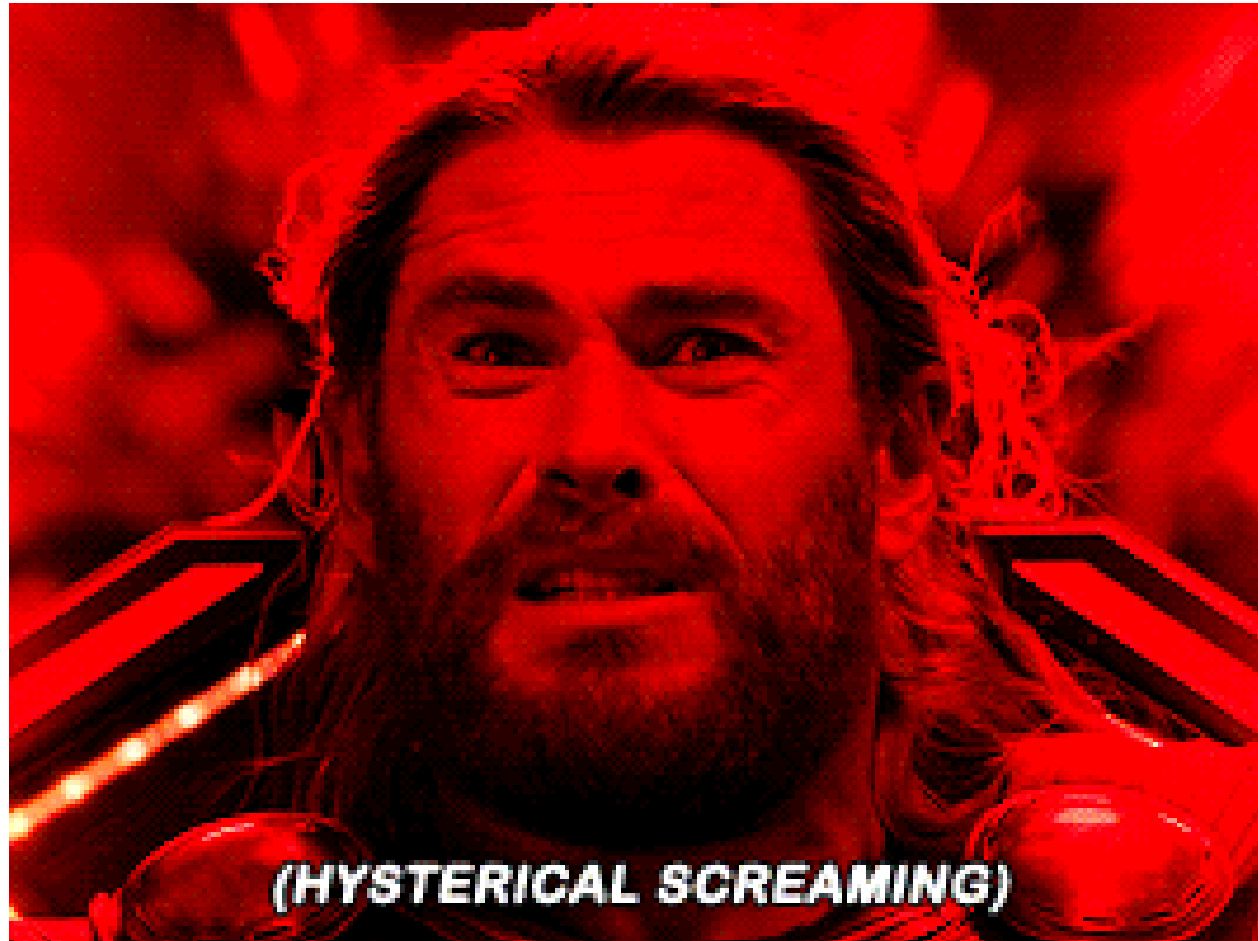


# Clustering

## K-means

June 14th, 2022

Brace yourselves



# Into statistical learning with unsupervised learning

What is **statistical learning**? Preface of Introduction to Statistical Learning with Applications in R (ISLR):

■ *refers to a set of tools for modeling and understanding complex datasets*

What is **unsupervised learning**?

We have  $p$  variables for  $n$  observations  $x_1, \dots, x_n$ , and for observation  $i$ :

$$x_{i1}, x_{i2}, \dots, x_{ip} \sim P$$

- $P$  is a  $p$ -dimensional distribution that we might not know much about *a priori*.
- *unsupervised*: none of the variables are **response** variables, i.e., there are no labeled data

Think of unsupervised learning as **an extension of EDA...**

- $\Rightarrow$  **there is no unique right answer!**

# What is clustering (aka cluster analysis)?

## ISLR 10.3:

very broad set of techniques for finding subgroups, or clusters, in a dataset

- observations **within** clusters are **more similar** to each other,
- observations **in different** clusters are **more different** from each other

How do we define **distance** / **dissimilarity** between observations?

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

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

### Units matter!

- one variable may *dominate* others when computing Euclidean distance because its range is much larger
- can standardize each variable / column of dataset to have mean 0 and standard deviation 1 with `scale()`
- **but we may value the separation in that variable!** (so just be careful...)

# What's the clustering objective?

- $C_1, \dots, C_K$  are *sets* containing indices of observations in each of the  $K$  clusters
  - if observation  $i$  is in cluster  $k$ , then  $i \in C_k$
- We want to minimize the **within-cluster variation**  $W(C_k)$  for each cluster  $C_k$  and solve:

$$\underset{C_1, \dots, C_K}{\text{minimize}} \left\{ \sum_{k=1}^K W(C_k) \right\}$$

- Can define using the **squared Euclidean distance** ( $|C_k| = n_k = \#$  observations in cluster  $k$ )

$$W(C_k) = \frac{1}{|C_k|} \sum_{i, j \in C_k} d(x_i, x_j)^2$$

- Commonly referred to as the within-cluster sum of squares (WSS)

**So how can we solve this?**

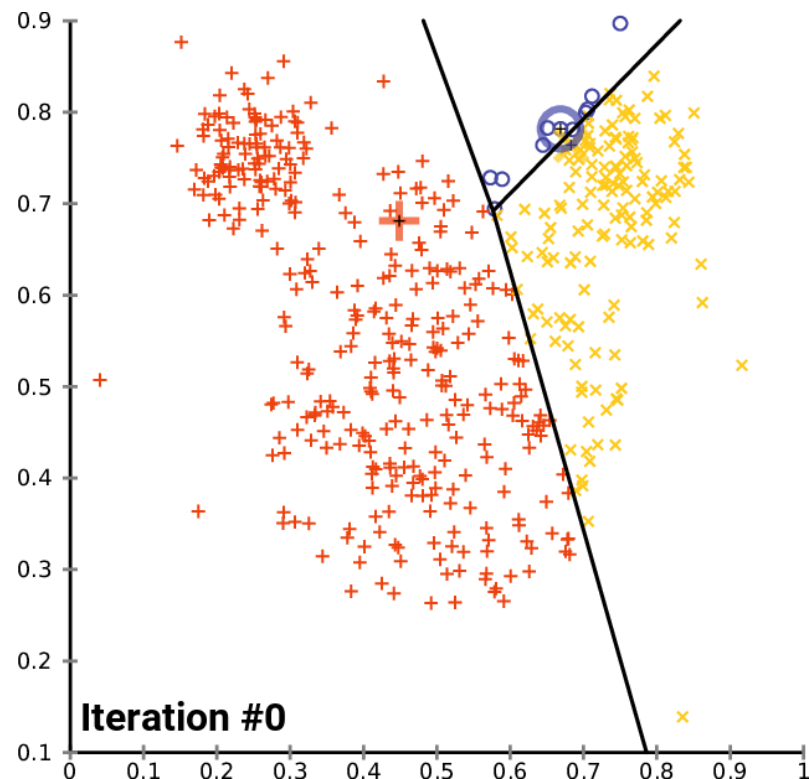
# Lloyd's algorithm

- 1) Choose  $K$  random centers, aka **centroids**
- 2) Assign each observation closest center (using Euclidean distance)
- 3) Repeat until cluster assignment stop changing:
  - Compute new centroids as the averages of the updated groups
  - Reassign each observations to closest center

**Converges to a local optimum**, not the global

**Results will change from run to run** (set the seed!)

**Takes  $K$  as an input!**



# Gapminder data

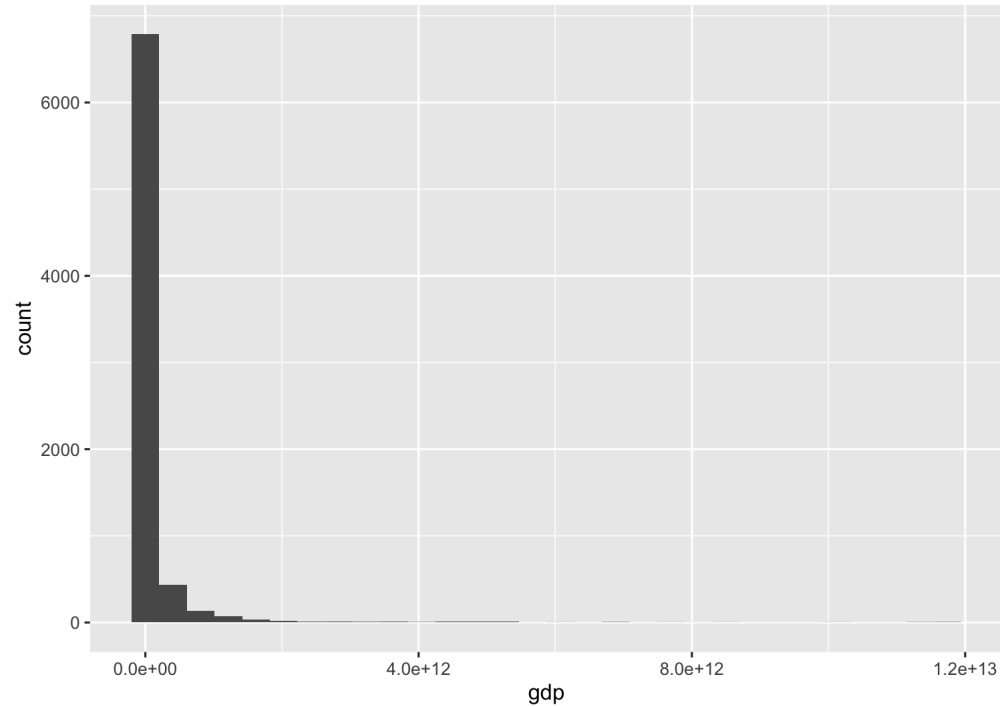
Health and income outcomes for 184 countries from 1960 to 2016 from the famous [Gapminder project](#)

```
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
```

# GDP is severely skewed right...

```
gapminder %>% ggplot(aes(x = gdp)) + geom_histogram()
```





# Some initial cleaning...

- 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

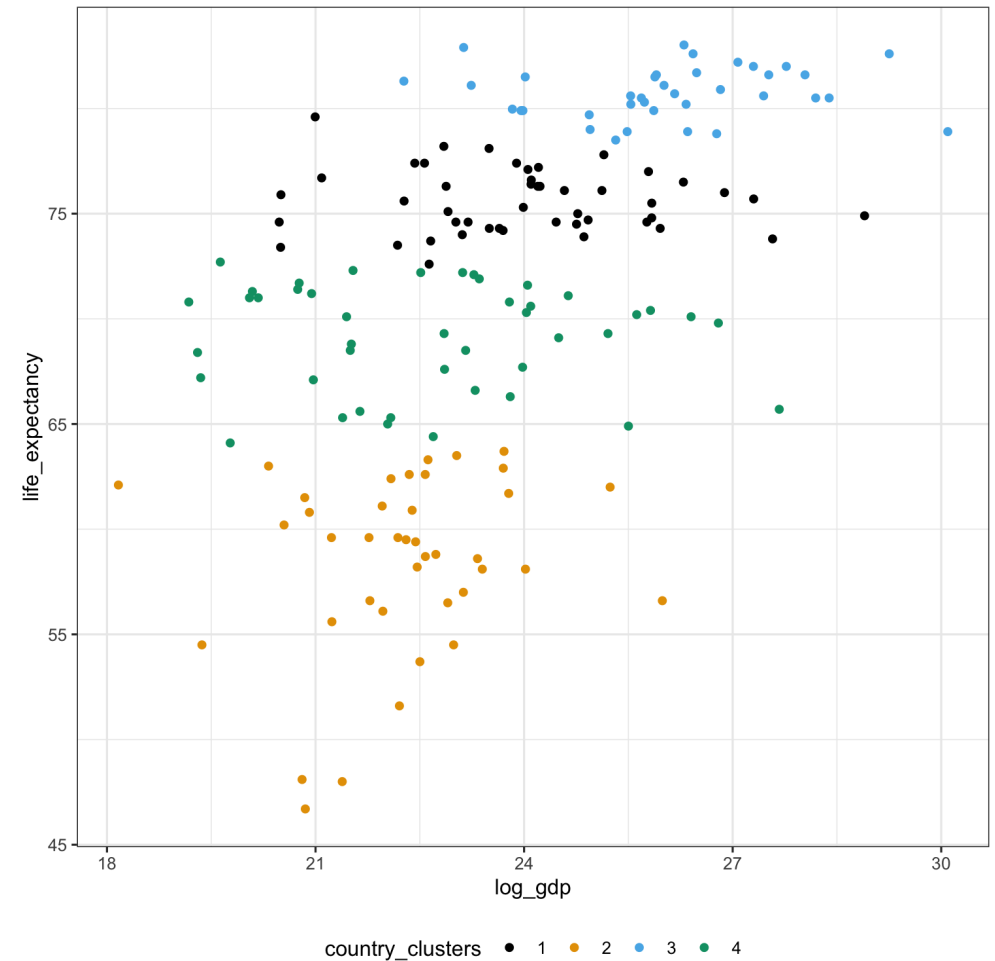
```
clean_gapminder <- gapminder %>%  
  filter(year == 2011, !is.na(gdp)) %>%  
  mutate(log_gdp = log(gdp))  
clean_gapminder
```

```
## # 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
```

# K-means clustering example (gdp and life\_expectancy)

- Use the `kmeans()` function, **but must provide number of clusters  $K$**

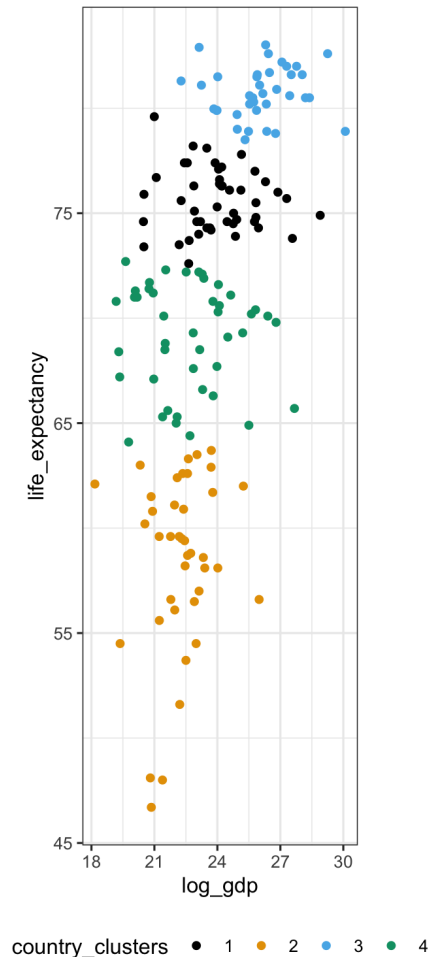
```
init_kmeans <-  
  kmeans(dplyr::select(clean_gapminder,  
                        log_gdp, life_expectan  
                        algorithm = "Lloyd", centers = 4,  
                        nstart = 1)  
  
clean_gapminder %>%  
  mutate(country_clusters =  
    as.factor(init_kmeans$cluster)) %>  
  ggplot(aes(x = log_gdp, y = life_expectancy  
             color = country_clusters)) +  
  geom_point() +  
  ggthemes::scale_color_colorblind() +  
  theme_bw() +  
  theme(legend.position = "bottom")
```



# Careful with units...

- Use the `coord_fixed()` so that the axes match with unit scales

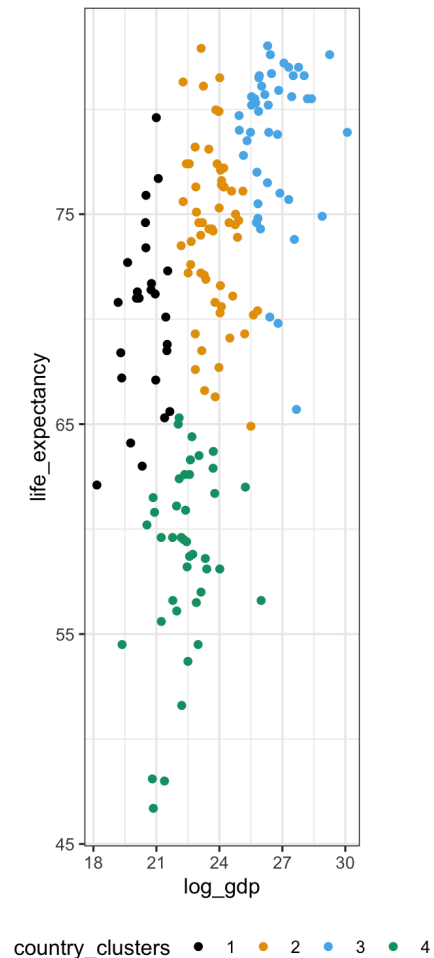
```
clean_gapminder %>%  
  mutate(country_clusters =  
    as.factor(init_kmeans$cluster)) %>  
  ggplot(aes(x = log_gdp, y = life_expectancy,  
             color = country_clusters)) +  
  geom_point() +  
  ggthemes::scale_color_colorblind() +  
  theme_bw() +  
  theme(legend.position = "bottom") +  
  coord_fixed()
```



# Standardize the variables!

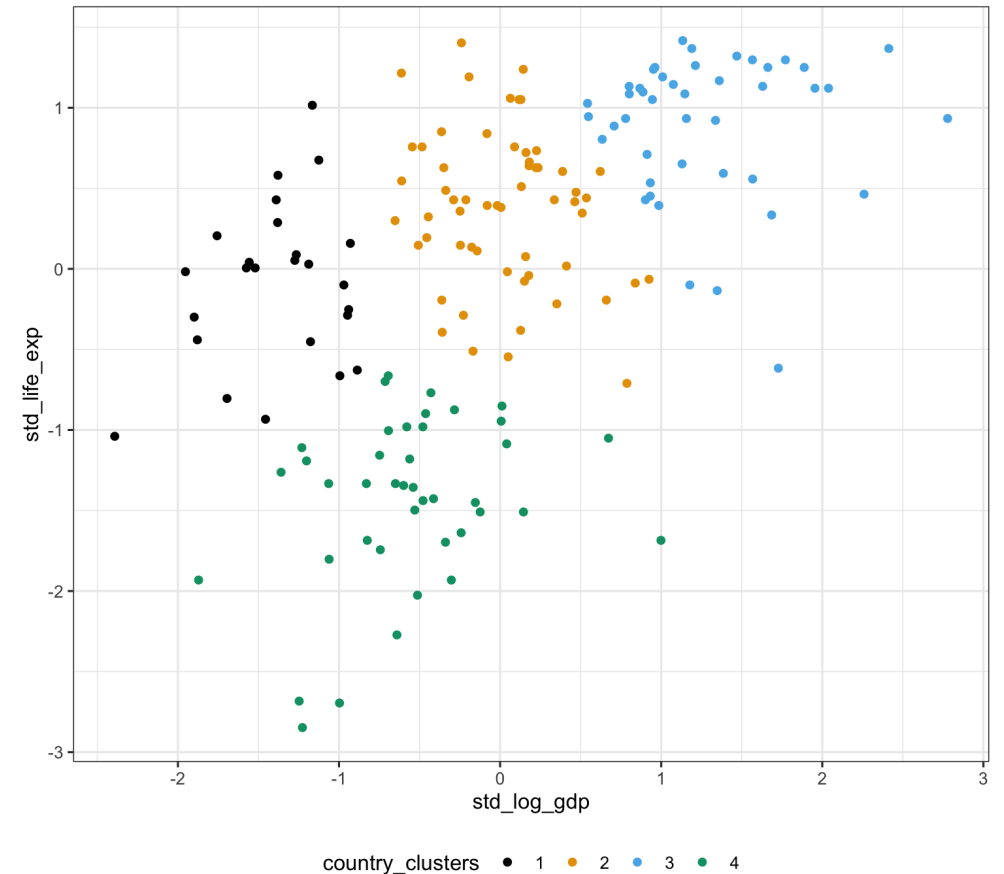
- Use the `scale()` function to first **standardize the variables**,  $\frac{\text{value} - \text{mean}}{\text{standard deviation}}$

```
clean_gapminder <- clean_gapminder %>%  
  mutate(std_log_gdp = as.numeric(scale(log_gdp)),  
         std_life_exp = as.numeric(scale(life_expectancy)))  
std_kmeans <-  
  kmeans(dplyr::select(clean_gapminder, std_log_gdp, std_life_exp),  
        algorithm = "Lloyd", centers = 4, ns  
        clean_gapminder %>%  
  mutate(country_clusters =  
    as.factor(std_kmeans$cluster)) %>%  
  ggplot(aes(x = log_gdp, y = life_expectancy,  
             color = country_clusters)) +  
  geom_point() +  
  ggthemes::scale_color_colorblind() +  
  theme_bw() +  
  theme(legend.position = "bottom") +  
  coord_fixed()
```



# Standardize the variables!

```
clean_gapminder %>%  
  mutate(country_clusters =  
    as.factor(std_kmeans$cluster)) %>%  
  ggplot(aes(x = std_log_gdp, y = std_life_exp,  
             color = country_clusters)) +  
  geom_point() +  
  ggthemes::scale_color_colorblind() +  
  theme_bw() +  
  theme(legend.position = "bottom") +  
  coord_fixed()
```



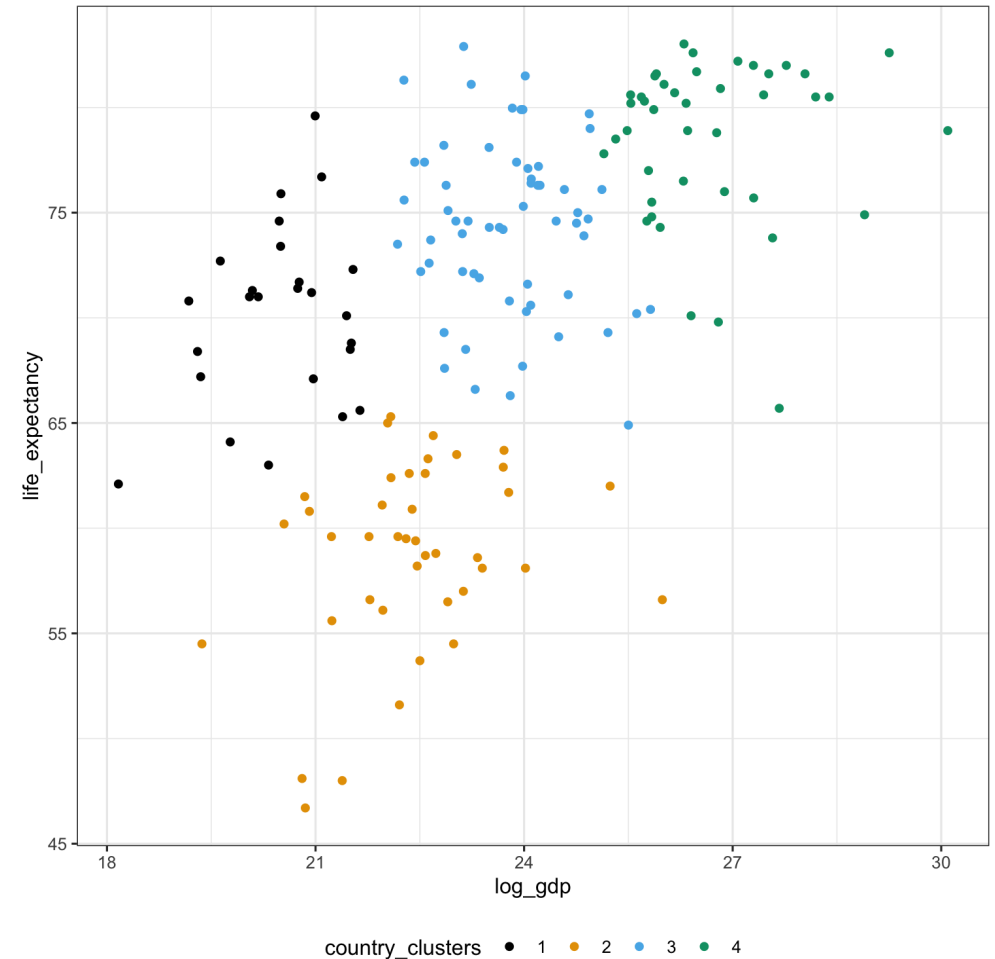
# And if we run it again?

We get different clustering results!

```
another_kmeans <-  
  kmeans(dplyr::select(clean_gapminder, std_l  
    algorithm = "Lloyd", centers = 4, ns  
  
clean_gapminder %>%  
  mutate(country_clusters =  
    as.factor(another_kmeans$cluster))  
  ggplot(aes(x = log_gdp, y = life_expectancy  
    color = country_clusters)) +  
  geom_point() +  
  ggthemes::scale_color_colorblind() +  
  theme_bw() +  
  theme(legend.position = "bottom")
```

**Results depend on initialization**

**Keep in mind: the labels / colors are arbitrary**



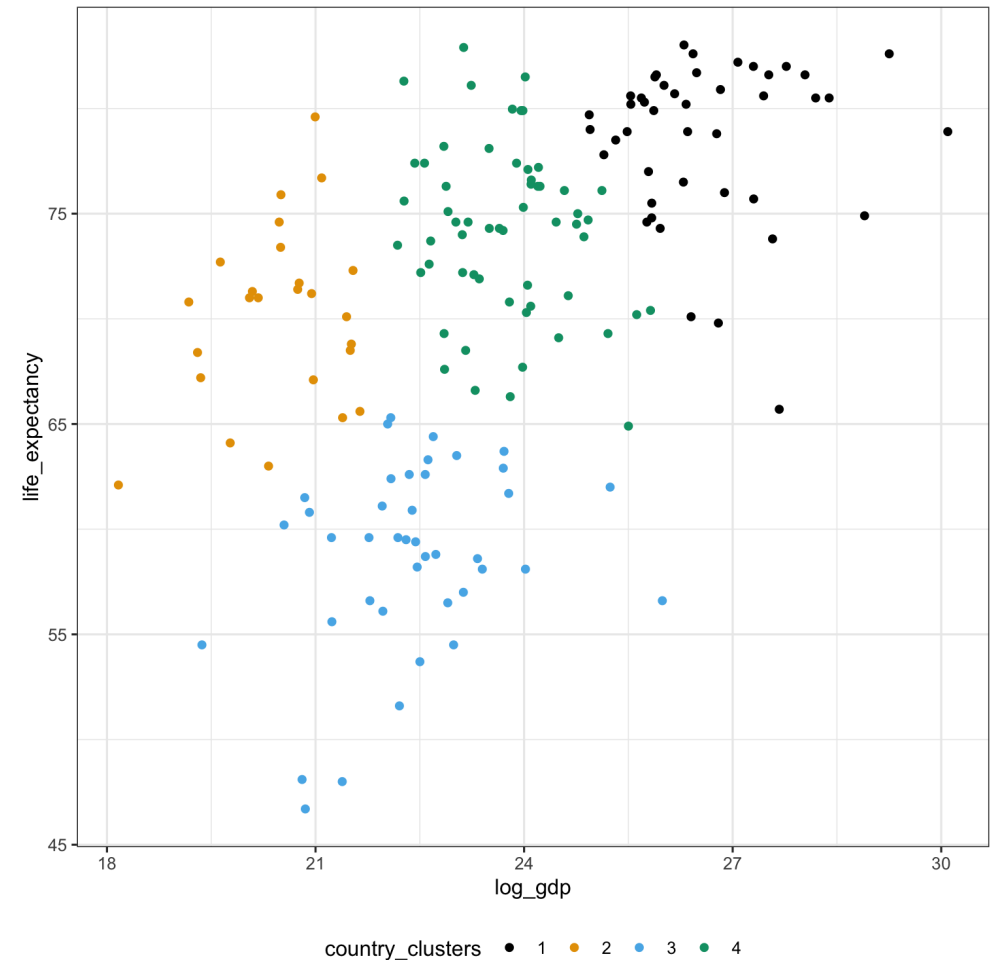
# Fix randomness issue with `nstart`

Run the algorithm `nstart` times, then **pick the results with lowest total within-cluster variation**

(total WSS =  $\sum_k^K W(C_k)$ )

```
nstart_kmeans <-  
  kmeans(dplyr::select(clean_gapminder,  
                        std_log_gdp, std_life_  
                        algorithm = "Lloyd", centers = 4,  
                        nstart = 30)
```

```
clean_gapminder %>%  
  mutate(country_clusters =  
    as.factor(nstart_kmeans$cluster))  
ggplot(aes(x = log_gdp, y = life_expectancy,  
           color = country_clusters)) +  
  geom_point() +  
  ggthemes::scale_color_colorblind() +  
  theme_bw() +  
  theme(legend.position = "bottom")
```

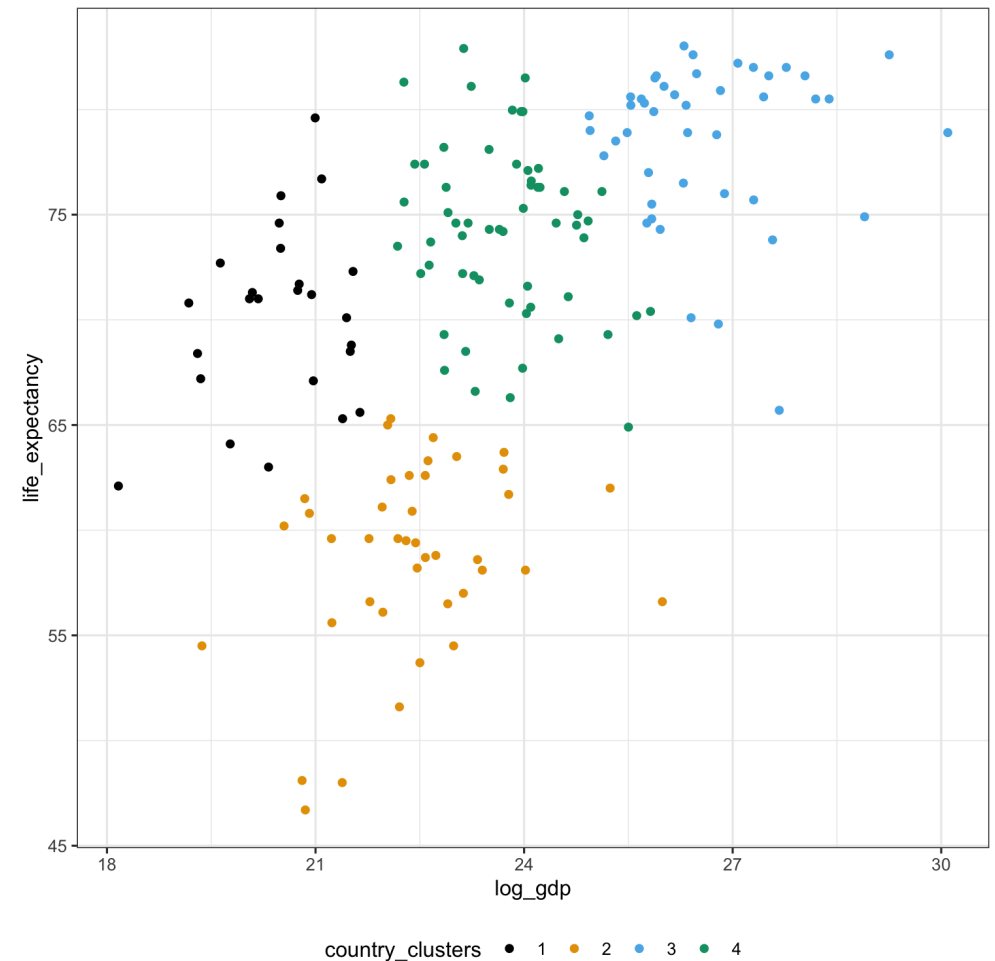


## By default R uses Hartigan and Wong algorithm

Updates based on changing a single observation

**Computational advantages over re-computing distances for every observation**

```
default_kmeans <-  
  kmeans(dplyr::select(clean_gapminder,  
                        std_log_gdp, std_life_  
                        algorithm = "Hartigan-Wong",  
                        centers = 4, nstart = 30)  
  
clean_gapminder %>%  
  mutate(country_clusters =  
           as.factor(default_kmeans$cluster))  
  ggplot(aes(x = log_gdp, y = life_expectancy,  
             color = country_clusters)) +  
  geom_point() +  
  ggthemes::scale_color_colorblind() +  
  theme_bw() +  
  theme(legend.position = "bottom")
```



Very little differences for our purposes...



## Better alternative to `nstart`: **K-means++**

Pick a random observation to be the center  $c_1$  of the first cluster  $C_1$

- This initializes a set  $Centers = \{c_1\}$

Then for each remaining cluster  $c^* \in 2, \dots, K$ :

- For each observation (that is not a center), compute  $D(x_i) = \min_{c \in Centers} d(x_i, c)$ 
  - Distance between observation and its closest center  $c \in Centers$
- Randomly pick a point  $x_i$  with probability:  $p_i = \frac{D^2(x_i)}{\sum_{j=1}^n D^2(x_j)}$
- As distance to closest center increases  $\Rightarrow$  probability of selection increases
- Call this randomly selected observation  $c^*$ , update  $Centers = Centers \cup c^*$ 
  - Same as `centers = c(centers, c_new)`

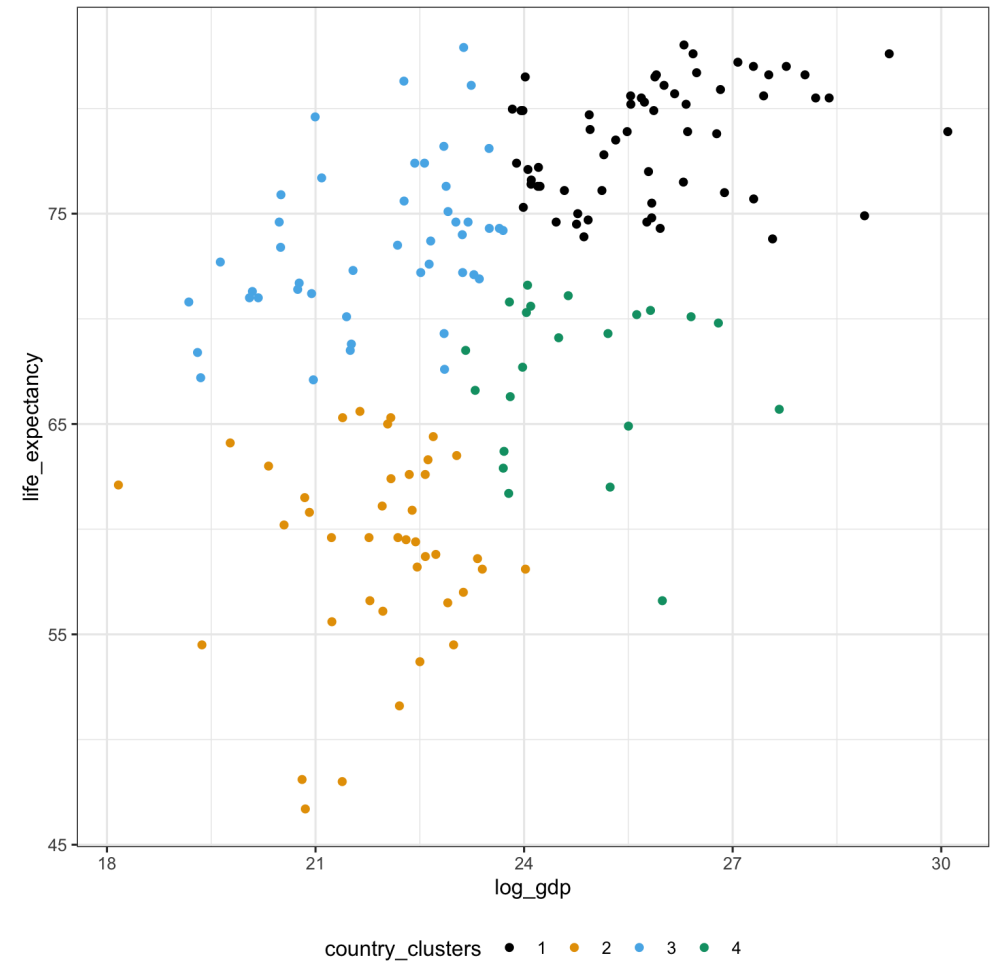
**Then run  $K$ -means using these  $Centers$  as the starting points**

# K-means++ in R using **flexclust**

```
library(flexclust)
init_kmeanspp <-
  kcca(dplyr::select(clean_gapminder,
                     std_log_gdp, std_life_ex
                     control = list(initcent = "kmeanspp"))

clean_gapminder %>%
  mutate(country_clusters =
    as.factor(init_kmeanspp@cluster))
ggplot(aes(x = log_gdp, y = life_expectancy,
           color = country_clusters)) +
  geom_point() +
  ggthemes::scale_color_colorblind() +
  theme_bw() +
  theme(legend.position = "bottom")
```

**Note the use of @ instead of \$...**



So, how do we choose the number of clusters?!



There is no universally accepted way to conclude that a particular choice of  $K$  is optimal!

## Popular heuristic: elbow plot (use with caution)

Look at the total within-cluster variation as a function of the number of clusters

```
# Initialize number of clusters to search over
n_clusters_search <- 2:12
tibble(total_wss =
  # Compute total WSS for each number by looping with sapply
  sapply(n_clusters_search,
    function(k) {
      kmeans_results <- kmeans(dplyr::select(clean_gapminder,
                                             std_log_gdp,
                                             std_life_exp),
                               centers = k, nstart = 30)
      # Return the total WSS for choice of k
      return(kmeans_results$tot.withinss)
    }) %>%
  mutate(k = n_clusters_search) %>%
  ggplot(aes(x = k, y = total_wss)) +
  geom_line() + geom_point() +
  labs(x = "Number of clusters K", y = "Total WSS") +
  theme_bw()
```

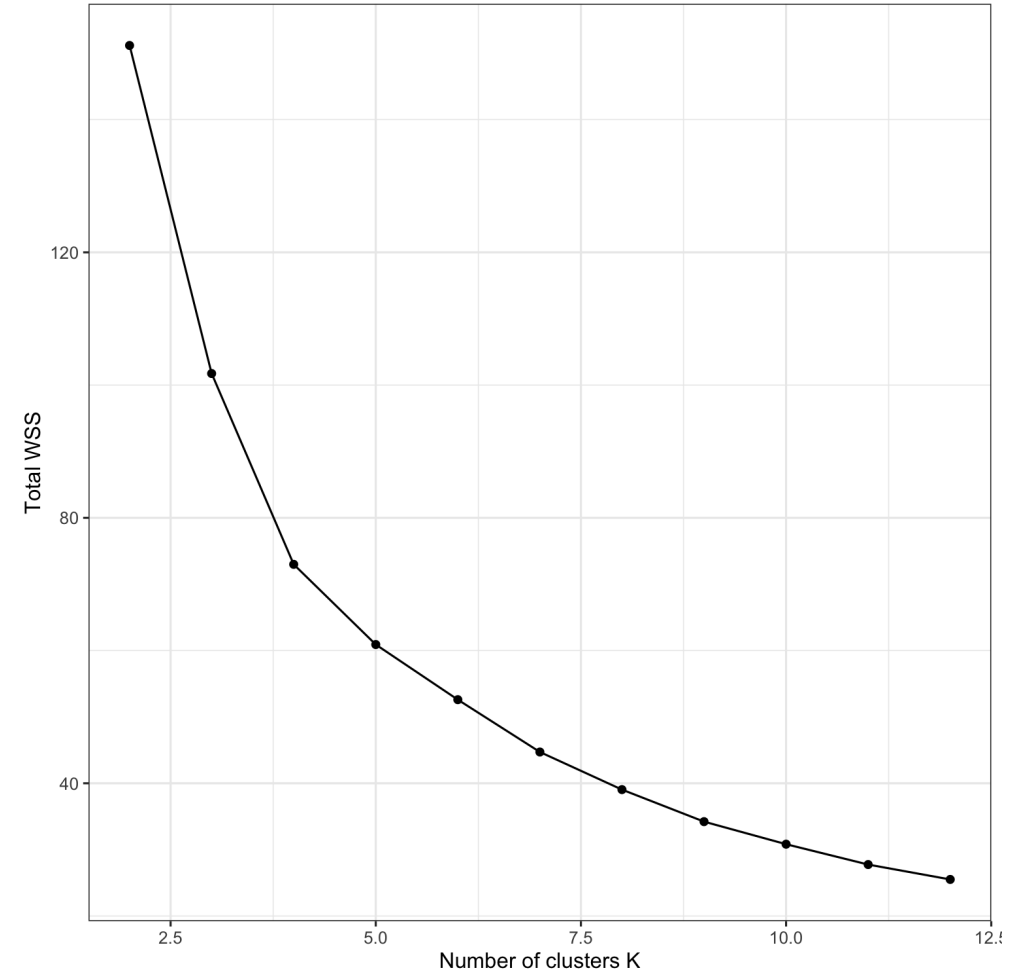
## Popular heuristic: elbow plot (use with caution)

Choose  $K$  where marginal improvements is low at the bend (hence the elbow)

**This is just a guideline and should not dictate your choice of  $K$ !**

Gap statistic is a popular choice (see `clusGap` function in `cluster` package)

**Next Tuesday:** model-based approach to choosing the number of clusters!



## Appendix: elbow plot with flexclust

```
# Initialize number of clusters to search over
n_clusters_search <- 2:12
tibble(total_wss =
  # Compute total WSS for each number by looping with sapply
  sapply(n_clusters_search,
    function(k_choice) {
      kmeans_results <- kcca(dplyr::select(clean_gapminder,
                                           std_log_gdp,
                                           std_life_exp),
                             k = k_choice,
                             control = list(initcent = "kmeanspp"))
      # Return the total WSS for choice of k
      return(sum(kmeans_results@clusinfo$size *
                 kmeans_results@clusinfo$av_dist))
    }) %>%
  mutate(k = n_clusters_search) %>%
  ggplot(aes(x = k, y = total_wss)) +
  geom_line() + geom_point() +
  labs(x = "Number of clusters K", y = "Total WSS") +
  theme_bw()
```

# Appendix: elbow plot with flexclust

