Clustering

Hierarchical clustering

June 14th, 2023

Gapminder data

Health and income outcomes for 184 countries from 1960 to 2016 from the famous Gapminder project

```
library(tidyverse)
library(dslabs)
gapminder <- as_tibble(gapminder)
head(gapminder)
```

```
## # A tibble: 6 × 9
##   country            year infan…¹ life_…² ferti…³ popul…⁴      gdp conti…⁵ region
##   <fct>             <int>   <dbl>   <dbl>   <dbl>   <dbl>    <dbl> <fct>   <fct>
## 1 Albania            1960   115.     62.9    6.19  1.64e6 NA       Europe  South…
## 2 Algeria            1960   148.     47.5    7.65  1.11e7  1.38e10 Africa  North…
## 3 Angola             1960   208      36.0    7.32  5.27e6 NA       Africa  Middl…
## 4 Antigua and Bar…   1960    NA      63.0    4.43  5.47e4 NA       Americ… Carib…
## 5 Argentina          1960    59.9    65.4    3.11  2.06e7  1.08e11 Americ… South…
## 6 Armenia            1960    NA      66.9    4.55  1.87e6 NA       Asia    Weste…
## # … with abbreviated variable names ¹infant_mortality, ²life_expectancy,
## #   ³fertility, ⁴population, ⁵continent
```

Cleaning and transformation...

- Each row is at the `country-year` level
- Will just focus on data for 2011 where `gdp` is not missing
- Take `log()` transformation of `gdp`

```
clean_gapminder <- gapminder %>%
  filter(year == 2011, !is.na(gdp)) %>%
  mutate(log_gdp = log(gdp))
clean_gapminder
```

```
## # A tibble: 168 × 10
##    country    year infan…¹ life_…² ferti…³ popul…⁴     gdp conti…⁵ region log_gdp
##    <fct>     <int>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <fct>   <fct>    <dbl>
##  1 Albania    2011    14.3    77.4    1.75  2.89e6 6.32e 9 Europe  South…    22.6
##  2 Algeria    2011    22.8    76.1    2.83  3.67e7 8.11e10 Africa  North…    25.1
##  3 Angola     2011   107.     58.1    6.1   2.19e7 2.70e10 Africa  Middl…    24.0
##  4 Antigua…   2011     7.2    75.9    2.12  8.82e4 8.02e 8 Americ… Carib…    20.5
##  5 Argenti…   2011    12.7    76      2.2   4.17e7 4.73e11 Americ… South…    26.9
##  6 Armenia    2011    15.3    73.5    1.5   2.97e6 4.29e 9 Asia    Weste…    22.2
##  7 Austral…   2011     3.8    82.2    1.88  2.25e7 5.73e11 Oceania Austr…    27.1
##  8 Austria    2011     3.4    80.7    1.44  8.42e6 2.31e11 Europe  Weste…    26.2
##  9 Azerbai…   2011    32.5    70.8    1.96  9.23e6 2.14e10 Asia    Weste…    23.8
```

Let's work from the bottom-up...

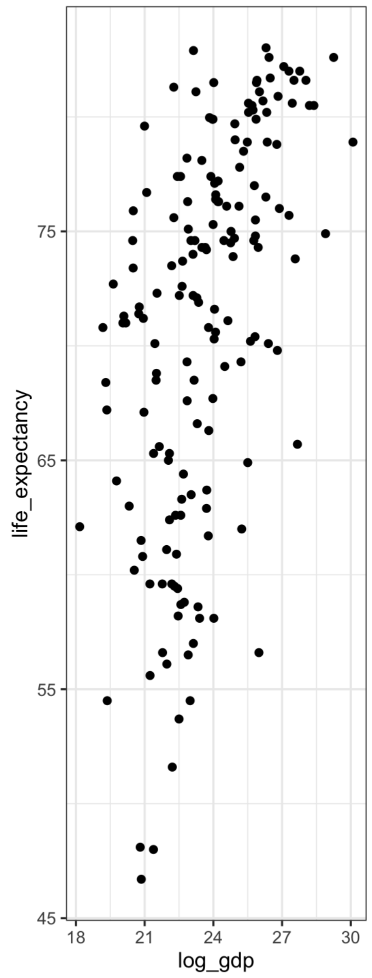
- **Review**: We have p variables for n observations $x_1, \ldots, x_n$,

- Compute the **distance / dissimilarity** between observations

- e.g. **Euclidean distance** between observations i and j

$$d(x_i, x_j) = \sqrt{(x_{i1} - x_{j1})^2 + \cdots + (x_{ip} - x_{jp})^2}$$

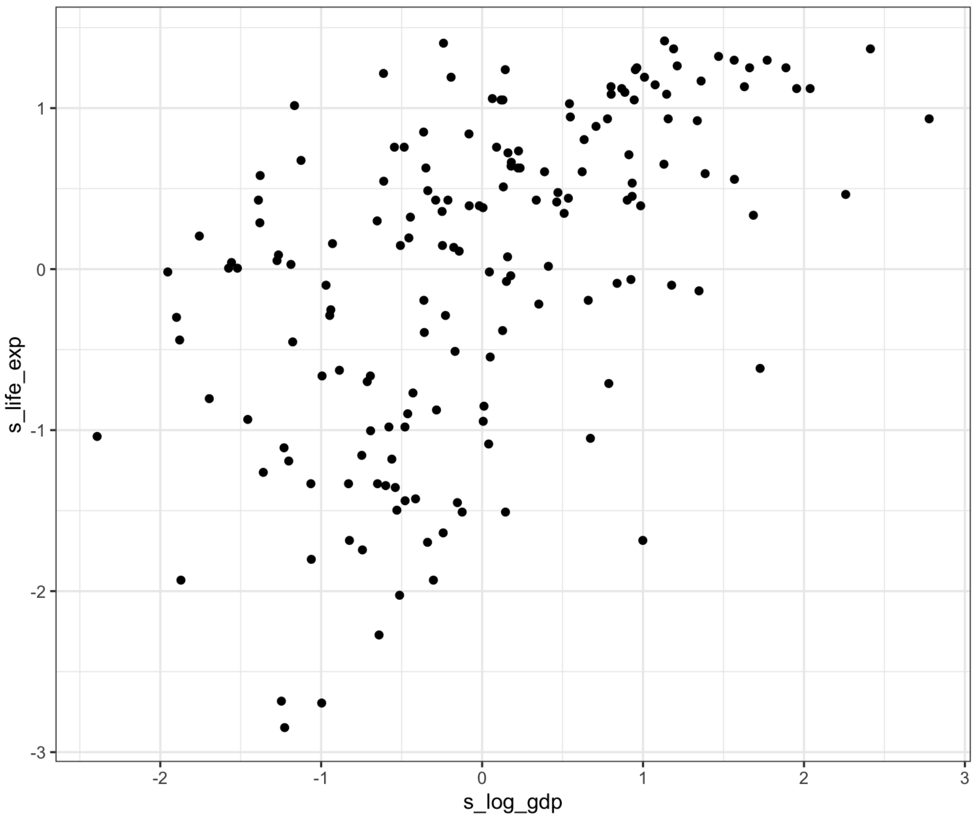
What are the distances between these countries using (log)GDP and life expectancy?



Remember to standardize!

```r
clean_gapminder <- clean_gapminder %>%
  mutate(s_log_gdp = as.numeric(scale(log_gdp
           center = TRUE, scale = TRUE)),
         s_life_exp =
           as.numeric(scale(life_expectancy,
           center = TRUE, scale = TRUE)))

clean_gapminder %>%
  ggplot(aes(x = s_log_gdp, y = s_life_exp))
  geom_point() +
  theme_bw() +
  coord_fixed()
```



Compute the distance matrix using `dist()`

- Compute pairwise Euclidean distance

```
gap_dist <- dist(dplyr::select(clean_gapminder, s_log_gdp,
                               s_life_exp))
```

- Returns an object of `dist` class - i.e., not a matrix
- Can convert to a matrix, then set the row and column names:

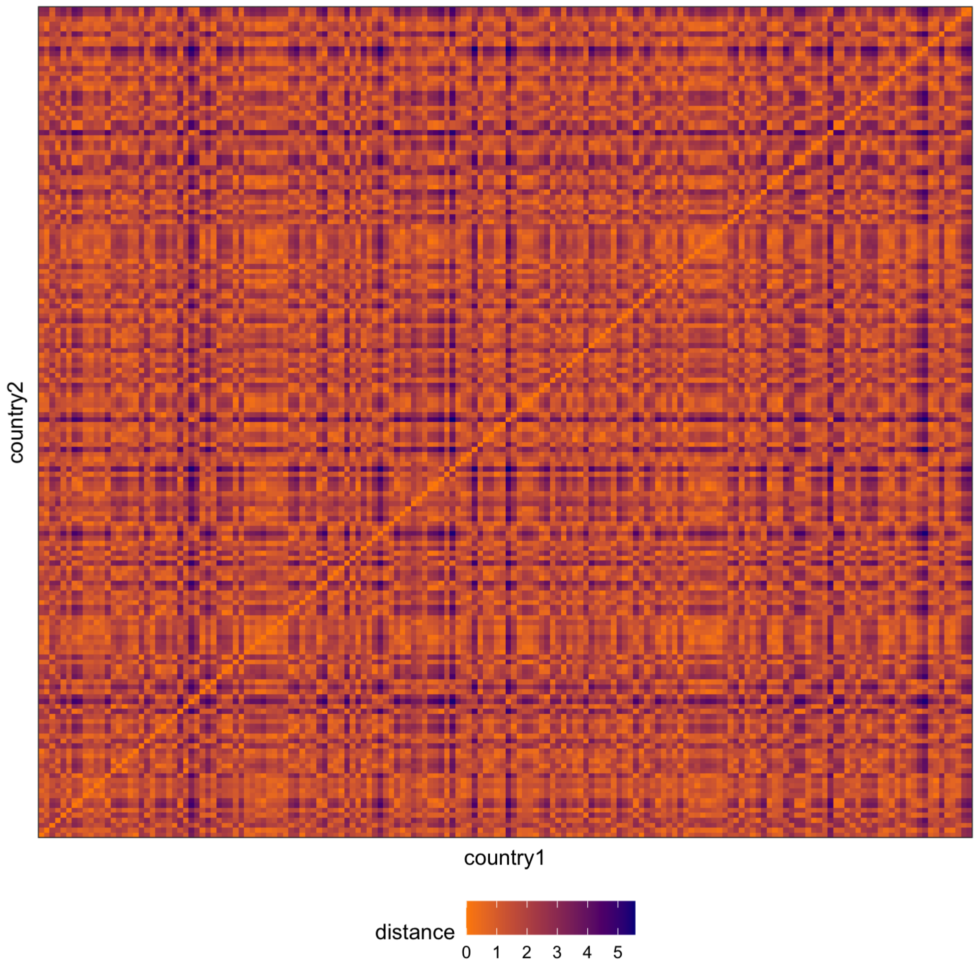
```
gap_dist_matrix <- as.matrix(gap_dist)
rownames(gap_dist_matrix) <- clean_gapminder$country
colnames(gap_dist_matrix) <- clean_gapminder$country
head(gap_dist_matrix[1:3, 1:3])
```

```
##          Albania  Algeria   Angola
## Albania 0.000000 1.116567 2.352044
## Algeria 1.116567 0.000000 2.166692
## Angola  2.352044 2.166692 0.000000
```

Plotting similarities

- Can convert to a long table for plotting with `ggplot`:

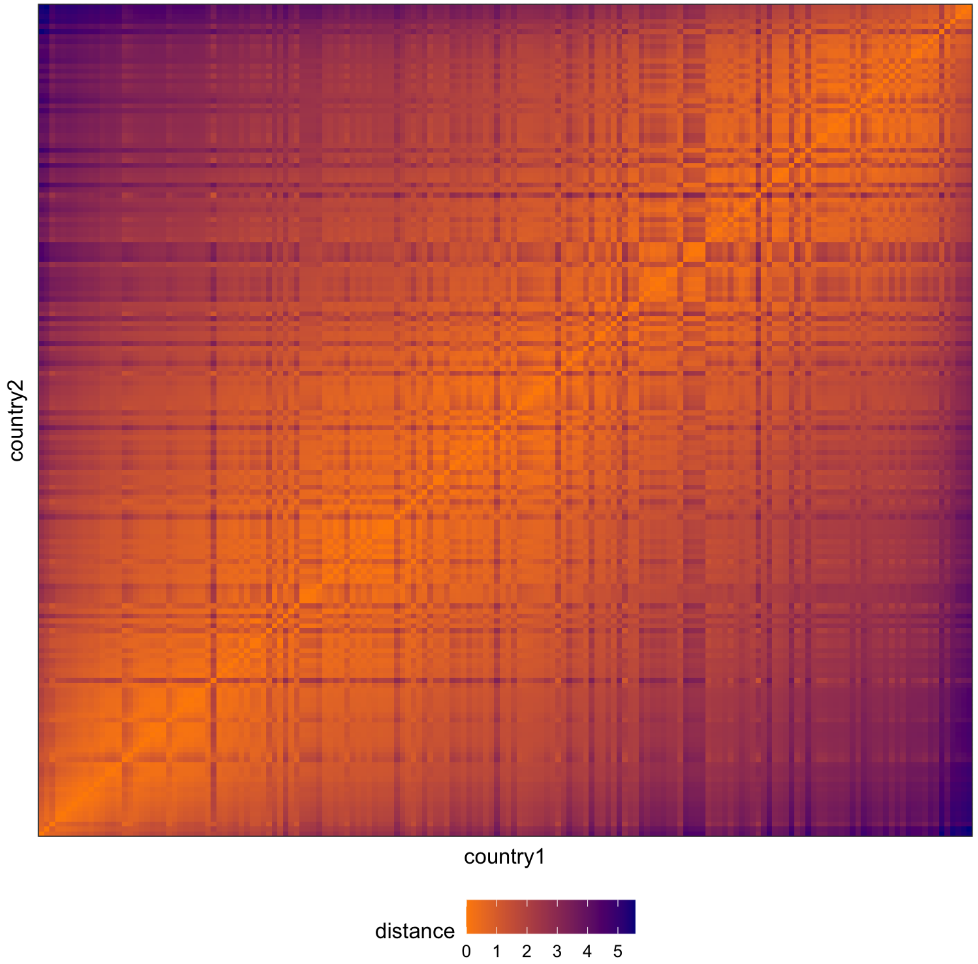
```
long_dist_matrix <-
  as_tibble(gap_dist_matrix) %>%
  mutate(country1 = rownames(gap_dist_matrix)) %>%
  pivot_longer(cols = -country1,
               names_to = "country2",
               values_to = "distance")
long_dist_matrix %>%
  ggplot(aes(x = country1, y = country2,
             fill = distance)) +
  geom_tile() +
  theme_bw() +
  theme(axis.text = element_blank(),
        axis.ticks = element_blank(),
        legend.position = "bottom") +
  scale_fill_gradient(low = "darkorange",
                      high = "darkblue")
```



Code interlude: arrange your heatmap with `seriation`

```
library(seriation)
gap_dist_seriate <- seriate(gap_dist)
gap_order <- get_order(gap_dist_seriate)
gap_countries_order <-
  as.character(clean_gapminder$country[gap_or

long_dist_matrix$country1 <-
  as_factor(long_dist_matrix$country1)
long_dist_matrix$country2 <-
  as_factor(long_dist_matrix$country2)
long_dist_matrix %>%
  mutate(country1 = fct_relevel(country1,
                                gap_countries_order),
         country2 = fct_relevel(country2,
                                gap_countries_order)) %>%
  ggplot(aes(x = country1, y = country2,
             fill = distance)) +
  geom_tile() + theme_bw() +
  theme(axis.text = element_blank(),
        axis.ticks = element_blank(),
        legend.position = "bottom") +
  scale_fill_gradient(low = "darkorange",
                      high = "darkblue")
```



(Agglomerative) Hierarchical clustering

Let's pretend all n observations are in their own cluster

- Step 1: Compute the pairwise dissimilarities between each cluster
  - e.g., distance matrix on previous slides
- Step 2: Identify the pair of clusters that are **least dissimilar**
- Step 3: Fuse these two clusters into a new cluster!
- **Repeat Steps 1 to 3 until all observations are in the same cluster**

"Bottom-up", agglomerative clustering that forms a **tree / hierarchy** of merging

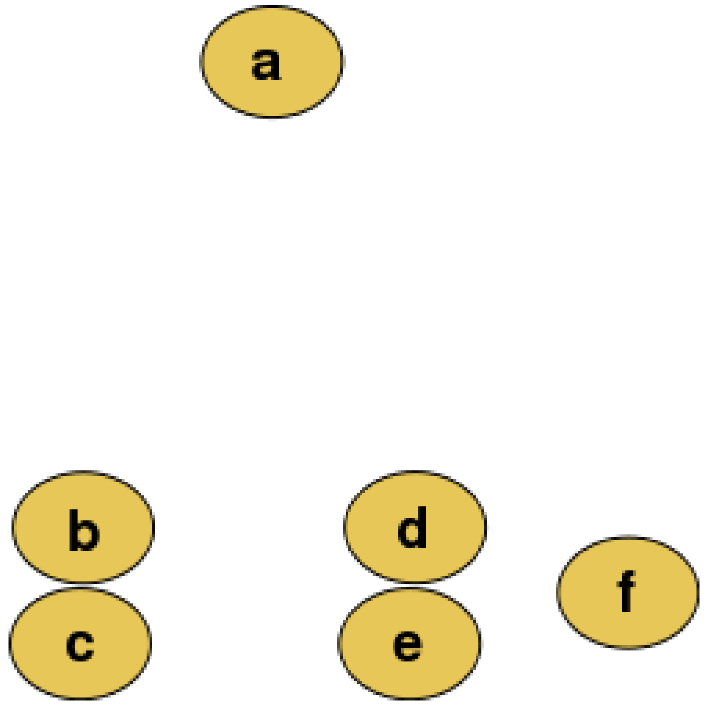
No mention of any randomness!

No mention of the number of clusters K!

(Agglomerative) Hierarchical clustering

Start with all observations in their own cluster

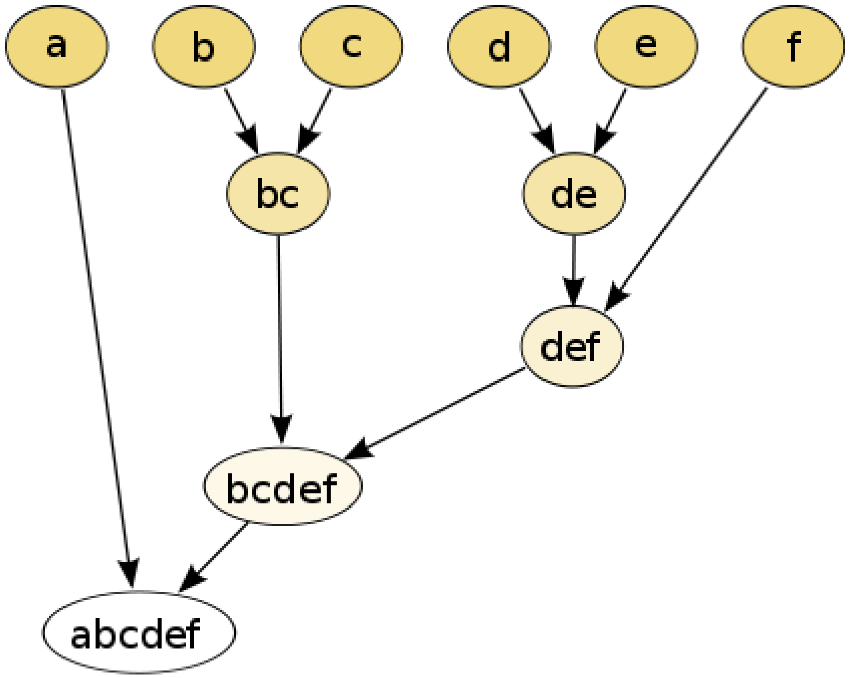
- Step 1: Compute the pairwise dissimilarities between each cluster
- Step 2: Identify the pair of clusters that are **least dissimilar**
- Step 3: Fuse these two clusters into a new cluster!
- **Repeat Steps 1 to 3 until all observations are in the same cluster**



(Agglomerative) Hierarchical clustering

Start with all observations in their own cluster

- Step 1: Compute the pairwise dissimilarities between each cluster
- Step 2: Identify the pair of clusters that are **least dissimilar**
- Step 3: Fuse these two clusters into a new cluster!
- **Repeat Steps 1 to 3 until all observations are in the same cluster**



Forms a **dendrogram** (typically displayed from bottom-up)

How do we define dissimilarity between clusters?

We know how to compute distance / dissimilarity between two observations

But how do we handle clusters?

- Dissimilarity between a cluster and an observation, or between two clusters

We need to choose a **linkage function**! Clusters are built up by **linking them together**

Compute all pairwise dissimilarities between observations in cluster 1 with observations in cluster 2

i.e. Compute the distance matrix between observations, $d(x_i, x_j)$ for $i \in C_1$ and $j \in C_2$

- **Complete linkage**: Use the **maximum** value of these dissimilarities: $\max\limits_{i \in C_1, j \in C_2} d(x_i, x_j)$
- **Single linkage**: Use the **minimum** value: $\min\limits_{i \in C_1, j \in C_2} d(x_i, x_j)$
- **Average linkage**: Use the **average** value: $\frac{1}{|C_1| \cdot |C_2|} \sum_{i \in C_1} \sum_{j \in C_2} d(x_i, x_j)$

Define dissimilarity between two clusters **based on our initial dissimilarity matrix between observations**

Complete linkage example

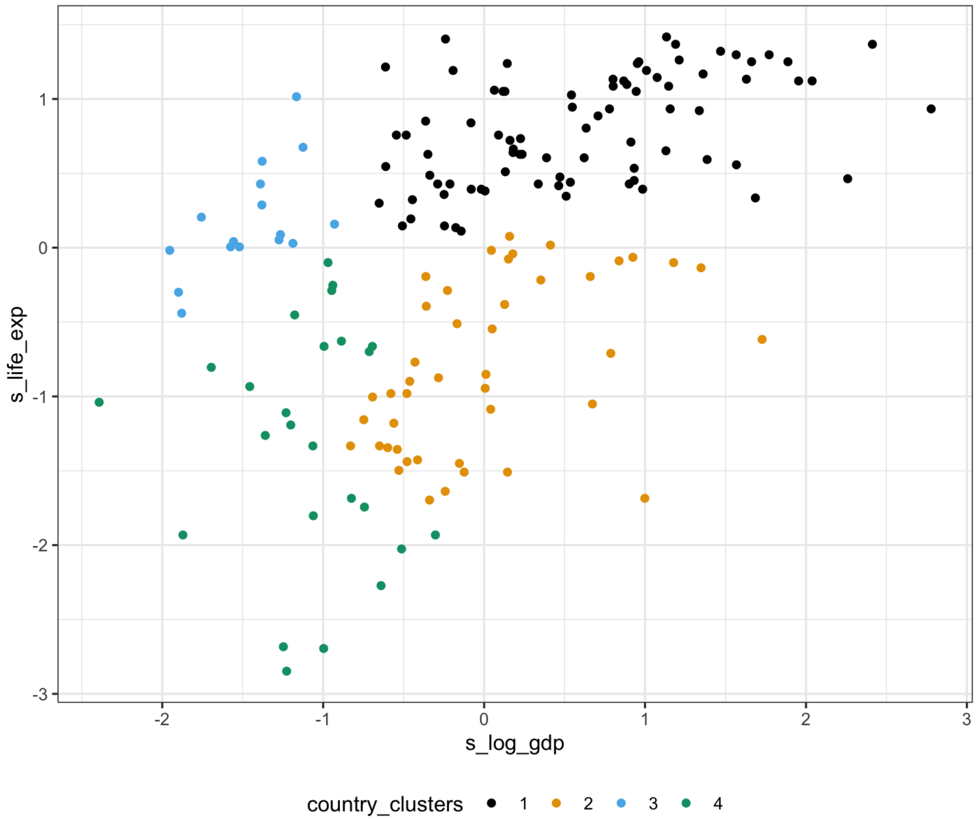
- Use the `hclust` function with a `dist()` object
- Uses `complete` linkage by default

```
gap_complete_hclust <-
  hclust(gap_dist, method = "complete")
```

- Need to use `cutree()` to return cluster labels:

```
clean_gapminder %>%
  mutate(country_clusters =
         as.factor(cutree(gap_complete_hclust,
                          k = 4))) %>%
  ggplot(aes(x = s_log_gdp, y = s_life_exp,
             color = country_clusters)) +
  geom_point() +
  ggthemes::scale_color_colorblind() +
  theme_bw() +
  theme(legend.position = "bottom")
```

Returns *compact* clusters, similar to K-means

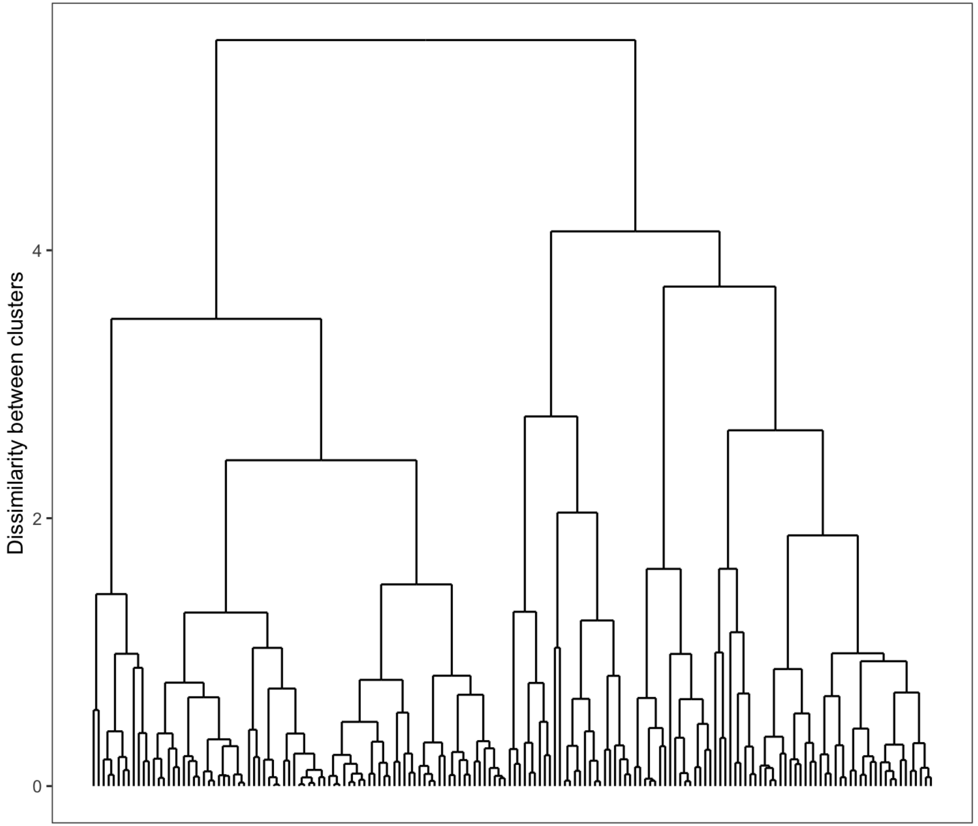


What are we cutting? Dendrograms

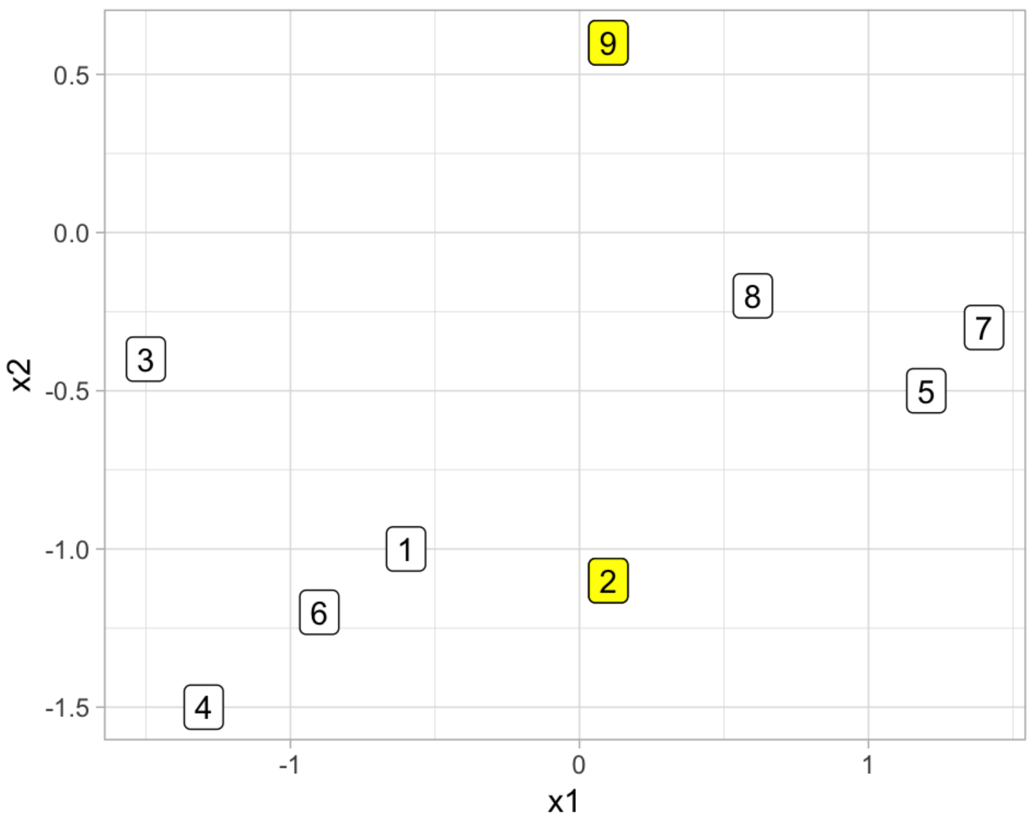
Use the ggdendro package (instead of `plot()`)

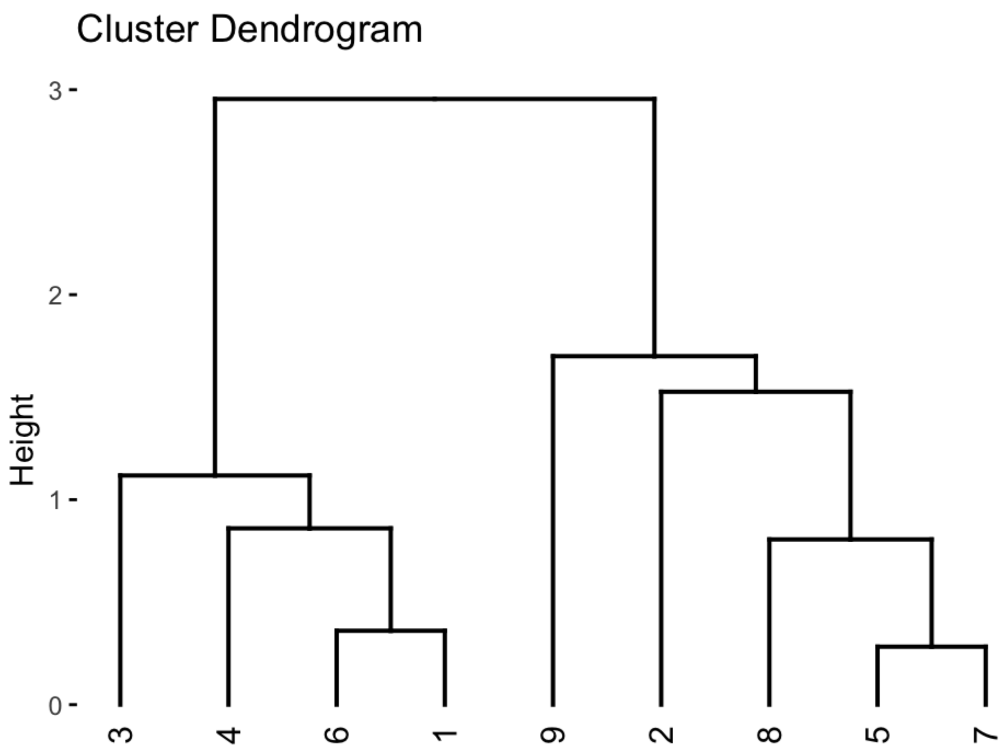
```
library(ggdendro)
ggdendrogram(gap_complete_hclust,
             theme_dendro = FALSE,
             labels = FALSE,
             leaf_labels = FALSE) +
  labs(y = "Dissimilarity between clusters") +
  theme_bw() +
  theme(axis.text.x = element_blank(),
        axis.title.x = element_blank(),
        axis.ticks.x = element_blank(),
        panel.grid = element_blank())
```

- Each **leaf** is one observation
- **Height of branch indicates dissimilarity between clusters**
    - (After first step) Horizontal position along x-axis means nothing



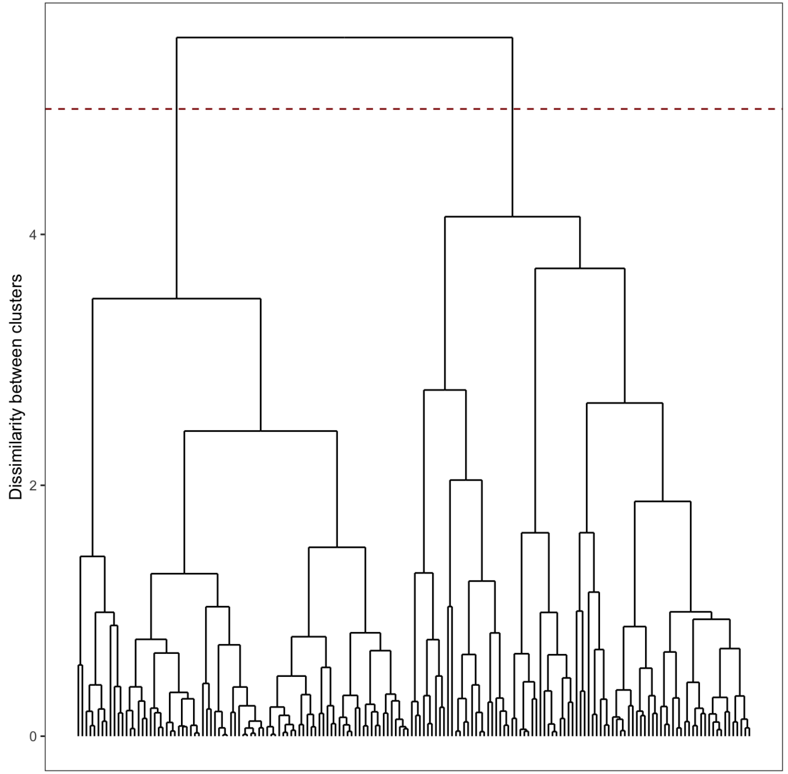
Textbook example



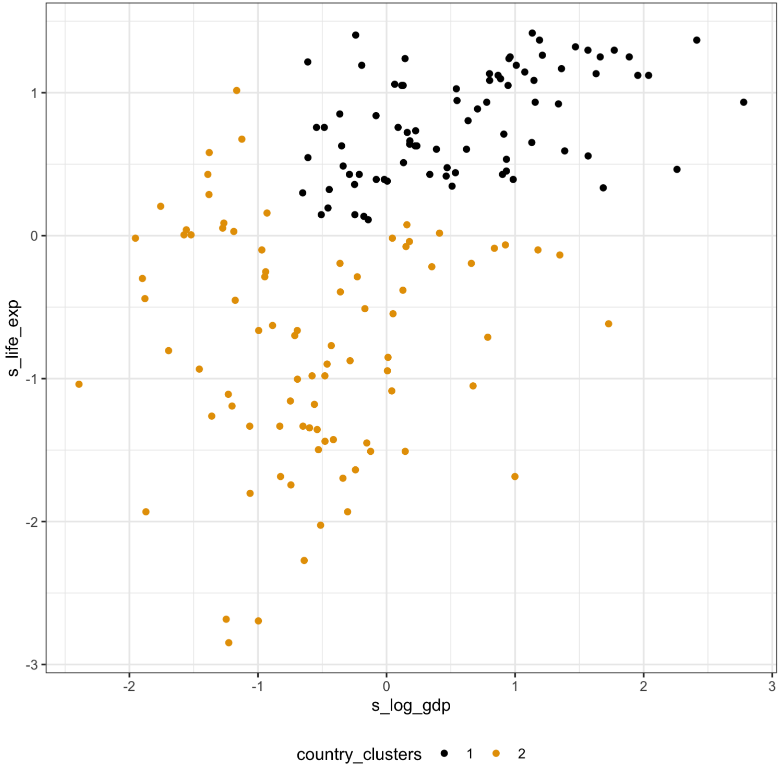


Cut dendrograms to obtain cluster labels

Specify the height to cut with `h` instead of `k`

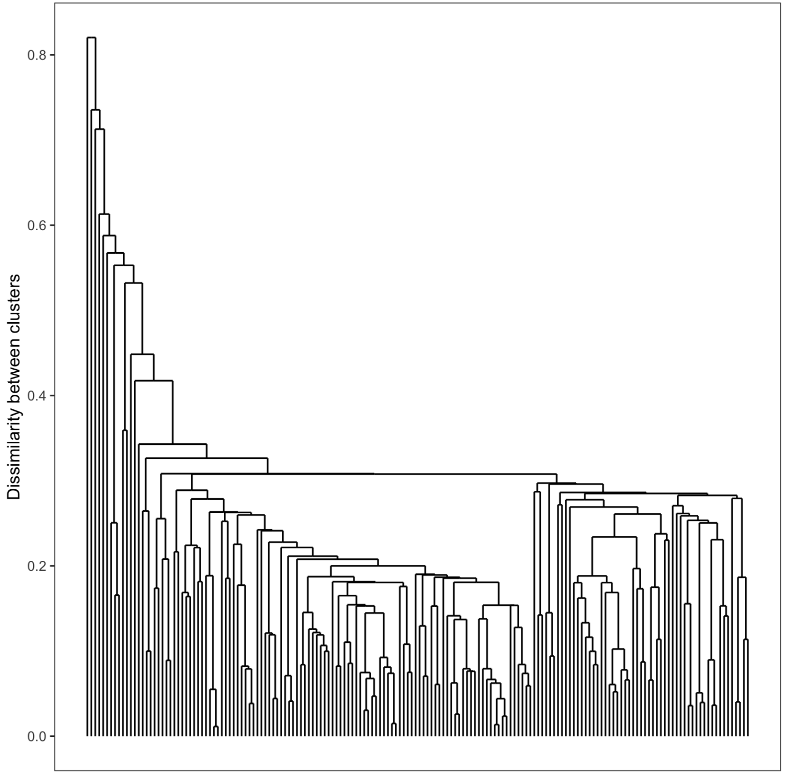



```r
cutree(gap_complete_hclust, h = 5)
```

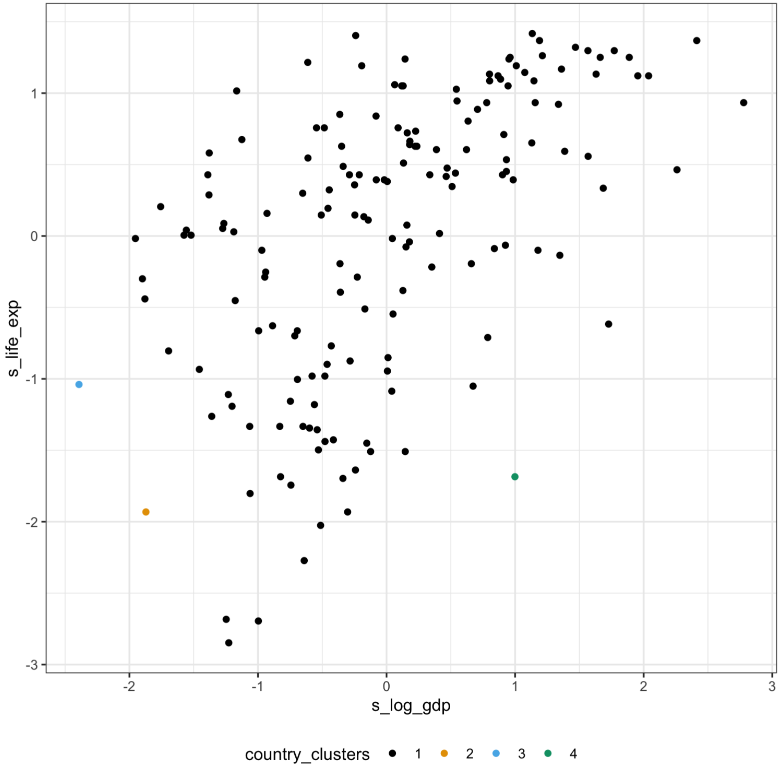


Single linkage example

Change the `method` argument to `single`

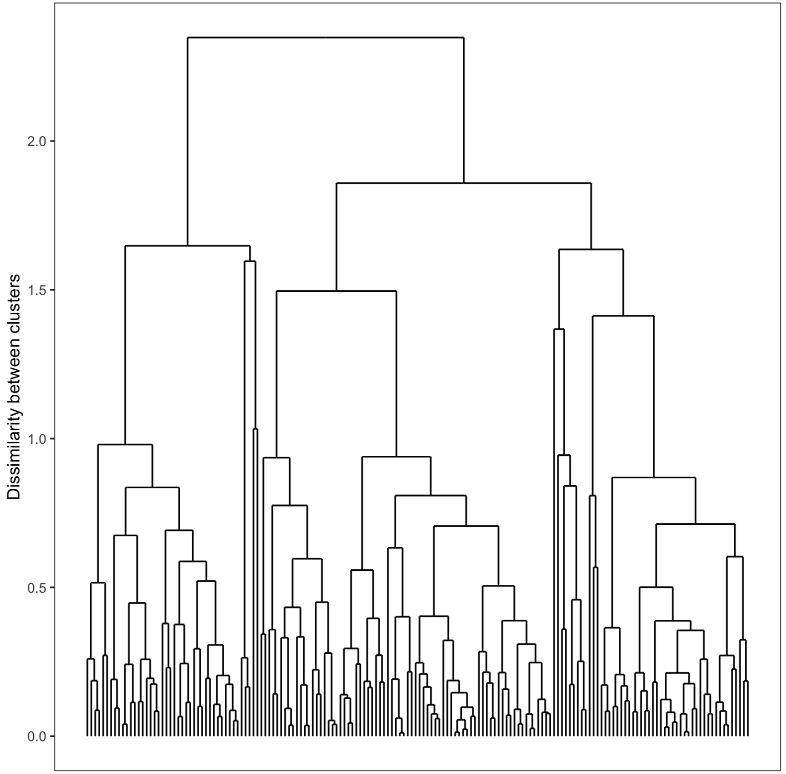


Results in a **chaining** effect

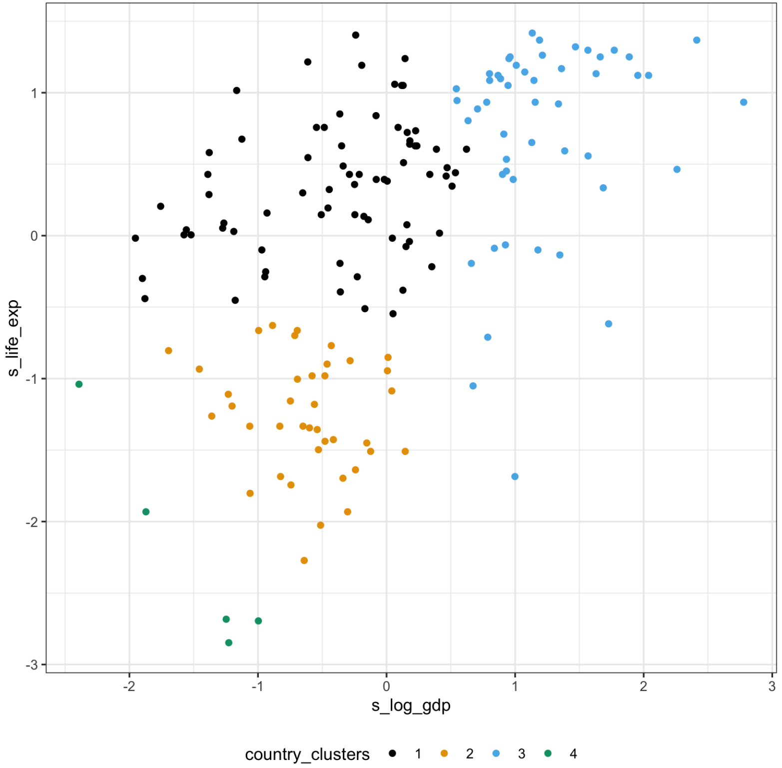


Average linkage example

Change the `method` argument to `average`



Closer to `complete` but varies in compactness

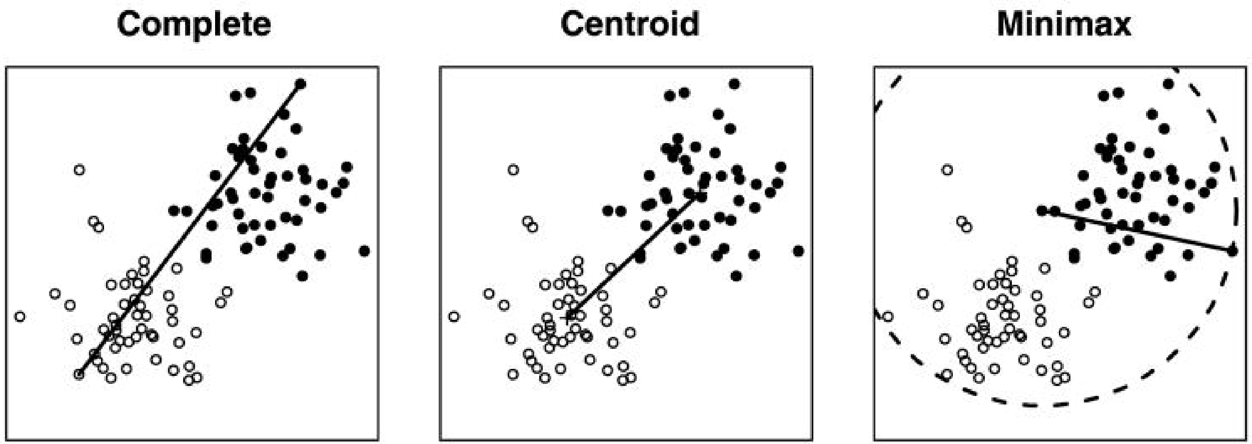


More linkage functions

- **Centroid linkage**: Computes the dissimilarity between the centroid for cluster 1 and the centroid for cluster 2
    - i.e. distance between the averages of the two clusters
    - use `method = centroid`
- **Ward's linkage**: Merges a pair of clusters to minimize the within-cluster variance
    - i.e. aim is to minimize the objective function from K-means
    - can use `ward.D` or `ward.D2` (different algorithms)

Minimax linkage

- Each cluster is defined **by a prototype** observation (most representative)
- **Identify the point whose farthest point is closest** (hence the minimax)

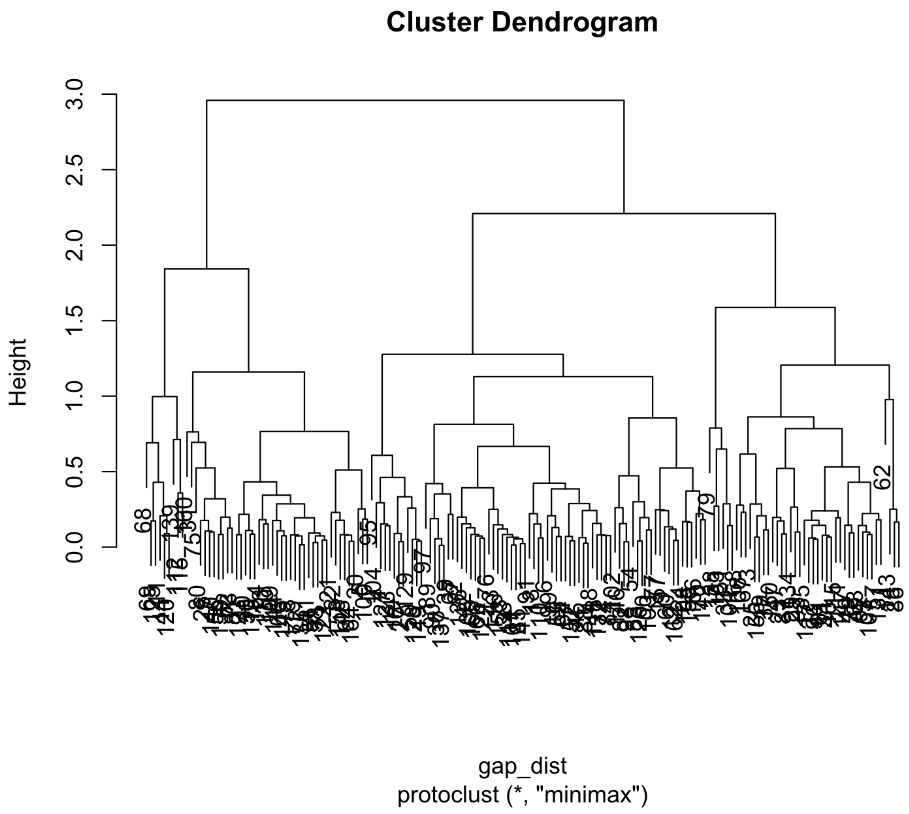


- Use this minimum-maximum distance as the measure of cluster dissimilarity
- Dendogram interpretation: each point point is $\leq h$ in dissimilarity to the **prototype** of cluster
- **Cluster centers are chosen among the observations themselves - hence prototype**

Minimax linkage example

- Easily done in `R` via the `protoclust` package
- Use the `protoclust()` function to apply the clustering to the `dist()` object

```
library(protoclust)
gap_minimax <- protoclust(gap_dist)
plot(gap_minimax)
# ggdendrogram was having issues
# with protoclust... so base R :(
```

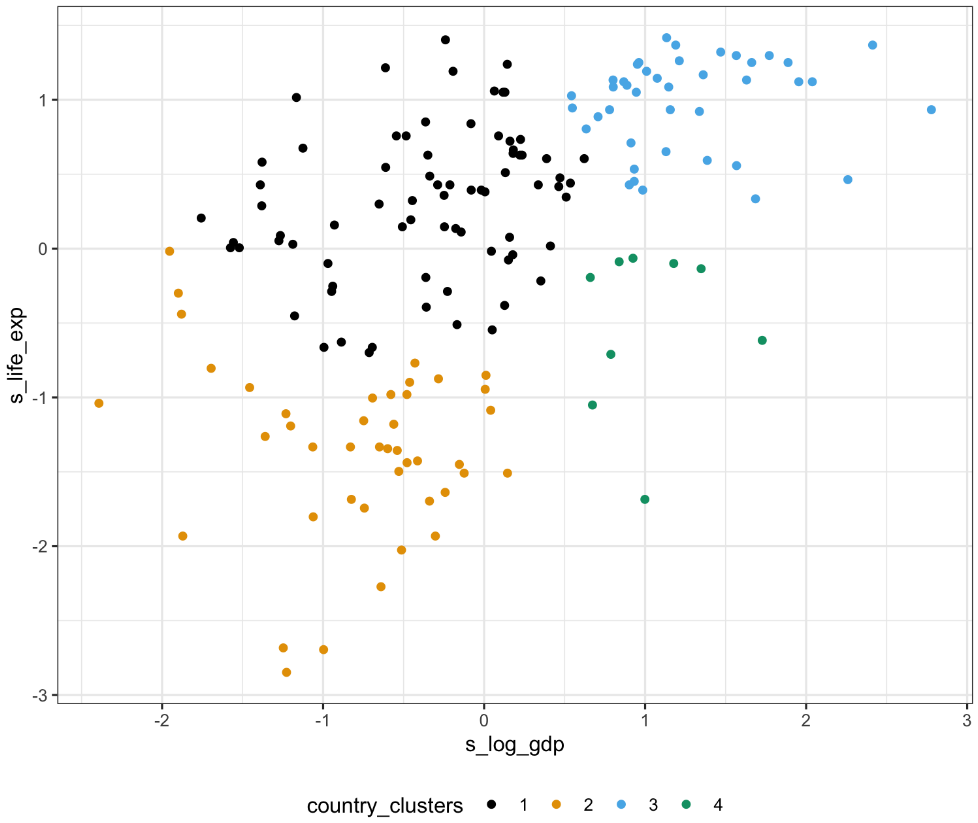


Minimax linkage example

- Use the `protocut()` function to make the cut
- But then access the cluster labels `cl`

```
minimax_country_clusters <-
  protocut(gap_minimax, k = 4)

clean_gapminder %>%
  mutate(country_clusters =
    as.factor(minimax_country_clusters$cl)) %
  ggplot(aes(x = s_log_gdp, y = s_life_exp,
             color = country_clusters)) +
  geom_point() +
  ggthemes::scale_color_colorblind() +
  theme_bw() +
  theme(legend.position = "bottom")
```



Minimax linkage example

- Want to check out the prototypes for the three clusters
- `protocut` returns the indices of the prototypes (in order of the cluster labels)

```
minimax_country_clusters$protos
```

```
## [1]  91 150  26 115
```

- View these country rows using `slice`:

```
clean_gapminder %>%
  dplyr::select(country, gdp, life_expectancy,
                population, infant_mortality) %>%
  slice(minimax_country_clusters$protos)
```

```
## # A tibble: 4 × 5
##   country                 gdp life_expectancy population infant_mortality
##   <fct>                 <dbl>           <dbl>      <dbl>            <dbl>
## 1 Macedonia, FYR   4713514754            75.6    2065888              7.5
## 2 Togo             1658132200            59.6    6566179             57.9
## 3 Canada         894251850391            81.6   34499905              4.7
## 4 Pakistan       118790417253            64.9  173669648             72.1
```

Wrapping up...

- How might this clustering example help us understand global public health?

```
table("Clusters" = minimax_country_clusters$cl,
      "Continents" = clean_gapminder$continent)
```

```
##          Continents
## Clusters Africa Americas Asia Europe Oceania
##        1     10       19   24     20       2
##        2     36        1    0      0       6
##        3      0        9   13     18       1
##        4      3        0    5      1       0
```

- Can see countries on different continents tend to fall within particular clusters...
- **We can easily include more variables** - just changes our distance matrix
- But we might want to explore **soft** assignments instead...