cambodia"  | "Nepal"      | "Angola"       |
| [16] | "Congo"     | "Ethiopia"   | "Gambia"       |
| [19] | "Malawi"    | "Mozambique" | "Nigeria"      |
| [22] | "Somalia"   | "Sudan"      | "Tanzania"     |
| [25] | "Zambia"    |              |                |

## Cluster 3

```
get_countries(3, vital.3)
```

|      |            |                |                |
|------|------------|----------------|----------------|
| [1]  | "Albania"  | "Ecuador"      | "Paraguay"     |
| [4]  | "Kuwait"   | "Oman"         | "Turkey"       |
| [7]  | "India"    | "Mongolia"     | "Pakistan"     |
| [10] | "Algeria"  | "Botswana"     | "Egypt"        |
| [13] | "Libya"    | "Morocco"      | "South_Africa" |
| [16] | "Zimbabwe" | "Brazil"       | "Columbia"     |
| [19] | "Guyana"   | "Peru"         | "Venezuela"    |
| [22] | "Bahrain"  | "Iraq"         | "Jordan"       |
| [25] | "Lebanon"  | "Saudi_Arabia" | "Indonesia"    |
| [28] | "Malaysia" | "Philippines"  | "Vietnam"      |
| [31] | "Kenya"    | "Tunisia"      |                |

# Cluster 1

```
get_countries(1, vital.3)
```

|                          |                   |
|--------------------------|-------------------|
| [1] "Czechoslovakia"     | "Hungary"         |
| [3] "Romania"            | "USSR"            |
| [5] "Ukrainian_SSR"      | "Chile"           |
| [7] "Uruguay"            | "Finland"         |
| [9] "France"             | "Greece"          |
| [11] "Italy"             | "Norway"          |
| [13] "Spain"             | "Switzerland"     |
| [15] "Austria"           | "Canada"          |
| [17] "Israel"            | "China"           |
| [19] "Korea"             | "Singapore"       |
| [21] "Thailand"          | "Bulgaria"        |
| [23] "Former_E._Germany" | "Poland"          |
| [25] "Yugoslavia"        | "Byelorussia_SSR" |
| [27] "Argentina"         | "Belgium"         |
| [29] "Denmark"           | "Germany"         |

# Problem!

- kmeans uses randomization. So result of one run might be different from another run.
- Example: just run again on 3 clusters, table of results:

```
vital.s %>%  
  select(-country) %>% kmeans(3) -> vital.km3a  
table(  
  first = vital.km3$cluster,  
  second = vital.km3a$cluster  
)
```

|       | second |    |    |
|-------|--------|----|----|
| first | 1      | 2  | 3  |
| 1     | 40     | 0  | 0  |
| 2     | 0      | 24 | 1  |
| 3     | 4      | 0  | 28 |

- Clusters are similar but *not same*.

## Solution to this

- `nstart` option on `kmeans` runs that many times, takes best. Should be same every time:

```
vital.s %>%  
  select(-country) %>%  
  kmeans(3, nstart = 20) -> vital.km3b
```

# How many clusters?

- Three was just a guess.
- Idea: try a whole bunch of `#clusters` (say 2–20), obtain measure of goodness of fit for each, make plot.
- Appropriate measure is `tot.withinss`.
- Run `kmeans` for each `#clusters`, get `tot.withinss` each time.

## Function to get tot.withinss

...for an input number of clusters, taking only numeric columns of input data frame:

```
ss <- function(i, d) {  
  d %>%  
    select(where(is.numeric)) %>%  
    kmeans(i, nstart = 20) -> km  
  km$tot.withinss  
}
```

Note: writing function to be as general as possible, so that we can re-use it later.

## Constructing within-cluster SS

Make a data frame with desired numbers of clusters, and fill it with the total within-group sums of squares. `ss` expects a single number of clusters, not a vector of several, so run rowwise:

```
tibble(clusters = 2:20) %>%  
  rowwise() %>%  
  mutate(wss = ss(clusters, vital.s)) -> ssd  
ssd
```

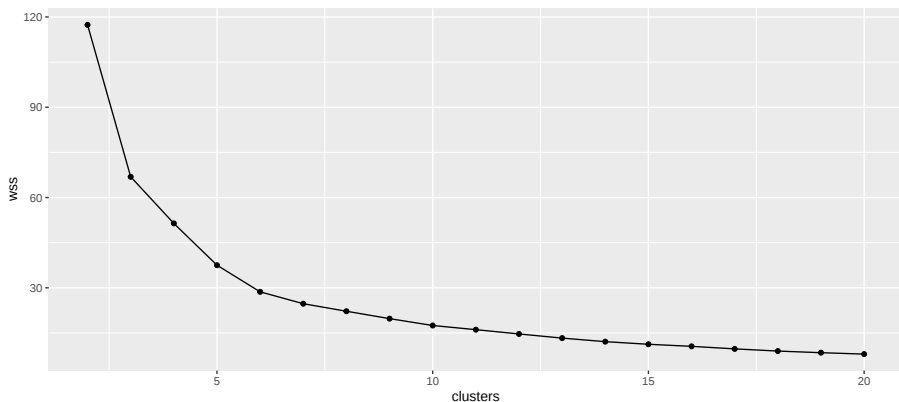
```
# A tibble: 19 x 2
```

```
# Rowwise:
```

|   | clusters | wss   |
|---|----------|-------|
|   | <int>    | <dbl> |
| 1 | 2        | 117.  |
| 2 | 3        | 66.9  |
| 3 | 4        | 51.4  |
| 4 | 5        | 37.5  |
| 5 | 6        | 28.7  |

# Scree plot

```
ggplot(ssd, aes(x = clusters, y = wss)) + geom_point() +  
  geom_line()
```



# Interpreting scree plot

- Lower wss better.
- But lower for larger #clusters, harder to explain.
- Compromise: low-ish wss and low-ish #clusters.
- Look for “elbow” in plot.
- Idea: this is where wss decreases fast then slow.
- On our plot, small elbow at 6 clusters. Try this many clusters.

## Six clusters, using nstart

```
set.seed(457299)
```

```
vital.s %>%  
  select(-country) %>%  
  kmeans(6, nstart = 20) -> vital.km6  
vital.km6$size
```

```
[1] 24 18 15  2  8 30
```

```
vital.km6$centers
```

|   | birth      | death      | infant     |
|---|------------|------------|------------|
| 1 | 0.4160993  | -0.5169988 | 0.2648754  |
| 2 | 1.2092406  | 0.7441347  | 1.0278003  |
| 3 | -0.4357690 | -1.1438599 | -0.7281108 |
| 4 | -0.2199722 | 2.1116577  | -0.4544435 |
| 5 | 1.3043848  | 2.1896567  | 1.9470306  |
| 6 | -1.1737104 | -0.1856375 | -0.9534370 |

## Make a data frame of countries and clusters

```
vital.6 <- tibble(  
  country = vital.s$country,  
  cluster = vital.km6$cluster  
)  
vital.6 %>% sample_n(10)
```

# A tibble: 10 x 2

|   | country<br><chr> | cluster<br><int> |
|---|------------------|------------------|
| 1 | Ghana            | 2                |
| 2 | Ukrainian_SSR    | 6                |
| 3 | Ethiopia         | 5                |
| 4 | Somalia          | 5                |
| 5 | Oman             | 1                |
| 6 | Botswana         | 2                |
| 7 | Paraguay         | 1                |
| 8 | Czechoslovakia   | 6                |

# Cluster 1

Below-average death rate, though other rates a little higher than average:

```
get_countries(1, vital.6)
```

|      |               |                |                |
|------|---------------|----------------|----------------|
| [1]  | "Ecuador"     | "Paraguay"     | "Oman"         |
| [4]  | "Turkey"      | "India"        | "Mongolia"     |
| [7]  | "Pakistan"    | "Algeria"      | "Egypt"        |
| [10] | "Libya"       | "Morocco"      | "South_Africa" |
| [13] | "Zimbabwe"    | "Brazil"       | "Guyana"       |
| [16] | "Peru"        | "Iraq"         | "Jordan"       |
| [19] | "Lebanon"     | "Saudi_Arabia" | "Indonesia"    |
| [22] | "Philippines" | "Vietnam"      | "Tunisia"      |

## Cluster 2

High on everything:

```
get_countries(2, vital.6)
```

|      |            |          |              |             |
|------|------------|----------|--------------|-------------|
| [1]  | "Bolivia"  | "Iran"   | "Bangladesh" | "Botswana"  |
| [5]  | "Gabon"    | "Ghana"  | "Namibia"    | "Swaziland" |
| [9]  | "Uganda"   | "Zaire"  | "Cambodia"   | "Nepal"     |
| [13] | "Congo"    | "Kenya"  | "Nigeria"    | "Sudan"     |
| [17] | "Tanzania" | "Zambia" |              |             |

## Cluster 3

Low on everything:

```
get_countries(3, vital.6)
```

|      |             |                        |
|------|-------------|------------------------|
| [1]  | "Albania"   | "Chile"                |
| [3]  | "Israel"    | "Kuwait"               |
| [5]  | "China"     | "Singapore"            |
| [7]  | "Thailand"  | "Argentina"            |
| [9]  | "Columbia"  | "Venezuela"            |
| [11] | "Bahrain"   | "United_Arab_Emirates" |
| [13] | "Hong_Kong" | "Malaysia"             |
| [15] | "Sri_Lanka" |                        |

## Cluster 4

Very high death rate, just below average on all else:

```
get_countries(4, vital.6)
```

```
[1] "Mexico" "Korea"
```

## Cluster 5

Very high on everything:

```
get_countries(5, vital.6)
```

```
[1] "Afghanistan"  "Sierra_Leone" "Angola"  
[4] "Ethiopia"     "Gambia"       "Malawi"  
[7] "Mozambique"   "Somalia"
```

## Cluster 6

A bit below average on everything:

```
get_countries(6, vital.6)
```

|      |                     |                   |
|------|---------------------|-------------------|
| [1]  | "Czechoslovakia"    | "Hungary"         |
| [3]  | "Romania"           | "USSR"            |
| [5]  | "Ukrainian_SSR"     | "Uruguay"         |
| [7]  | "Finland"           | "France"          |
| [9]  | "Greece"            | "Italy"           |
| [11] | "Norway"            | "Spain"           |
| [13] | "Switzerland"       | "Austria"         |
| [15] | "Canada"            | "Bulgaria"        |
| [17] | "Former_E._Germany" | "Poland"          |
| [19] | "Yugoslavia"        | "Byelorussia_SSR" |
| [21] | "Belgium"           | "Denmark"         |
| [23] | "Germany"           | "Ireland"         |
| [25] | "Netherlands"       | "Portugal"        |
| [27] | "Sweden"            | "U.K."            |

## Comparing our 3 and 6-cluster solutions

```
table(three = vital.km3$cluster, six = vital.km6$cluster)
```

|       | six |    |   |   |   |    |
|-------|-----|----|---|---|---|----|
| three | 1   | 2  | 3 | 4 | 5 | 6  |
| 1     | 0   | 0  | 9 | 1 | 0 | 30 |
| 2     | 0   | 16 | 0 | 1 | 8 | 0  |
| 3     | 24  | 2  | 6 | 0 | 0 | 0  |

Compared to 3-cluster solution:

- most of (old) cluster 1 gone to (new) cluster 6
- Old cluster 2 split into new clusters 2 and 5 (two types of “developing” countries)
- Old cluster 3 split into new clusters 1 and 3 (two types of “intermediate” countries, divided by death rate).

## Getting a picture from `kmeans`

- Use discriminant analysis on clusters found, treating them as “known” groups.

# Discriminant analysis

- So what makes the groups different?
- Uses package MASS (loaded):

```
vital.lda <- lda(vital.km6$cluster ~ birth + death + infant,  
                data = vital.s)  
vital.lda$svd
```

```
[1] 21.687195  8.851811  1.773006
```

```
vital.lda$scaling
```

|        | LD1       | LD2        | LD3        |
|--------|-----------|------------|------------|
| birth  | 2.6879695 | 1.1224202  | -1.9483853 |
| death  | 0.6652712 | -2.7213044 | -0.6049358 |
| infant | 2.1111801 | 0.7650912  | 2.3542296  |

- LD1 is some of everything (high=poor, low=rich).
- LD2 mainly death rate, high or low.

# A data frame to make plot from

- Get predictions first:

```
vital.pred <- predict(vital.lda)
d <- data.frame(
  country = vital.s$country,
  cluster = vital.km6$cluster,
  vital.pred$x
)
```

|   | country        | cluster | LD1         |
|---|----------------|---------|-------------|
| 1 | Albania        | 3       | -2.74034473 |
| 2 | Czechoslovakia | 6       | -5.01874312 |
| 3 | Hungary        | 6       | -4.97189595 |
| 4 | Romania        | 6       | -4.40612396 |
| 5 | USSR           | 6       | -3.87181416 |
| 6 | Ukrainian_SSR  | 6       | -4.95502329 |
| 7 | Bolivia        | 2       | 7.04719692  |
| 8 | Chile          | 3       | -3.61284528 |
| 9 | Ecuador        | 1       | 0.60813286  |

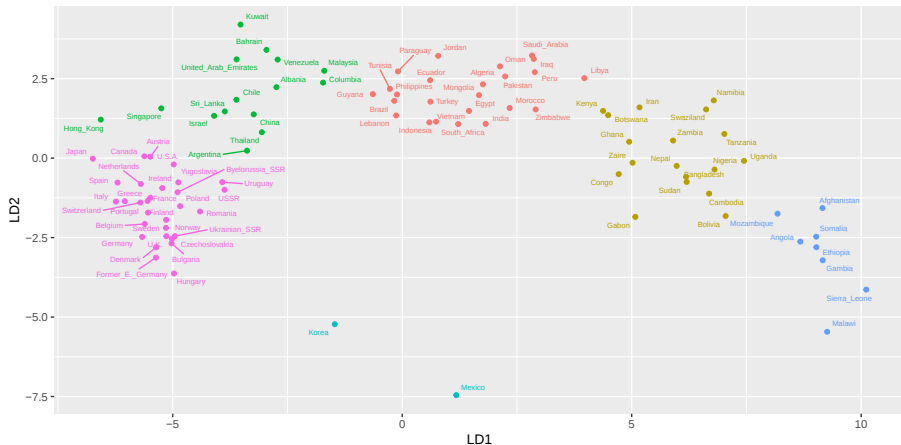
# What's in there; making a plot

- d contains country names, cluster memberships and discriminant scores.
- Plot LD1 against LD2, colouring points by cluster and labelling by country:

```
g <- ggplot(d, aes(  
  x = LD1, y = LD2, colour = factor(cluster),  
  label = country  
)) + geom_point() +  
  geom_text_repel(size = 2, max.overlaps = Inf) +  
  guides(colour = "none")
```

# The plot

g

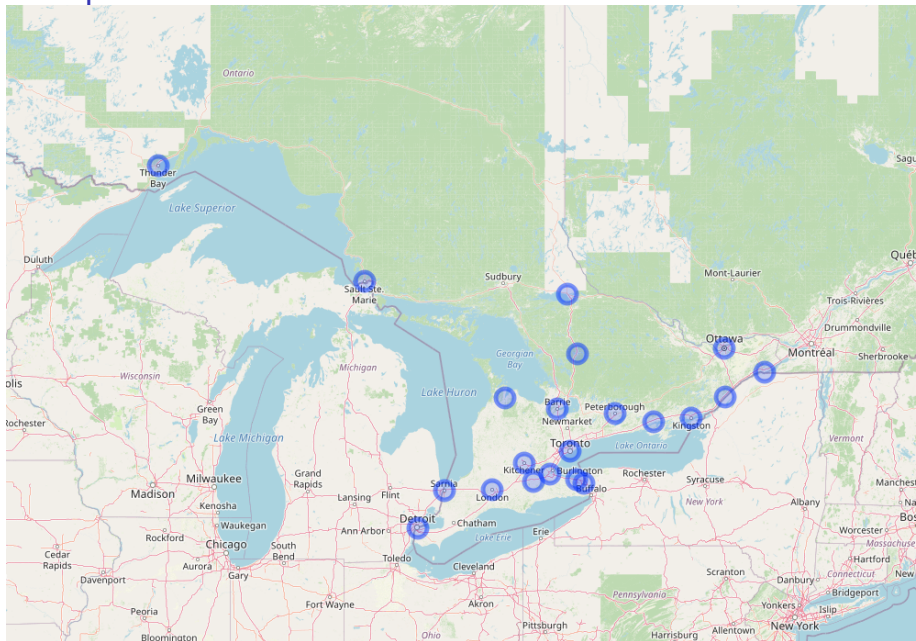


It would be better to zoom in on parts of this plot.

## Final example: a hockey league

- An Ontario hockey league has teams in 21 cities. How can we arrange those teams into 4 geographical divisions?
- Distance data in spreadsheet.
- Take out spaces in team names.
- Save as “text/csv”.
- Distances, so back to `hclust`.

# A map

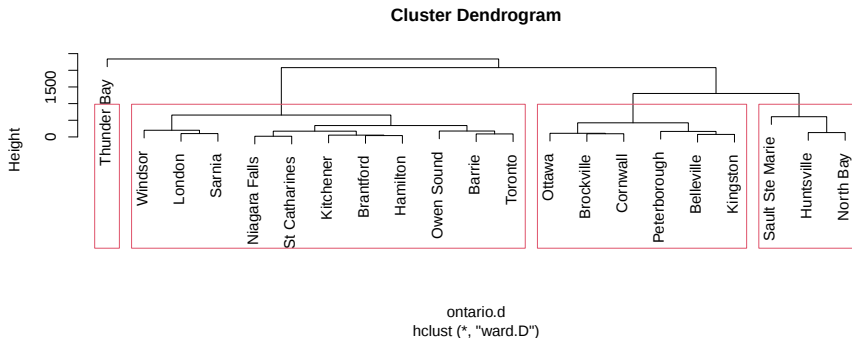


# Attempt 1

```
my_url <-  
  "http://ritsokiguess.site/datafiles/ontario-road-distances.csv"  
ontario <- read_csv(my_url)  
ontario.d <- ontario %>% select(-1) %>% as.dist()  
ontario.hc <- hclust(ontario.d, method = "ward.D")
```

# Plot, with 4 clusters

```
plot(ontario.hc)  
rect.hclust(ontario.hc, 4)
```

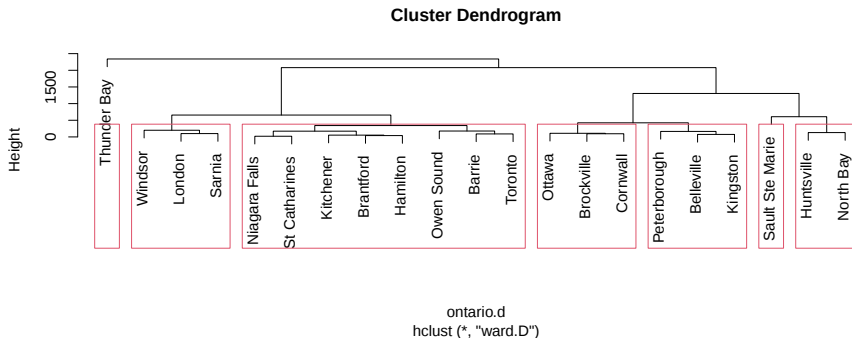


# Comments

- Can't have divisions of 1 team!
- “Southern” divisions way too big!
- Try splitting into more. I found 7 to be good:

# Seven clusters

```
plot(ontario.hc)  
rect.hclust(ontario.hc, 7)
```



## Divisions now

- I want to put Huntsville and North Bay together with northern teams.
- I'll put the Eastern teams together. Gives:
- North: Sault Ste Marie, Sudbury, Huntsville, North Bay
- East: Brockville, Cornwall, Ottawa, Peterborough, Belleville, Kingston
- West: Windsor, London, Sarnia
- Central: Owen Sound, Barrie, Toronto, Niagara Falls, St Catharines, Brantford, Hamilton, Kitchener
- Getting them same size beyond us!

## Another map

