Exploring 2020 across many variables including Covid-19 By: Suren Bhakta and Johnny Henriquez

2022-10-19

Introduction and expectations (Suren)

Within our first report, our group was interested in identifying different trends that would help in depicting where covid-19 was most prominent based on country for the year 2020. After our first report, we were able to answer questions such as, “Is GDP a factor into how many cases were present in each continent and does the development of a country matter?” Now, continuing with our data, we will implememtnt techniques learned further throughout the course to identify much more specific trends. However, the most prominent trend to be expected is that GDP will the best predictor for covid-19 based off performance whereas variables such as units tested will have poor performance.

Sources (Suren)

To answer these respective questions and many more, we will continue with the use of the three data sets from project report 1: For our research, we wanted to include many different metrics that may be indicators for Covid-19. Hence, we found data sets that had different variables from one another. For our Population dataset, we used a dataset from “https://www.worldometers.info/world-population/population-by-country/ (https://www.worldometers.info/world-population/population-by-country/)”. Within this dataset, there were 235 entries with a unique observation displaying the country, population, yearly change, net change, and many other factors relating to population. This dataset holds both numerical and categorical variables. For our world GPD dataset, this was found from “https://ourworldindata.org/grapher/gross-domestic-product (https://ourworldindata.org/grapher/gross-domestic-product)” which has 10,457 entries with a unique observation in the dataset displaying the country, GPD, year of observation, and country code. This dataset has exclusively numerical variables besides country. For our Covid-19 dataset, this was found on https://github.com/owid/covid-19-data/blob/master/public/data/README.md (https://github.com/owid/covid-19-data/blob/master/public/data/README.md) which had 226113 observations. The reason for the extreme amount of observations is due to there being an observation for each country for each day. A unique observation from this data set will include iso code, continent, location, date, and many more variables. This dataset holds both numerical and categorical variables.

Research Questions (Suren)

Now with far more data manipulation techniques at our disposal, we were hoping to see if certain variables were better indicators than others in identifying the spread of the virus in specific countries. One research question that as a group we are interested in solving is, “Is there a grouping that can be done to seperate continents into their own respective groups? Or, will continents be so similar that there is no distinguishing possible?” With this question, we are hoping to find if the countries can be separated into their own respective clusters then observing if relevant trends are shared within respective clusters. Another question we have is, “Do there exist variables

such as GDP or population density that can predict over fifty-percent of a countries population being diagnosed with cases?" With this, we are hoping to use one of our classifying techniques to observe if variables can predict prevalence of the virus.

Tidying (Johnny)

Taking a look at our datasets sourced above, we can see that the data was tidy for our population and gdp datasets. However, the Covid-19 data set was not tidy as we believed there to be variable names for the tests_units variable within a column. For elaboration, tests_units before tidying is a categorical variable which represents how countries were reporting their testing.

```
# "head" used to view top six rows of each dataset to determine if tidying is necessary
head(gross_domestic_product)
```

```
## # A tibble: 6 × 4
##   Entity      Code   Year `GDP (constant 2015 US$)`
##   <chr>       <chr> <dbl>                     <dbl>
## 1 Afghanistan AFG    2002                7228792320
## 2 Afghanistan AFG    2003                7867259392
## 3 Afghanistan AFG    2004                7978511360
## 4 Afghanistan AFG    2005                8874475520
## 5 Afghanistan AFG    2006                9349916672
## 6 Afghanistan AFG    2007               10642666496
```

```
head(pop)
```

```
## # A tibble: 6 × 12
##      no Countr…¹ Popul…² Yearl…³ Net C…⁴ Densi…⁵ Land …⁶ Migra…⁷ Fert.…⁸ Med. …⁹
##   <dbl> <chr>      <dbl> <chr>     <dbl>   <dbl>   <dbl>   <dbl> <chr>   <chr>
## 1     1 China     1.44e9 0.39%    5.54e6     153 9388211 -348399 1.7     38
## 2     2 India     1.38e9 0.99%    1.36e7     464 2973190 -532687 2.2     28
## 3     3 United …  3.31e8 0.59%    1.94e6      36 9147420  954806 1.8     38
## 4     4 Indones…  2.74e8 1.07%    2.90e6     151 1811570  -98955 2.3     30
## 5     5 Pakistan  2.21e8 2.00%    4.33e6     287  770880 -233379 3.6     23
## 6     6 Brazil    2.13e8 0.72%    1.51e6      25 8358140   21200 1.7     33
## # … with 2 more variables: `Urban Pop %` <chr>, `World Share` <chr>, and
## #   abbreviated variable names ¹`Country (or dependency)`, ²`Population 2020`,
## #   ³`Yearly Change`, ⁴`Net Change`, ⁵`Density  (P/Km²)`, ⁶`Land Area (Km²)`,
## #   ⁷`Migrants (net)`, ⁸`Fert. Rate`, ⁹`Med. Age`
```

```
head(owid_covid_data)
```

```
## # A tibble: 6 × 67
##   iso_code continent location date       total…¹ new_c…² new_c…³ total…⁴ new_d…⁵
##   <chr>    <chr>     <chr>    <date>       <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 AFG      Asia      Afghani… 2020-02-24       5       5  NA          NA      NA
## 2 AFG      Asia      Afghani… 2020-02-25       5       0  NA          NA      NA
## 3 AFG      Asia      Afghani… 2020-02-26       5       0  NA          NA      NA
## 4 AFG      Asia      Afghani… 2020-02-27       5       0  NA          NA      NA
## 5 AFG      Asia      Afghani… 2020-02-28       5       0  NA          NA      NA
## 6 AFG      Asia      Afghani… 2020-02-29       5       0   0.714      NA      NA
## # … with 58 more variables: new_deaths_smoothed <dbl>,
## #   total_cases_per_million <dbl>, new_cases_per_million <dbl>,
## #   new_cases_smoothed_per_million <dbl>, total_deaths_per_million <dbl>,
## #   new_deaths_per_million <dbl>, new_deaths_smoothed_per_million <dbl>,
## #   reproduction_rate <dbl>, icu_patients <dbl>,
## #   icu_patients_per_million <dbl>, hosp_patients <dbl>,
## #   hosp_patients_per_million <dbl>, weekly_icu_admissions <dbl>, …
```

```
# Since the owid_covid_data dataset was not tidy, will use pivot_wider to expand the tes
ts_units column into its own respective columns with values coming from the variable tes
ts_per_case
tidy_owid<-owid_covid_data%>%
  pivot_wider(names_from = tests_units,
              values_from = tests_per_case)
```

Joining/Merging (Johnny)

We planned to join our various datasets based off country. However, referring to the tidying section above, it is clear to see that not all datasets are named specifically by country. Thus to address this issue, we will be using an assortment of dplyr functions to allow us to join the three datasets together.

Another issue is that our datasets as said above may have an extreme amount of observations due to observations being made each day for each country. Thus before joining, we will filter the data for the data just in the year 2020.

```
# Rename so country is a named variable as well as filter for only values from 2020
GDP<-gross_domestic_product%>%
  rename(GDP= "GDP (constant 2015 US$)")%>%
  rename(Country = "Entity")%>%
  filter(Year=="2020")
nrow(GDP) # Count observations
```

```
## [1] 210
```

```
# 2020 was already the only year for observations so only had to rename such that countr
y is a unique variable name
Population <- pop %>%
  rename(Country = "Country (or dependency)")
nrow(Population) # Count observations
```

```
## [1] 235
```

```
# Once again rename so that country is a unique variable name
# This dataset was a little more tricky to work for only 2020 values since there was an
 observation per country per day.
# So, we filtered for only data found at the end of 2020
Covid<-tidy_owid%>%
  rename(Country="location")%>%
  filter(date=="2020-12-31") # We filter by this data so that we get the total amount of
cases ammased by the end of the year 2020
nrow(Covid) # Count observations
```

```
## [1] 228
```

Now that we have addressed the issue extreme amounts of observations, we now are able to join our datasets. Luckily, the rename function within dplyr allowed us to rename all the columns within the datasets that held the country name with simply the word “Country.” This will now be our key variable as we prepare to join our dataset.

Also before joining, observe how the GDP dataset had 210 observations which had a unique variable of GDP, the Population set having 235 observations which had a unique variable of “Net Change” (measure of pop. change from prev. year to next), and the Covid dataset having 228 observations with a unique variable being “total_cases”. As said above, all countries now share the ID Country. As for ID’s that may be left out, there are countries in the population dataset such as the “Carribean Netherlands” which is not in the other datasets.

```
# Now we can join our data
# First join GDP dataset to population dataset with key variable Country
join1 <- Population %>%
  inner_join(GDP,by="Country")

# Now with our two previously joined datasets, join the Covid dataset with key variable
 country
join2 <- join1 %>%
  inner_join(Covid,by="Country")
nrow(join2)
```

```
## [1] 176
```

As seen with our code above, we joined our three completely different datasets by the key variable “Country.” We opted to use the inner join function to join our data to give us more control of our data. We felt that by using inner join, we would be able to manually select which variables we want to manipulate and which variables we want to remove instead of r joining only by key variable.

After joining our data with inner join, we were left with 176 observations, meaning that 59 rows were dropped. Although at face value this does not seem like a problem, this may be an issue later on. Our suspicion is that some of the smaller countries or countries with two word names were dropped from the dataset. This will cause skewing of data that is largely unavoidable.

This will be the data set that we will use for the remainder of our project

```
# Saving to a new data set
# Using dplyr function select which allows us to select columns by index number rather t
han by name which is far more efficient
new_data <- join2 %>%
  select(2,3,6,7,11,15,17,19,22,40,48,49,63,64,76,74,10,73)
```

Exploratory Data Analysis

Now, we are interested in how the respective variables correlate with each other. However before doing so, we are only capable on running the correlation function on numeric variables, hence we must remove na values and non-numeric variables which will allow us to create our correllation matrix.

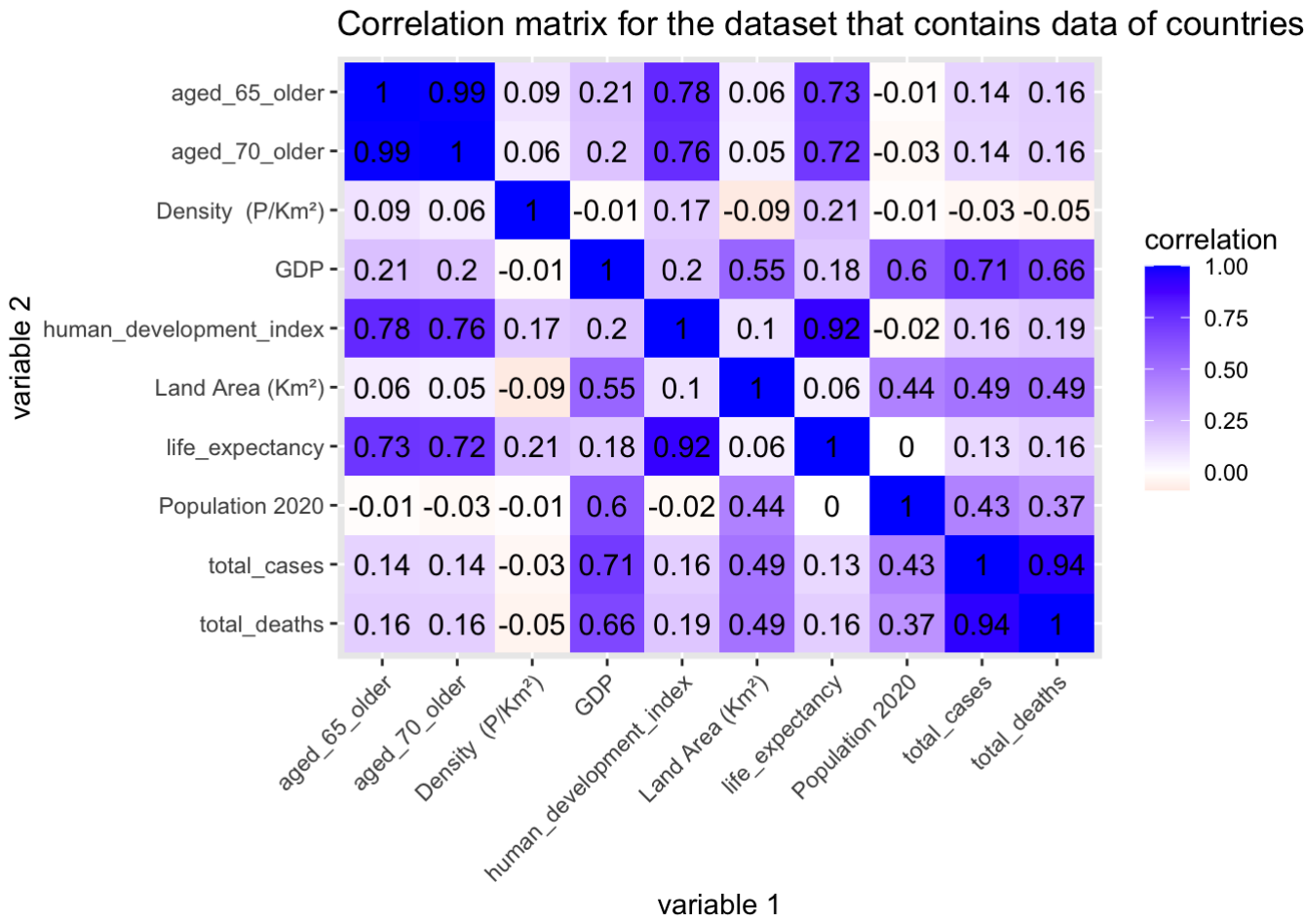
```
cordat<-new_data%>%
  select_if(is.numeric)%>% # if statement that keeps only numeric variables
  select(1,2,3,4,5,6,10,11,13,14)%>% # keep variables that are interested in
  drop_na() # remove na values
new_data
```

```
## # A tibble: 176 × 18
##    Country       Popul…¹ Densi…² Land …³ Urban…⁴     GDP conti…⁵ total…⁶ total…⁷
##    <chr>           <dbl>   <dbl>   <dbl> <chr>     <dbl> <chr>     <dbl>   <dbl>
##  1 China          1.44e9     153  9.39e6 61%     1.46e13 Asia     9.37e4    4634
##  2 India          1.38e9     464  2.97e6 35%     2.50e12 Asia     1.03e7  148994
##  3 United States  3.31e8      36  9.15e6 83%     1.93e13 North …  2.02e7  350544
##  4 Indonesia      2.74e8     151  1.81e6 56%     1.03e12 Asia     7.43e5   22138
##  5 Pakistan       2.21e8     287  7.71e5 35%     3.20e11 Asia     4.82e5   10176
##  6 Brazil         2.13e8      25  8.36e6 88%     1.75e12 South …  7.68e6  195072
##  7 Nigeria        2.06e8     226  9.11e5 52%     4.94e11 Africa   8.76e4    1289
##  8 Bangladesh     1.65e8    1265  1.30e5 39%     2.71e11 Asia     5.14e5    7559
##  9 Russia         1.46e8       9  1.64e7 74%     1.42e12 Europe   3.13e6   56271
## 10 Mexico         1.29e8      66  1.94e6 84%     1.15e12 North …  1.43e6  125807
## # … with 166 more rows, 9 more variables: total_tests <dbl>,
## #   people_vaccinated <dbl>, people_fully_vaccinated <dbl>,
## #   aged_65_older <dbl>, aged_70_older <dbl>,
## #   excess_mortality_cumulative_absolute <dbl>, human_development_index <dbl>,
## #   `Med. Age` <chr>, life_expectancy <dbl>, and abbreviated variable names
## #   ¹`Population 2020`, ²`Density  (P/Km²)`, ³`Land Area (Km²)`,
## #   ⁴`Urban Pop %`, ⁵continent, ⁶total_cases, ⁷total_deaths
```

```
cordat
```

```
## # A tibble: 157 × 10
##    Population …¹ Densi…² Land …³     GDP total…⁴ total…⁵ aged_…⁶ aged_…⁷ human…⁸
##            <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1    1439323776     153  9.39e6 1.46e13  9.37e4    4634   10.6     5.93   0.761
##  2    1380004385     464  2.97e6 2.50e12  1.03e7  148994    5.99    3.41   0.645
##  3     331002651      36  9.15e6 1.93e13  2.02e7  350544   15.4     9.73   0.926
##  4     273523615     151  1.81e6 1.03e12  7.43e5   22138    5.32    3.05   0.718
##  5     220892340     287  7.71e5 3.20e11  4.82e5   10176    4.50    2.78   0.557
##  6     212559417      25  8.36e6 1.75e12  7.68e6  195072    8.55    5.06   0.765
##  7     206139589     226  9.11e5 4.94e11  8.76e4    1289    2.75    1.45   0.539
##  8     164689383    1265  1.30e5 2.71e11  5.14e5    7559    5.10    3.26   0.632
##  9     145934462       9  1.64e7 1.42e12  3.13e6   56271   14.2     9.39   0.824
## 10     128932753      66  1.94e6 1.15e12  1.43e6  125807    6.86    4.32   0.779
## # … with 147 more rows, 1 more variable: life_expectancy <dbl>, and abbreviated
## #   variable names ¹`Population 2020`, ²`Density  (P/Km²)`, ³`Land Area (Km²)`,
## #   ⁴total_cases, ⁵total_deaths, ⁶aged_65_older, ⁷aged_70_older,
## #   ⁸human_development_index
```

```
# Find the correlations among the 10 disciplines
cor(cordat, use = "pairwise.complete.obs") %>%
  # Save as a data frame
  as.data.frame %>%
  # Convert row names to an explicit variable
  rownames_to_column %>%
  # Pivot so that all correlations appear in the same column
  pivot_longer(-1,
               names_to = "other_var",
               values_to = "correlation") %>%

  # Define ggplot (reorder values on y-axis)
  ggplot(aes(x = rowname,
             y = ordered(other_var, levels = rev(sort(unique(other_var)))),
             fill = correlation)) +
  # Heat map with geom_tile
  geom_tile() +
  # Change the scale to make the middle appear neutral
  scale_fill_gradient2(low = "red", mid = "white", high = "blue") +
  # Overlay values
  geom_text(aes(label = round(correlation,2)), color = "black", size = 4) +
  # Angle the x-axis label to 45 degrees
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  # Give title and labels
  labs(title = "Correlation matrix for the dataset that contains data of countries all o
ver the world",
       x = "variable 1", y = "variable 2")
```



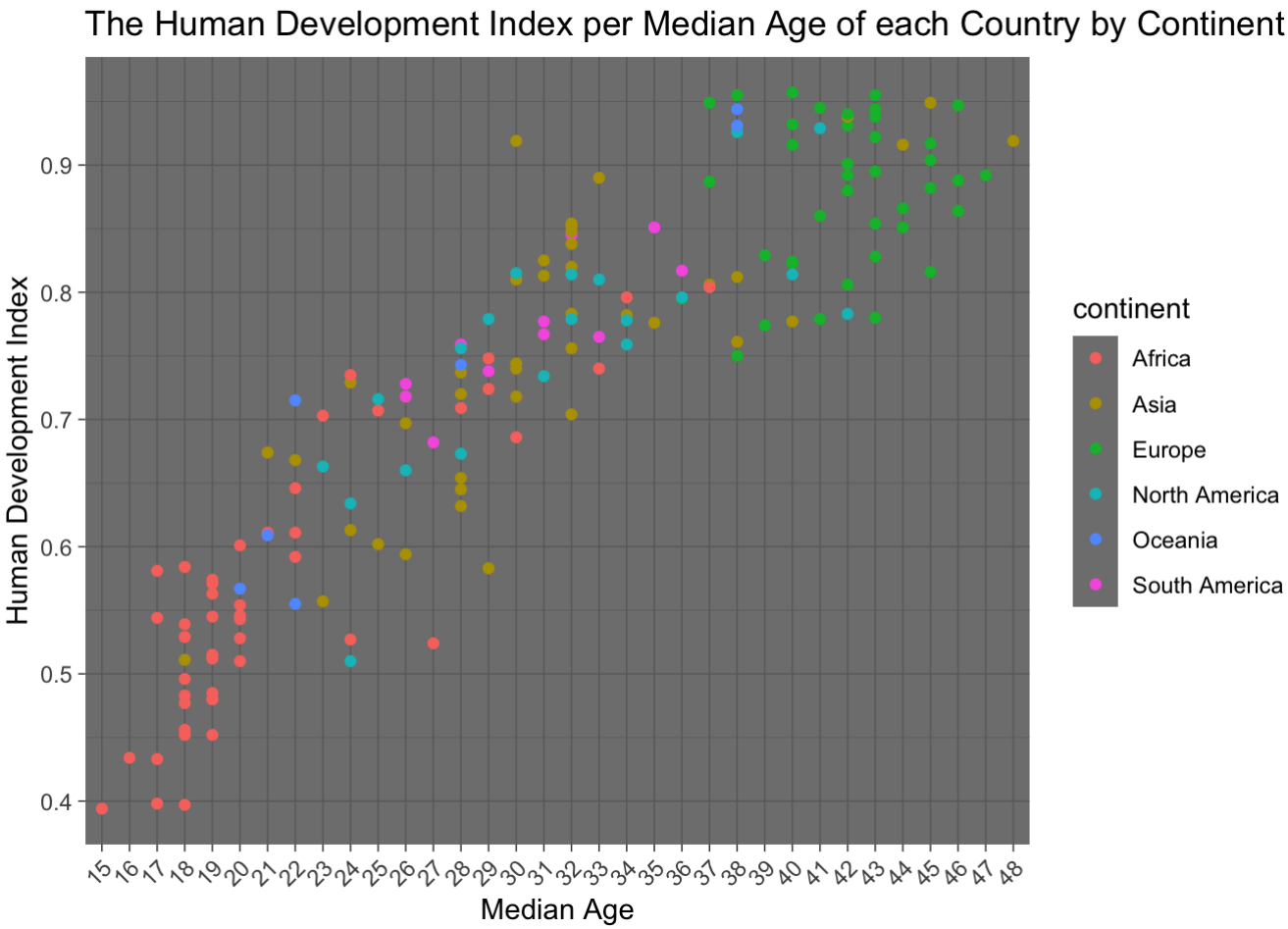
Based off our created correlation matrix, we are able to identify trends that before may have not been expected or found. The variables that have the most positive correlation are signified by the deep, dark blue coloring where the variables meet. Hence, the most positively correlated variables are the total deaths and total cases with a correlation factor of 0.94. This indicates that as the number of cases increase, as do the number of deaths as well and vice-versa. This means in the context of our report that as a whole, the countries around the world struggled to care for patients that were diagnosed with the virus. We can also see that there were not any variables in our data set that were highly negatively correlated. However, we had many variable interactions that were either close to zero or a perfect zero. This indicates that there is no relation between the data at all. An example of this interaction is life expectancy and total population which obviously would not have any correlation since some of the population values are skewed so highly, there would be no chance of finding a correlation. One trend that was found from the matrix is that variables that are related to each other in context such as aged_65_older and aged_75_older, have very strong correlations.

Now with our correlation matrix created, we can now begin to make assumptions on how our variables interact with each other. To see these various interactions, we will create four seperate visualizations.

Visualization 1 (Suren)

For our visualization, we wanted to investigate the influence median age had on the variables human development index and by continent. Human development index is the measure of life expectancy, education, and per capita income in a single metric. With this, the hope is to visualize some sort of connection between the three variables which could be an indicator of covid 19 presence within continents.

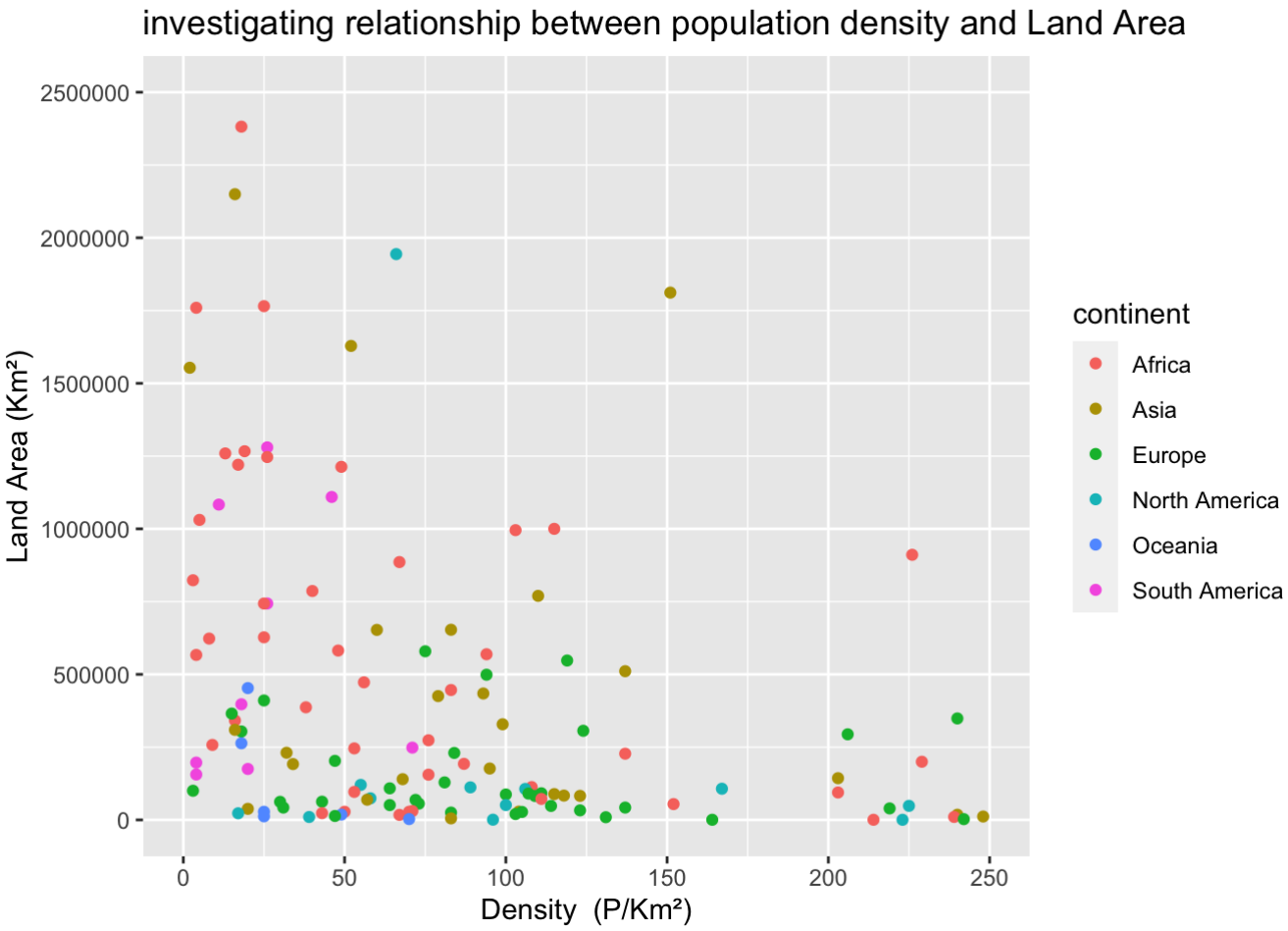
```
new_data%>%
  filter(`Med. Age` != "N.A.")%>% # taking out "N.A." values from `Med. Age`
  ggplot(aes(x=as.factor(`Med. Age`),y=human_development_index,color=continent))+
  geom_point() +
  theme_dark()+ # changes theme
  theme(axis.text.x = element_text(angle = 45, hjust = 1))+ #rotates x scale numbers
  scale_y_continuous(breaks = seq(0, 1, .1))+ # rescales the y axis
  labs(x = "Median Age", y = "Human Development Index",
       title = "The Human Development Index per Median Age of each Country by Continent"
) #labels
```



After creating a trivariable representation which maps human development index by median age per country by continent, we interestingly found a positive relationship between median age and the human development index. In contextual terms, this means that when median age increases, the development index increases as well. THis makes sense because if a country were to have a higher median age, that means that the country has a much more sustainable health care system which keeps the elderly alive and well. Since this was a trivariable graph, we were able to see where continents ranked among other continents based off the two variables. Based off the graph, one can see that the majority of countries within the African continent had th elowest median ages of all. Although more subtle, at the height of the median age range, that is where the majority of European countries are found.

Visualization 2 (Suren) For our second visualization, our group was intrigued of how populated continents were. However, we wanted to see if creating a scatter plot mapping Density of the respective country to the land area would give any insight. Our prediction is that there will not be a very strong correlation in either direction at all.

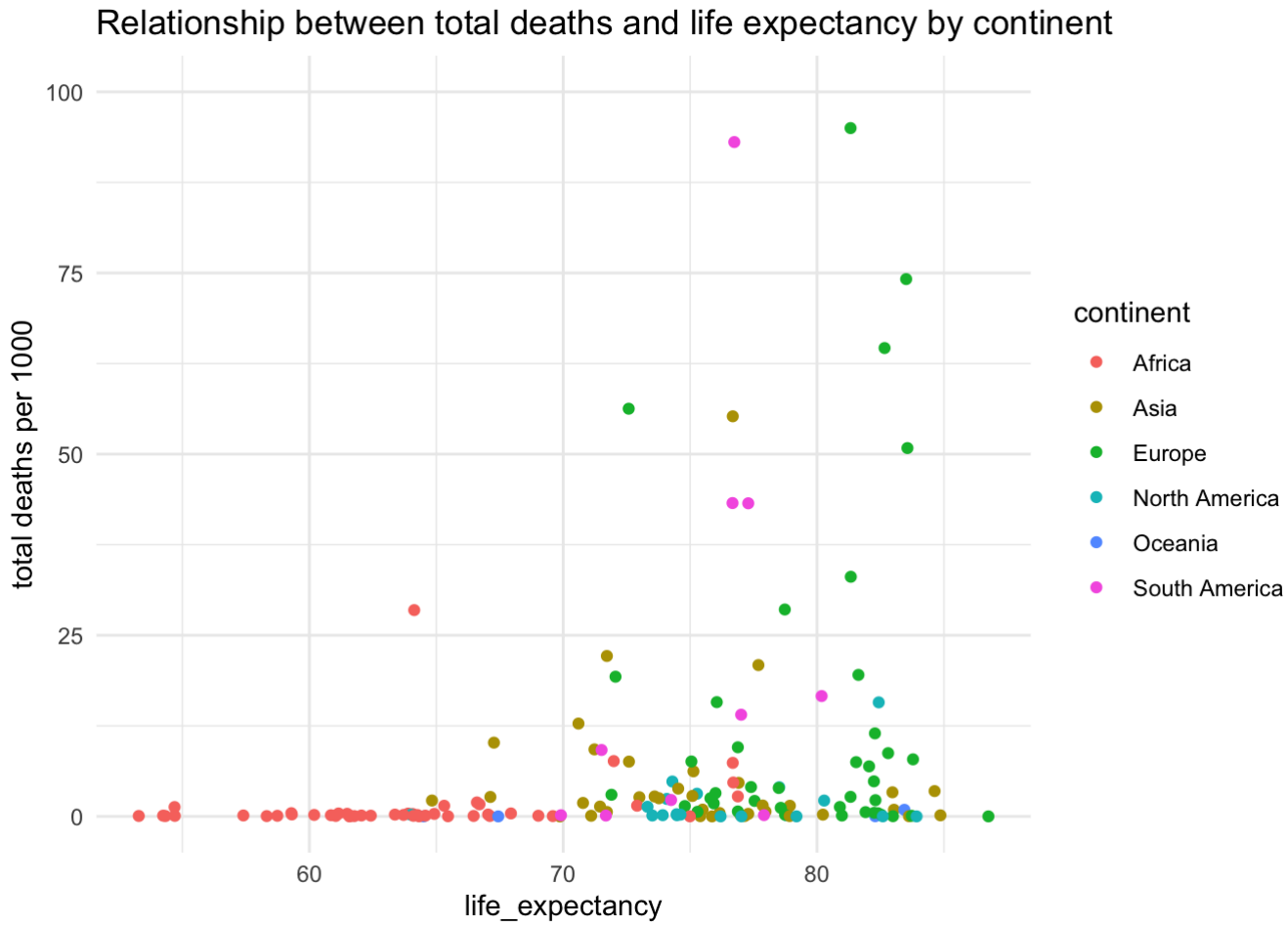
```
new_data %>%
  filter(!is.na(`Density  (P/Km²)`), !is.na(`Land Area (Km²)`)) %>% # remove NA values
  ggplot(aes(x=`Density  (P/Km²)`, y = `Land Area (Km²)` , color = continent)) + # creat
e plot
  geom_point() +
  ylim(0, 10000000/4) + # adjust limits for better viewing of graph
  xlim(0, 250) +
  labs(title = "investigating relationship between population density and Land Area")
```



As expected, after creating a scatterplot mapping the Density of a country to the Land Area of the respective country, there was not found to be any correllation between the two. However, if you look closely, you can see that almost all the South American countris were grouped towards the origin which indicates that the countries were not very large, but they were also not densely populated either. Also, most European countries with a few exceptions were huggling along the x-axis. This means that these areas are densely populated with not much room, which could be an indicator of many cases being present due to close proximity to other individuals.

Visualization 3 (Johnny) After viewing the correlation matrix, we were surprised to find that Life Expectancy and total deaths had little to no correlation at all since a higher life expectancy would mean that the amount of deaths in one continent would be much larger correct? We will see.

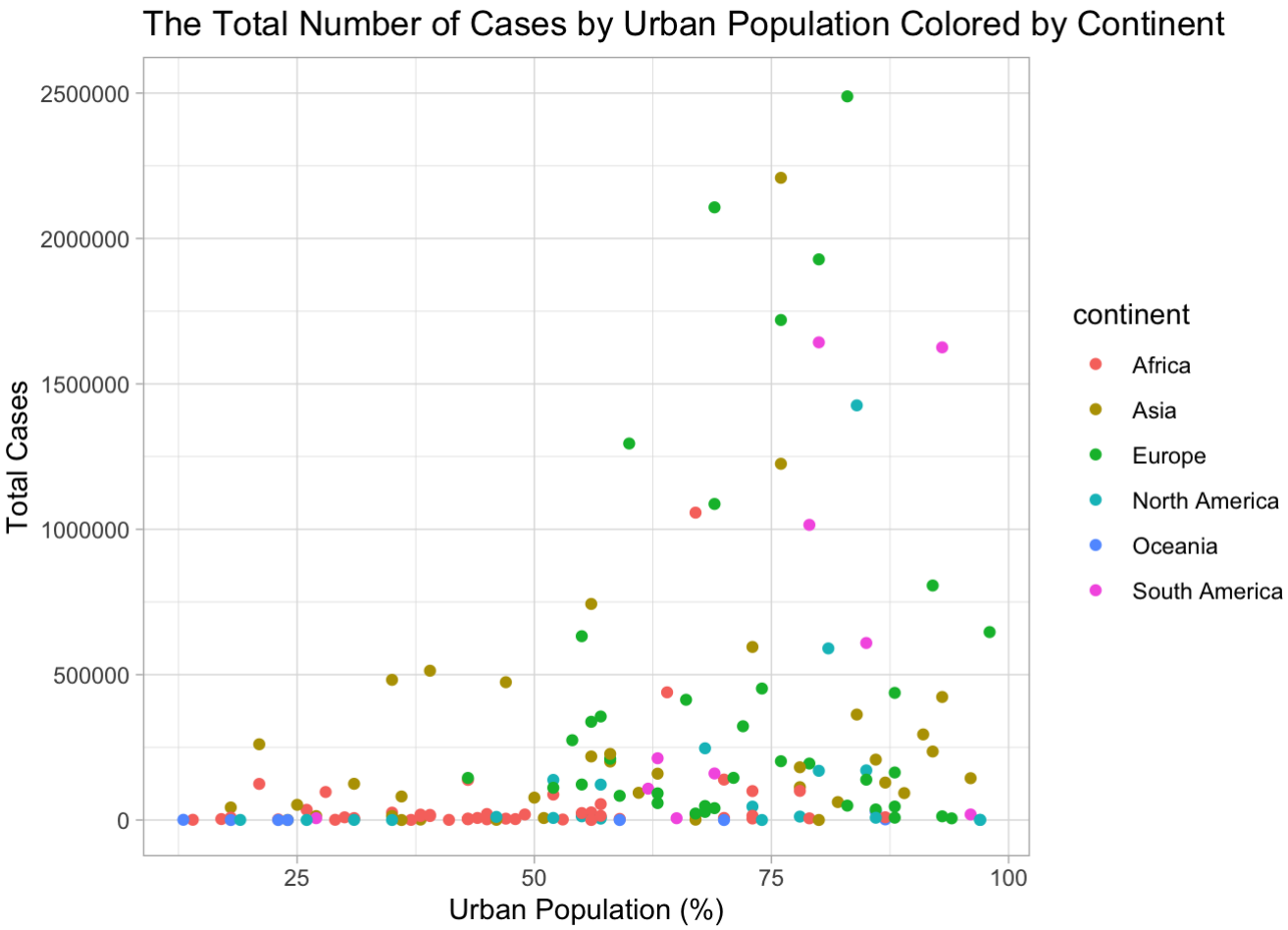
```
new_data %>%
  ggplot(aes(x= life_expectancy, y = total_deaths/1000, color = continent)) + # create g
raph
  geom_point()+
  ylim(0,100)+
  theme_minimal()+ # change theme of graph
  labs("Life expectancy" , y = "total deaths per 1000", title = "Relationship between to
tal deaths and life expectancy by continent")  # title
```



After adjusting the limits on our graph to obtain a more accurate visualization, regardless of continent or country, it can be seen that there is a very mild positive correllation between life expectancy and total deaths which as said earlier is very surprising since if there was a higher life expectancy, there should be less deaths. However, upon further analysis we obtained why there is a slight positive correlation. This is due to since countries already had a large life expectancy, then once the elderly were infected, they were not able to recover, hence why larger countries had far more deaths.

Visualization 4 (Johnny) For our final plot, we wanted to visualize the trend between urban population and total number of cases. Our prediction is that as the urban percentage increases, as will the total number of cases since that would imply that the population is much more compact into sertain areas. However, sice urban population is a categorical variable, we must do some manipulation to be able to plot the graph accurately.

```
new_data %>%
  mutate(urban_pop = as.numeric(str_remove_all(new_data$`Urban Pop %`, "[%]"))) %>% # Re
moves the percent sign from all observations
  select(continent, total_cases, urban_pop) %>%
  filter(!is.na(total_cases), !is.na(urban_pop)) %>% # remove NA values
  ggplot(aes(x = urban_pop, y = total_cases, color = continent)) +
  geom_point() +
  ylim(0, 2500000) +
  theme_light() +
  labs(x = "Urban Population (%)", y = "Total Cases",
       title = "The Total Number of Cases by Urban Population Colored by Continent")
```



After creating our final trivariable representation mappingtotal cases for each percentage of urban population, interestngly enough, there was little to no correlation between the two variables. As Urban population did exceed 60%, the number of cases did increase as well. However, there was no constant correlation implying that the two variables were related at all. However, this subtle increase as the urban population increased indicates that the number of cases does peak further more in cities compared to more rural areas.

Clustering

Now with clustering, we want to keep the continent variable along with all our numeric variables, hence we will create a new data set for continuity.

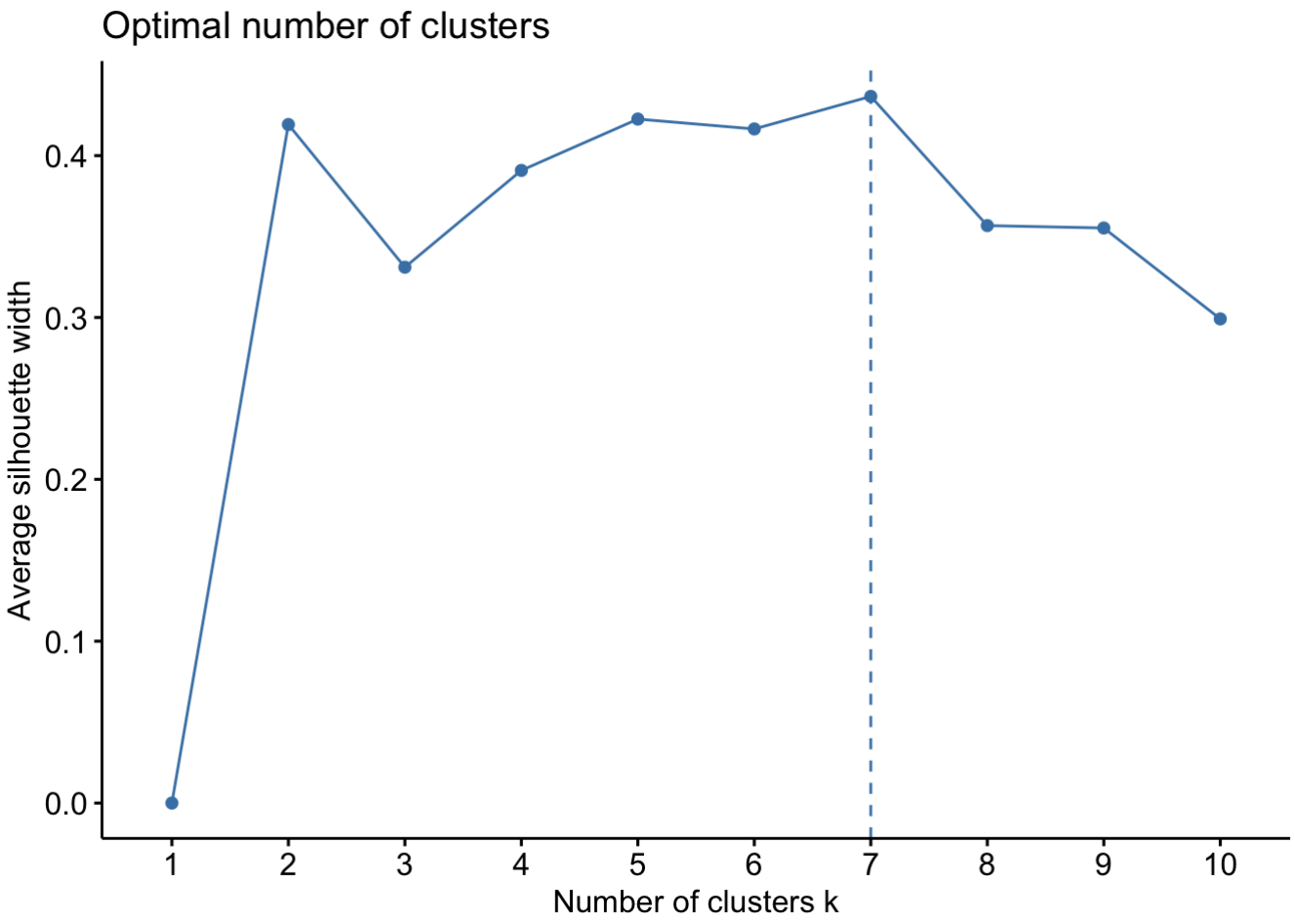
```
cluster_data<-new_data %>%
  select(2,3,4,6,7,8,9,13,14,16,18) %>% # select function to keep variables by column nu
mber
  drop_na() # drop na values
```

First off, before we can apply a clustering algorithm, we must scale our data first

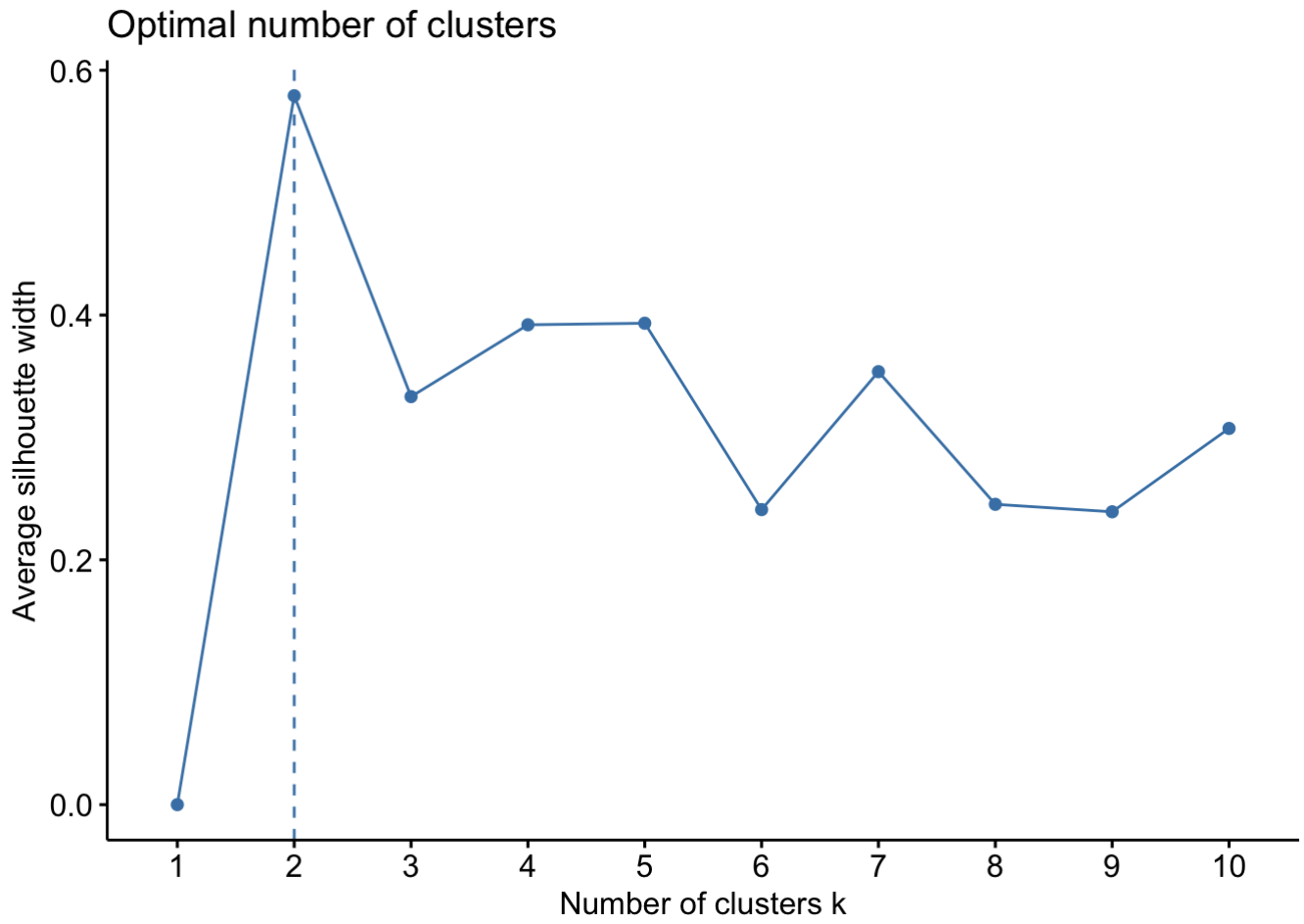
```
cluster_scale<- cluster_data%>%
  select(-(continent))%>% # remove the categorical variable
  scale # scale
```

Now with our scaled data, it is time to perform a clustering algorithm of our choosing. However, we first need to determine how many clusters need to be made. To do this we will use the number of cluster visualization with silhoulette width. However, we will observe the widths of both pam and kmeans to see which number will make sense.

```
# Maximize the silhouette while keeping a small number of clusters
fviz_nbclust(cluster_scale, pam, method = "silhouette")
```




```
fviz_nbclust(cluster_scale, kmeans, method = "silhouette")
```

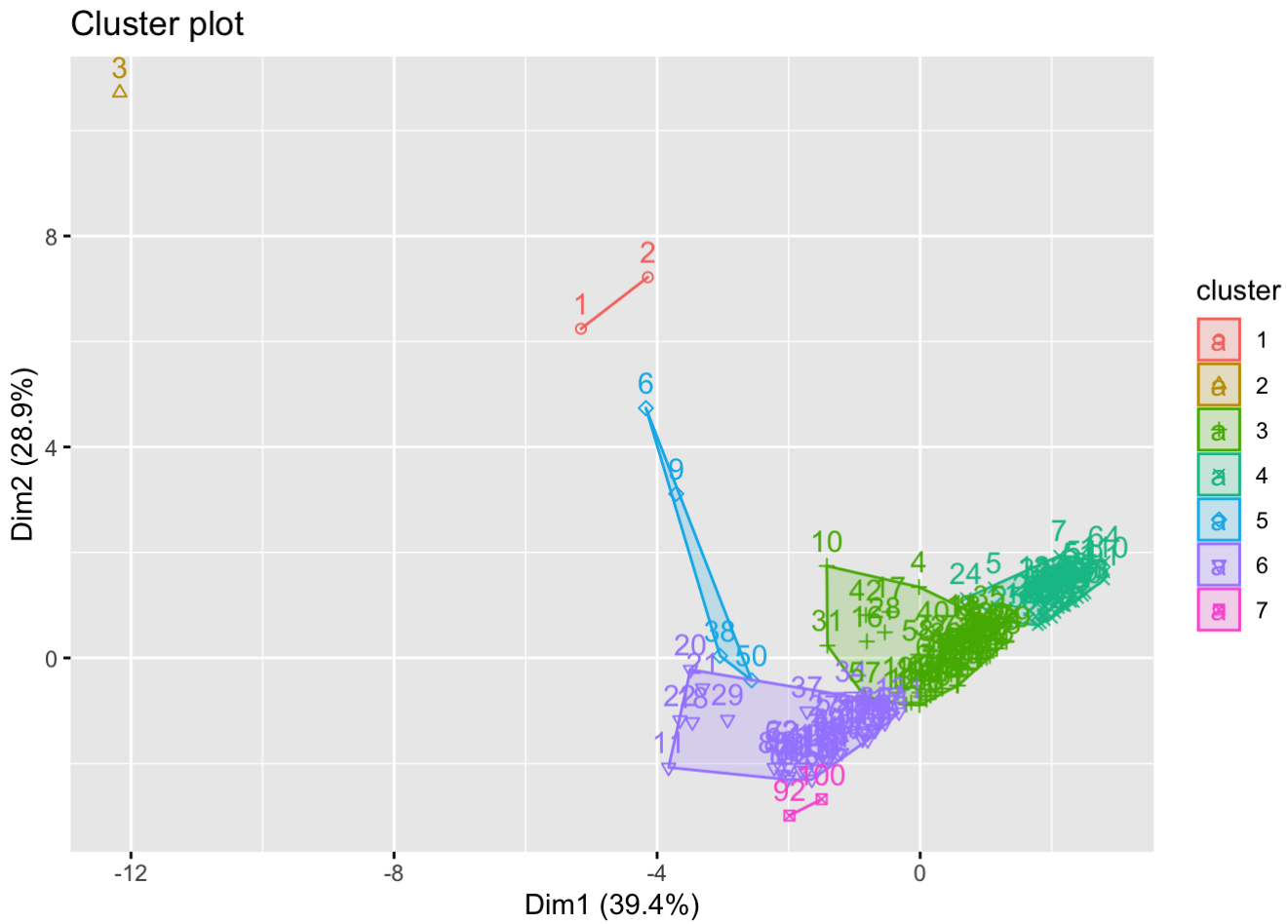


It is shown that it is optimal with kmeans to use 2 clusters. However, with pam it is most optimal to use 7 clusters. With there being 7 separate continents in our dataset, we are interested if clusters would be assigned accordingly. Hence, we will perform clustering on our data with a center of 7 via the pam algorithm with the hopes that the seven different clusters will be assigned to the seven different continents.

```
pam_cluster <- cluster_scale %>%
  pam(k = 7) # number of clusters
```

```
# Save cluster assignment as a column in the original dataset
cluster_cont <- cluster_data %>%
  mutate(cluster = as.numeric(pam_cluster$cluster))
```

```
fviz_cluster(pam_cluster, data = cluster_scale) # visualize our clusters
```



With the 7 centers, there seems to be much more overlap within the widths of assignments, but why? We believe this to be how similar much of the countries are within their respective continents. However, the very minimal countries that were towards the left side of the plot could be the most developed countries on Earth such as India, China, and the United States. In general, it was likely best to proceed with 2 centers instead of 7.

Now we will be able to perform statistics on our clusters to see if there is anyhthing distinguishable that we may have missed since the graph is so congested.

First we will observe the strength of our algorithm by observing our silhouette width.

```
# Average silhouette width
pam_cluster$silinfo$avg.width
```

```
## [1] 0.4366092
```

With our obtained silhouette width of 0.04366092, we can confidently say that our algorithm did not do a great job at all of distinguishing the continents like we had hoped. However, this does provide an insight into the fact that since the clusters were so close and unique to each other, then much of our observations are more similar than we thought!

Now, we are interested in observing if there is any minimal difference between the clusters in terms of our original variables. Hence, we will find the mean values of our variables by cluster.

```
cluster_cont %>%
  group_by(cluster)%>%
  summarize_if(is.numeric, mean, na.rm = T)
```

```
## # A tibble: 7 × 11
##   cluster Population 2…¹ Densi…² Land …³     GDP total…⁴ total…⁵ aged_…⁶ aged_…⁷
##     <dbl>          <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1       1    1409664080.   308.   6.18e6 8.57e12  5.19e6  7.68e4    8.32    4.67
## 2       2     331002651     36    9.15e6 1.93e13  2.02e7  3.51e5   15.4     9.73
## 3       3      28883113.   207.   5.13e5 1.65e11  2.84e5  9.01e3    6.35    3.89
## 4       4      30658722.   115.   4.99e5 4.87e10  5.64e4  1.29e3    3.21    1.86
## 5       5     105433979.    10.2  1.04e7 1.57e12  2.86e6  6.70e4   13.8     8.84
## 6       6      17936320.   175.   1.51e5 5.60e11  4.64e5  1.14e4   17.6    11.7
## 7       7       6673662.  7749    8.75e2 3.21e11  3.37e4  8.85e1   14.6     8.60
## # … with 2 more variables: human_development_index <dbl>,
## #   life_expectancy <dbl>, and abbreviated variable names ¹`Population 2020`,
## #   ²`Density  (P/Km²)`, ³`Land Area (Km²)`, ⁴total_cases, ⁵total_deaths,
## #   ⁶aged_65_older, ⁷aged_70_older
```

As we can see, there are very different means for all seven clusters. Then why did we have an overlap? THis could be the result of extremely large observations for variables such as population which clumped the data together.

Also what can be found is the center of each cluster in terms of our VERY ORIGINAL dataset with hopes of being able to correctly identify the clusters by names rather by numbers.

```
# Look at the final medoids
new_data[pam_cluster$id.med,]
```

```
## # A tibble: 7 × 18
##   Country       Popula…¹ Densi…² Land …³ Urban…⁴     GDP conti…⁵ total…⁶ total…⁷
##   <chr>            <dbl>   <dbl>   <dbl> <chr>     <dbl> <chr>     <dbl>   <dbl>
## 1 China          1.44e9     153 9388211 61%     1.46e13 Asia     9.37e4    4634
## 2 United States  3.31e8      36 9147420 83%     1.93e13 North …  2.02e7  350544
## 3 Togo           8.28e6     152   54390 43%     5.19e 9 Africa   3.63e3      68
## 4 Guinea         1.31e7      53  245720 39%     1.29e10 Africa   1.37e4      81
## 5 Canada         3.77e7       4 9093510 81%     1.61e12 North …  5.90e5   15736
## 6 Honduras       9.90e6      89  111890 57%     2.20e10 North …  1.22e5    3130
## 7 Lebanon        6.83e6     667   10230 78%     3.67e10 Asia     1.82e5    1468
## # … with 9 more variables: total_tests <dbl>, people_vaccinated <dbl>,
## #   people_fully_vaccinated <dbl>, aged_65_older <dbl>, aged_70_older <dbl>,
## #   excess_mortality_cumulative_absolute <dbl>, human_development_index <dbl>,
## #   `Med. Age` <chr>, life_expectancy <dbl>, and abbreviated variable names
## #   ¹`Population 2020`, ²`Density  (P/Km²)`, ³`Land Area (Km²)`,
## #   ⁴`Urban Pop %`, ⁵continent, ⁶total_cases, ⁷total_deaths
```

As stated earlier, it was expected that China and United States would be representatives of their own clusters, but what was not expected is that there are no European countries as centers of any cluster, indicating that many of the countries are more similar than what we imagined! This means that regardless of where a country is on the Earth, there will be a high chance of a country being similar to multiple other countries.

Dimensionality reduction

For dimensionality reduction, we will perform PCA on a sub-dataset that holds only the numeric variables from our original joined data set since pca is only capable of being ran on numeric variables.

```
pca_data <- new_data%>%
  select(2,3,4,6,7,8,9,13,14,16,18)%>%
  mutate_if(is.character, as.factor) %>% # converts charcters to factors
  drop_na()
```

Now we will actually perform PCA on our dataset.

```
pca_scale <- pca_data %>%
  select_if(is.numeric)%>%
  # Scale the variables
  scale %>%
  # Save as a data frame
  as.data.frame
```

Now performing PCA

```
pca_done <- pca_scale %>%
  prcomp # pca on the sclaed values
```

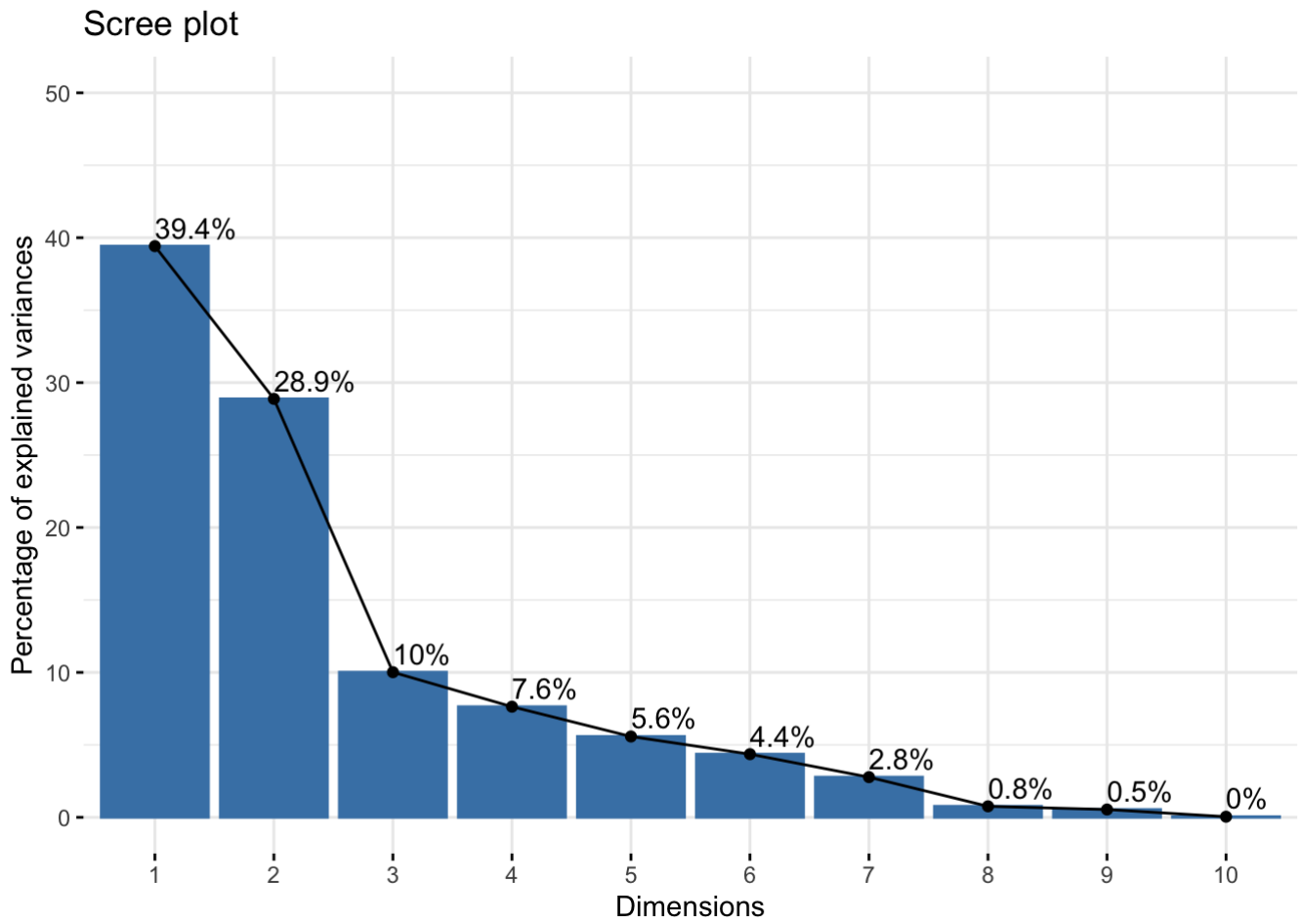
Now to determine the number of PC's that should be used, we will use the scree plot vizualization fvis_eig along with the variation table to see if our interpretation matches the scree plot.

```
# Look at percentage of variance explained for each PC in a table
get_eigenvalue(pca_done)
```

```
##         eigenvalue variance.percent cumulative.variance.percent
## Dim.1  3.941980324      39.41980324                     39.41980
## Dim.2  2.888060076      28.88060076                     68.30040
## Dim.3  1.001492597      10.01492597                     78.31533
## Dim.4  0.763374525       7.63374525                     85.94908
## Dim.5  0.557894564       5.57894564                     91.52802
## Dim.6  0.435734020       4.35734020                     95.88536
## Dim.7  0.277297900       2.77297900                     98.65834
## Dim.8  0.075902223       0.75902223                     99.41736
## Dim.9  0.053944670       0.53944670                     99.95681
## Dim.10 0.004319103       0.04319103                    100.00000
```

Based off the table, we generally would want to keep the number of pc's that will sum to be less than or equal to 80 in our cumulative variance percentage. Hence we will keep the first three dimensions.

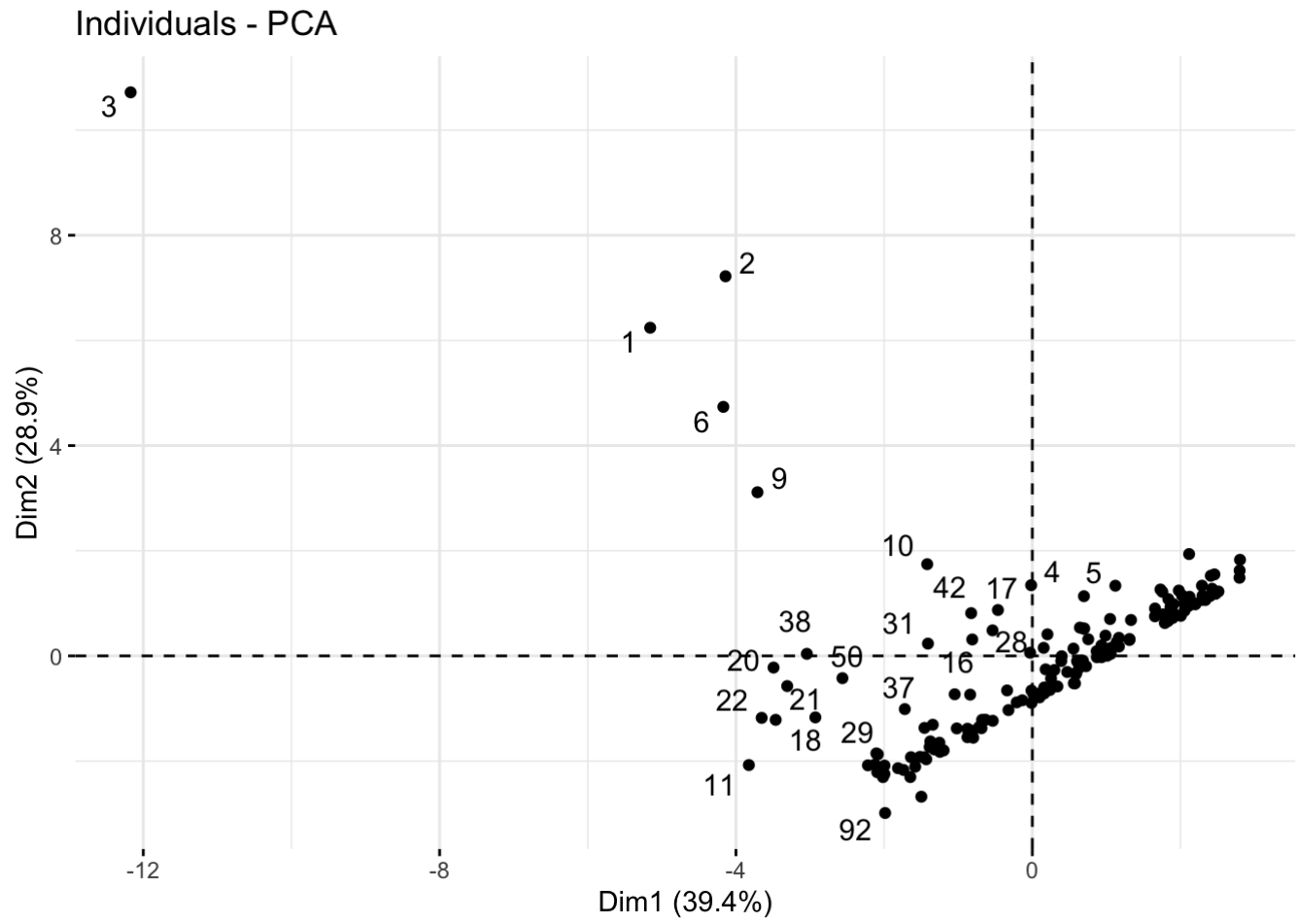
```
# Visualize percentage of variance explained for each PC in a scree plot
fviz_eig(pca_done, addlabels = TRUE, ylim = c(0, 50))
```



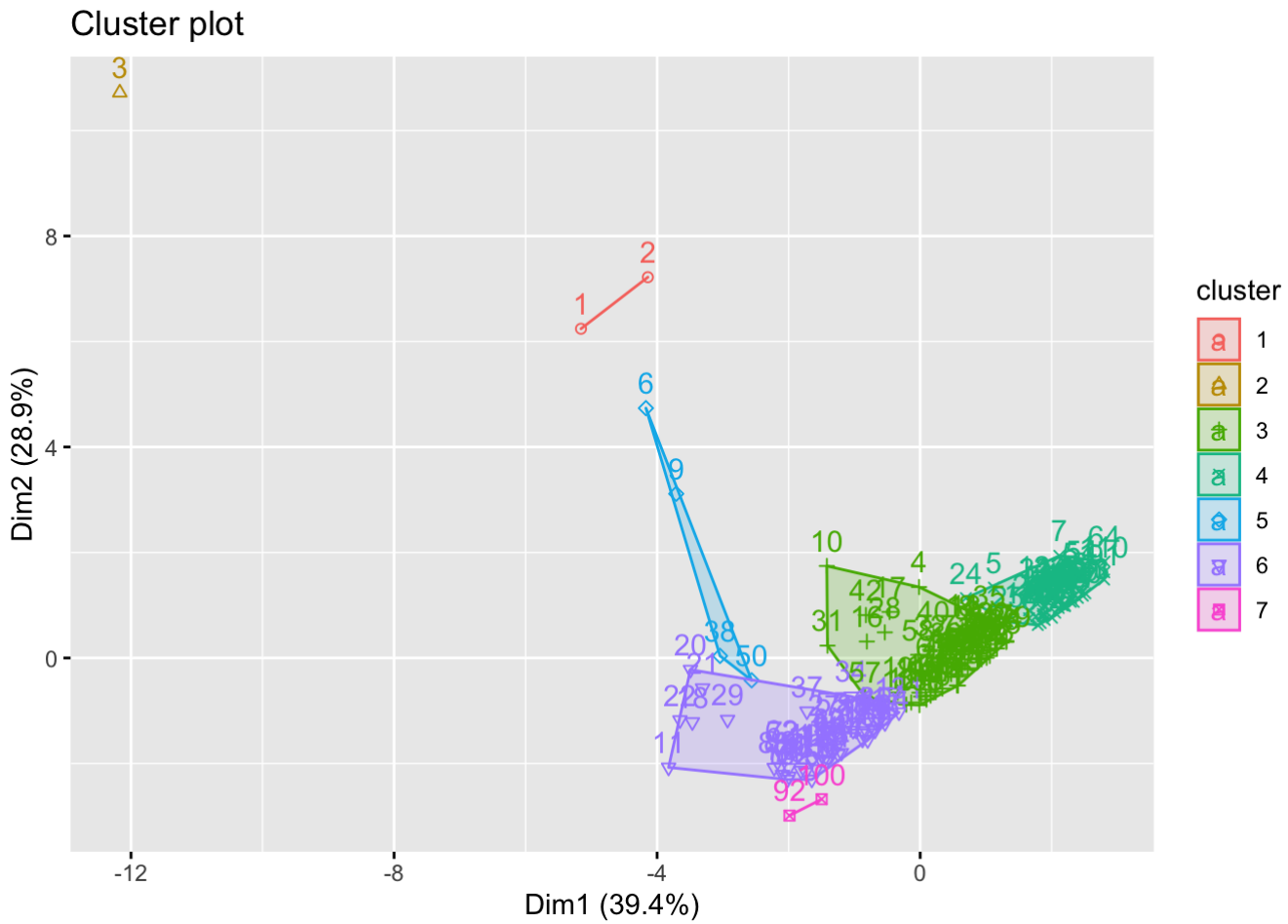
With our scree plot and variation table, we want to have an added variance that adds up to about 80 or 81%. So, here we will keep the first three components as we expected based off our interpretation.

Now we can visualize our first two components with either our cluster vizualization from above or the fvis_pca_ind function

```
# Visualize the individuals according to PC1 and PC2
fviz_pca_ind(pca_done,
             repel = TRUE) # Avoid text overlapping for the row number
```




```
fviz_cluster(pam_cluster, data = cluster_scale)
```



Now after running fvis_pca_ind, we recognized immediately that the visualization was the exact same as our clustering visualization above! However, there is not much else we can determine as of right now. Hence, we will perform summary statistics on our first two pc's to identify trends and abnormalities.

To interpret our first two principal components, we can look at the total variation that each variable has on it.

```
get_pca_var(pca_done)$coord %>%
  as.data.frame %>%
  select(Dim.1,Dim.2)
```

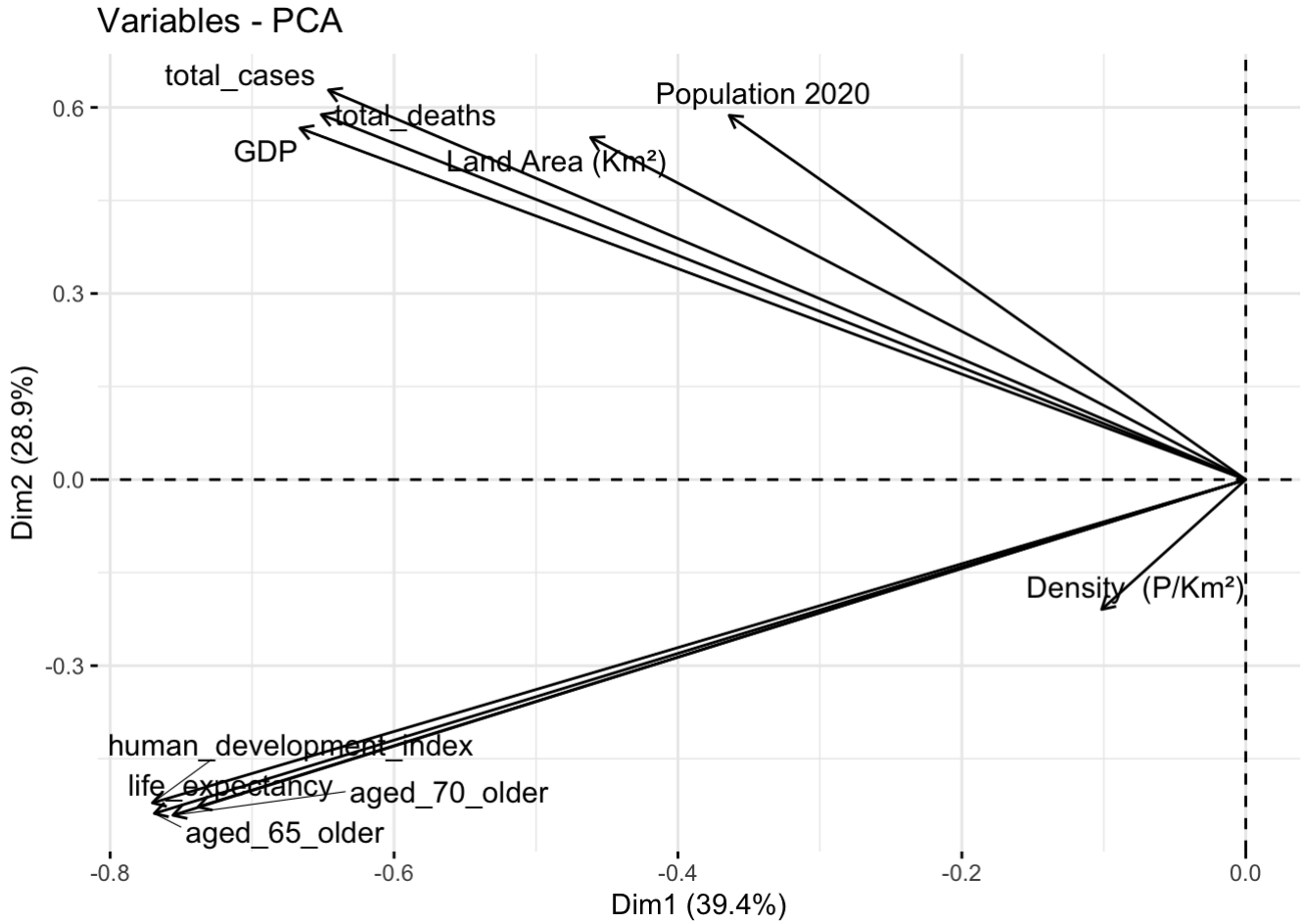
```
##                              Dim.1      Dim.2
## Population 2020          -0.3639183  0.5873049
## Density  (P/Km²)         -0.1015212 -0.2093779
## Land Area (Km²)          -0.4615487  0.5515885
## GDP                      -0.6663441  0.5667007
## total_cases              -0.6462864  0.6281480
## total_deaths             -0.6512859  0.5885358
## aged_65_older            -0.7688838 -0.5383198
## aged_70_older            -0.7557569 -0.5411840
## human_development_index -0.7702383 -0.5212998
## life_expectancy          -0.7380502 -0.5277579
```

For a variable to score low on the pc scale, that means that the variable has increasingly very low effect on the overall pc whereas if there is a high score of variability then there is alot more influence of that specific variable on the pc. Now based off the scree plot, we see that the first two PC's have a total variation of 39.4 + 28.9 or a

percentage of 68.3% of the total variation. A large number indicates a large positive correlation with the data. In pc1, the largest positive correllation is Density which is not even positive. However, aged_65_older has a very strong negative correllation. In pc2, total cases has alarge positive correllation whereas aged_70_older has an increasingly negative correllation. With a lower variance, then the pc's seperately cannot predict the variance of the data as a whole.

We also want to visualize the variances of these variables so we will use the appropriate PCA functions to do so.

```
fviz_pca_var(pca_done, col.var = "black", repel = TRUE) # Avoid text overlapping of the
 variable names
```



Classification and Cross-Validation

Now with our exploratory data and PCA being performed, we are ready to see if our data as a whole can be an accurate predictor for something Covid-19 related.

More specifically, if a countries ratio of deaths to cases is greater than the mean of that ratio of the whole data set, then we could confidently say that they were not able to contain the outbreak as well as other countries did

```
new_data%>%
  select(total_deaths,total_cases) %>% #keep only needed columns
  drop_na()%>%
  summarise(avg=(mean(total_deaths))/mean(total_cases)) # find the average ratio
```

```
## # A tibble: 1 × 1
##      avg
##    <dbl>
## 1 0.0228
```

Now, we will use an ifelse statement to predict a countries performance in controlling the outbreak. Particularly, we wanted to see if the ratio of total deaths to total cases could be used to predict the dataset. So we created new variables to portray the ratio of deaths to cases per country, and the outcome as 1 if equal to or higher than 0.02279165 and 0 if lower. Once again, a value of 1 for outcome indicates that the ratio was higher than the datasets average which indicates that they were not adequete in controlling the virus.

```
new_data2 <- new_data %>%
  mutate(ratiodtoc = (total_deaths/total_cases),
         outcome = ifelse(ratiodtoc >= 0.02279165, 1, 0)) %>%  # one indicates poor cont
ainment and 0 is great containment
  select_if(is.numeric) %>%
  select(1,2,3,4,5,6,10,11,13,14,15,16) %>%
  drop_na()

new_data2
```

```
## # A tibble: 157 × 12
##    Population …¹ Densi…² Land …³     GDP total…⁴ total…⁵ aged_…⁶ aged_…⁷ human…⁸
##            <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
##  1    1439323776     153  9.39e6 1.46e13  9.37e4    4634   10.6     5.93   0.761
##  2    1380004385     464  2.97e6 2.50e12  1.03e7  148994    5.99    3.41   0.645
##  3     331002651      36  9.15e6 1.93e13  2.02e7  350544   15.4     9.73   0.926
##  4     273523615     151  1.81e6 1.03e12  7.43e5   22138    5.32    3.05   0.718
##  5     220892340     287  7.71e5 3.20e11  4.82e5   10176    4.50    2.78   0.557
##  6     212559417      25  8.36e6 1.75e12  7.68e6  195072    8.55    5.06   0.765
##  7     206139589     226  9.11e5 4.94e11  8.76e4    1289    2.75    1.45   0.539
##  8     164689383    1265  1.30e5 2.71e11  5.14e5    7559    5.10    3.26   0.632
##  9     145934462       9  1.64e7 1.42e12  3.13e6   56271   14.2     9.39   0.824
## 10     128932753      66  1.94e6 1.15e12  1.43e6  125807    6.86    4.32   0.779
## # … with 147 more rows, 3 more variables: life_expectancy <dbl>,
## #   ratiodtoc <dbl>, outcome <dbl>, and abbreviated variable names
## #   ¹`Population 2020`, ²`Density  (P/Km²)`, ³`Land Area (Km²)`, ⁴total_cases,
## #   ⁵total_deaths, ⁶aged_65_older, ⁷aged_70_older, ⁸human_development_index
```

We will separate our entire dataset into a `train` set to train our model and a `test` set to test our model.

```
# Define a sampling process
sample_process <- sample(c(TRUE, FALSE), # take value TRUE or FALSE
                 nrow(new_data2), # for each row in new_data
                 replace = TRUE, # TRUE or FALSE can repeat
                 prob = c(0.5, 0.5)) # 50% TRUE, 50% FALSE

# Select values for the train set (corresponding to TRUEs in sample_process)
train <- new_data2[sample_process, ]

# Select values for the test set (corresponding to FALSEs in sample_process)
test <- new_data2[!sample_process, ]
```

After separating our data into training and testing sets, we have created probabilities of how accuarate each prediction will be! Now, we can fit a linear model to test our model

First, we fit the logistic regression model to predict the `outcome` based on all predictors in the `train` set:

```
# Fit a logistic with all predictors
new_data_log <- glm(outcome ~ ., data = train, family = "binomial")
summary(new_data_log)
```

```
##
## Call:
## glm(formula = outcome ~ ., family = "binomial", data = train)
##
## Deviance Residuals:
##          Min            1Q        Median            3Q           Max
## -0.000046422  -0.000000021  -0.000000021   0.000000021   0.000055979
##
## Coefficients:
##                                    Estimate            Std. Error z value
## (Intercept)                -378.5068961156862  194960.1690050416219  -0.002
## `Population 2020`             0.0000004546797       0.0007672083041   0.001
## `Density  (P/Km²)`           -0.0021849842167     151.8171045335054   0.000
## `Land Area (Km²)`             0.0000050778044       0.0253769863710   0.000
## GDP                          -0.0000000000423       0.0000001654284   0.000
## total_cases                  -0.0001056245540       0.4492282624039   0.000
## total_deaths                  0.0050197082799      21.4577737516264   0.000
## aged_65_older               -60.6122714132272   57181.4688292369901  -0.001
## aged_70_older                87.0327670256109   84174.7051991811604   0.001
## human_development_index    -100.3322056217796  443348.3845847615157   0.000
## life_expectancy               2.9494916317969    3689.1514070799685   0.001
## ratiodtoc                 12436.8356444781220 6505623.6112666903064   0.002
##                          Pr(>|z|)
## (Intercept)                 0.998
## `Population 2020`           1.000
## `Density  (P/Km²)`          1.000
## `Land Area (Km²)`           1.000
## GDP                         1.000
## total_cases                 1.000
## total_deaths                1.000
## aged_65_older               0.999
## aged_70_older               0.999
## human_development_index     1.000
## life_expectancy             0.999
## ratiodtoc                   0.998
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 99.921103109681  on 73  degrees of freedom
## Residual deviance:  0.000000011722  on 62  degrees of freedom
## AIC: 24
##
## Number of Fisher Scoring iterations: 25
```

The model has now been fitted with the training model

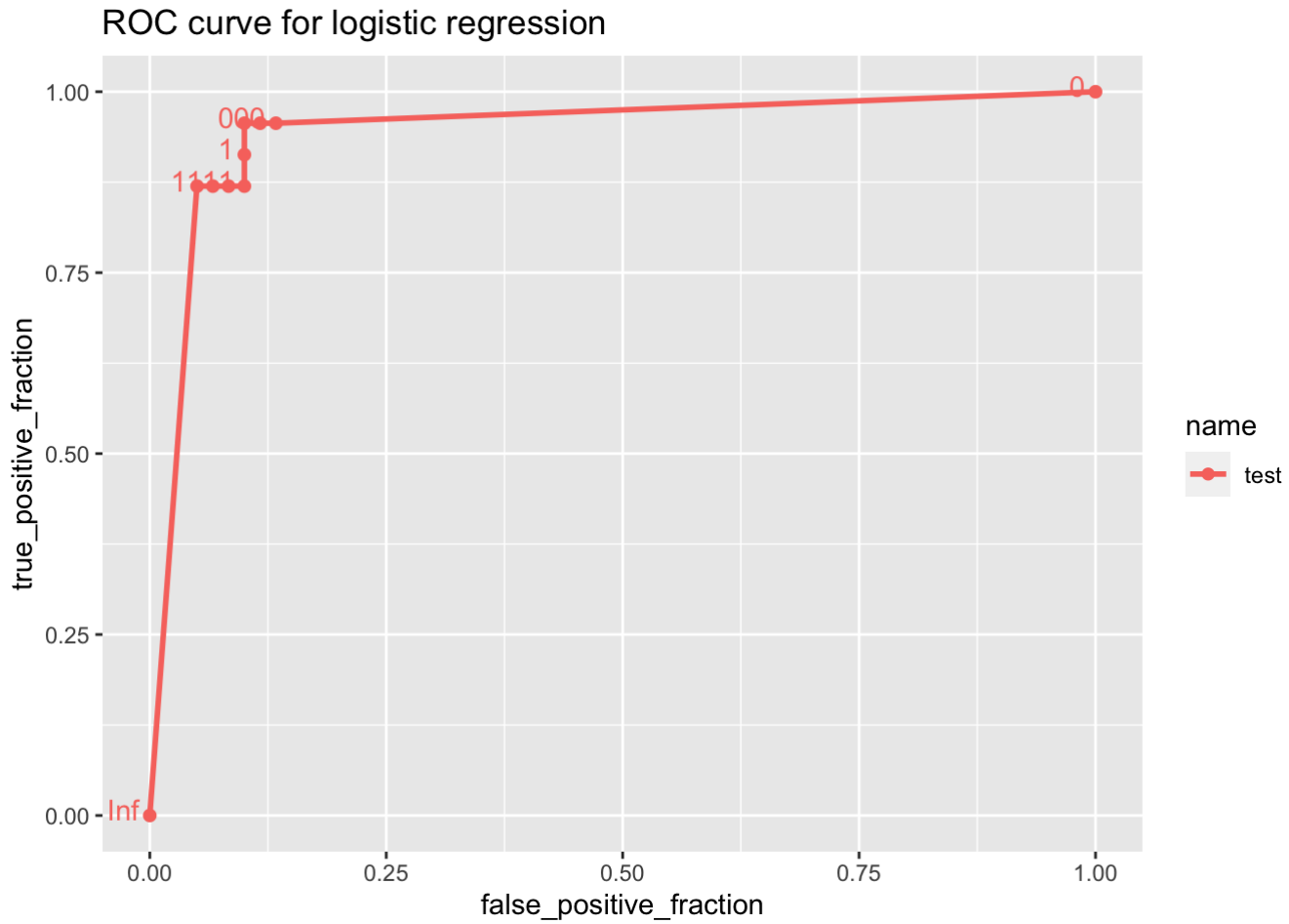
Now, let's compare the predictions on the `train` set and on the `test` set:

```
# Results in a data frame for train data
df_train <- data.frame(
  predictions = predict(new_data_log, newdata = train, type = "response"),
  outcome = train$outcome,
  name = "train")

# Results in a data frame for test data
df_test <- data.frame(
  predictions = predict(new_data_log, newdata = test, type = "response"),
  outcome = test$outcome,
  name = "test")

# Combined results
df_combined <- rbind(df_train, df_test)
```

This predicts based off the pobabilities.

Finally, let's evaluate the performance of our classifier on the `train` and `test` sets:

```
dfk<-ggplot(df_test) +
  geom_roc(aes(d = outcome, m = predictions, color = name), n.cuts = 10) +
  labs(title = "ROC curve for logistic regression")
dfk
```



Now to test our ROC curves performance, we will calculate the AUC. After calculating the AUC, we found it to be a value of 0.901667. THe AUC is far lower than what we expected, indicating that our model needs some adjusting or further testing.

k-fold cross-validation

```
# Choose number of folds
k = 10

# Randomly order rows in the dataset
data <- new_data2[sample(nrow(new_data2)), ]

# Create k folds from the dataset
folds <- cut(seq(1:nrow(data)), breaks = k, labels = FALSE)
```

Then we fit a logistic regression model and repeat the process for each k-fold:

```
# Initialize a vector to keep track of the performance
perf_k <- NULL

# Use a for loop to get diagnostics for each test set
for(i in 1:k){
  # Create train and test sets
  train <- data[folds != i, ] # all observations except in fold i
  test <- data[folds == i, ]  # observations in fold i

  # Train model on train set (all but fold i)
  new_data_log <- glm(outcome ~ ., data = train, family = "binomial")

  # Test model on test set (fold i)
  df <- data.frame(
    predictions = predict(new_data_log, newdata = test, type = "response"),
    outcome = test$outcome)

  # Consider the ROC curve for the test dataset
  ROC <- ggplot(df) +
    geom_roc(aes(d = outcome, m = predictions))

  # Get diagnostics for fold i (AUC)
  perf_k[i] <- calc_auc(ROC)$AUC
}
```

Finally, find the average performance on new data:

```
# Average performance
mean(perf_k)
```

```
## [1] 0.9826389
```

Now we are ready to conclude our findings for our classification and cross validation. We as a group were interested in creating a response variable dependent on the ratio of total deaths to total cases for applicaple countries. If a country had a ratio larger than the global average ratio of total deaths to total cases, the outcome variable was denoted with a 1. This indicates that the country was not able to contain the virus outbreak better than the global average. Conversely, a zero indicated that they were able to contain the spread better than the global average. Following that we were able to fit the logistic model. After creating the outcome variable, we split the data into training and testing sets to sample our data. Following that we were able to plot a roc curve of our testing set which was found to have a AUC of 0.901667, which was not as high as we would wont it to be. Now, we perform k-fold cross validations. With this, we found that the average performance of the k_folds was extremely higher with a value of 0.9897844, indicating that the cross validations performance was significantly higher than that of the logistic test set. Overall we found our models to be very accurate and found no signs of over-fitting.