Public Expenditure and Poverty

Team 1: Le Van Minh, Callista Stephine Yu, Muhammad Imtiyaaz B M A M , Thum Jun Long Issac

2022-11-18

1. Introduction

The project aim is to explore different models using the same dataset to evaluate which method will be the best to build a model for the dependent variable. The dataset we are using for this project has 1901 data values from 23 different variables. Our plan for this project is to first build a model with all the variables in the dataset. We will then follow it up by builidng a model without the missing values and build a ridge regression and Lasso regression model. Our second approach will be to selectively remove variables from the model and to build the best model using regularisation and regression. Our next approach was to add the variables into the model one at a time to see if a different model can be achieved. Lastly, we performed imputation on our model to tackle the missing values present in our dataset.We then compared all of this approaches to come to a conclusion.

Github repo: https://github.com/IntrovertHedgehog/PublicExpenditureAndPoverty

2. Data Characteristic

```
library(needs)

needs("dplyr", "MASS", "readr", "tidyr", "ggplot2",
    "e1071", "moments", "corrplot", "Hmisc", "PerformanceAnalytics",
    "mice", "car", "glmnet", "ggforce", "lmtest")

prioritize(dplyr)
```

2.1. Nature of Data

The data set is collection The World Bank Data, the variables of interest are extracted from the raw data files and combined into a single data frame for analysis. The final data set includes:

1. **country.code**: Country code
2. **country.name**: Country name
3. **year**: Year
4. **income**: Income class

- Low income (L)
- Lower middle income (LM)
- Upper middle income (UM)
- High income (H)

5. **reg**: Region
6. **pov**: Poverty headcount ratio based on cut-off value of $2.15 per day
7. **mpi**: Multidimensional Poverty Index
8. **edu.total**: Total expenditure on education (% of GDP)
9. **edu.pri**: Total expenditure on primary education (% of total education expenditure)
10. **edu.sec**: Total expenditure on secondary education (% of total education expenditure)
11. **edu.ter**: Total expenditure on tertiary education (% of total education expenditure)
12. **hlth**: Total expenditure on health (% of GDP)
13. **mil**: Total expenditure on military (% of GDP)
14. **fdi**: Foreign Direct Investment
15. **lbr.part**: Labour force participation (% of population ages 15+)
16. **unemp**: Unemployment rate
17. **pop.gwth.total**: Total population growth rate
18. **pop.gwth.rural**: Total rural population growth rate
19. **pop.gwth.urban**: Total urban population growth rate
20. **gdp.dflt**: GDP deflator
21. **gdr.eql**: Gender equality rating
22. **gcf**: Gross Capital Formation
23. **trade**: Trade = import + export (% of GDP)
24. **gdp.pc**: GDP per capita (current US$)

Data imports and combining:

```
# helper functions
importWDI <- function(filepath, value_name) {
    df <- read_csv(filepath, skip = 4)
```

```
    colnames(df) <- tolower(gsub(" ", ".", colnames(df)))

    df <- df %>%
        pivot_longer(5:ncol(.), names_to = "year",
            values_to = "value") %>%
        filter(!is.null(value) & !is.na(value)) %>%
        mutate(country.code = factor(country.code),
            country.name = factor(country.name), year = as.numeric(year)) %>%
        select(country.code, country.name, year, value)

    colnames(df)[4] <- value_name

    df
}

importRegionClass <- function(filepath) {
    df <- read_csv(filepath, skip = 4)

    colnames(df) <- tolower(gsub(" ", ".", colnames(df)))

    df %>%
        mutate(country.name = factor(country.name),
            region = factor(region)) %>%
        select(country.name, reg = region)
}

importIncomeClass <- function(filepath) {
    df <- read_csv(filepath, skip = 4)

    colnames(df) <- tolower(gsub(" ", ".", colnames(df)))

    df %>%
        pivot_longer(3:ncol(.), names_to = "year",
            values_to = "income") %>%
        filter(!is.null(income) & !is.na(income)) %>%
        mutate(country.code = factor(country.code),
            country.name = factor(country.name), year = as.numeric(year),
            income = factor(income)) %>%
        select(country.code, country.name, year, income)
}
```

```
# import data
setwd("../data")

poverty.headcount <- importWDI("poverty.headcount.215dollar.csv",
    "pov")
mpi <- importWDI("mpi.csv", "mpi")
education.expenditure.total <- importWDI("total.education.expenditure.csv",
    "edu.total")
education.expenditure.primary <- importWDI("primary.education.expenditure.csv",
    "edu.pri")
education.expenditure.secondary <- importWDI("secondary.education.expenditure.csv",
    "edu.sec")
```

```
education.expenditure.tertiary <- importWDI("tertiary.education.expenditure.csv",
    "edu.ter")
health.expenditure <- importWDI("health.expenditure.csv",
    "hlth")
military.expenditure <- importWDI("military.expenditure.csv",
    "mil")
fdi <- importWDI("fdi.csv", "fdi")
labour.force.participation <- importWDI("labour.force.participation.csv",
    "lbr.part")
unemployment.rate <- importWDI("unemployment.csv",
    "unemp")
population.growth <- importWDI("population.growth.csv",
    "pop.gwth.total")
rural.population.growth <- importWDI("rural.population.growth.csv",
    "pop.gwth.rural")
urban.population.growth <- importWDI("urban.population.growth.csv",
    "pop.gwth.urban")
gdp.deflator <- importWDI("gdp.deflator.csv", "gdp.dflt")
gender.equality <- importWDI("gender.equality.csv",
    "gdr.eql")
gross.capital.formation <- importWDI("gross.capital.formation.csv",
    "gcf")
trade <- importWDI("trade.csv", "trade")
region.class <- importRegionClass("region.class.csv")
income.class <- importIncomeClass("income.class.csv")
gdp.pc <- importWDI("gdp.pc.csv", "gdp.pc")

setwd("../src")
```

We found that the data sets collected from World Bank's Data helpdesk and The World Bank's Data have different naming convention for certain countries (e.g. "Czechia" vs. "Czechnia Republic"). So we need to rename these countries to avoid some error when joining.

Furthermore, WDI's data sets rate also account for non-country (e.g. country.name = "Low income" or "South Asia"). These special groups are not in our scope of interest, which is national, so we eliminate them.

```
# using poverty.headcount as a naming standard
# (as other data from WDI also use this
# convention) join a subset of data to process
# the names
d <- poverty.headcount %>%
    select(country.name) %>%
    mutate(inPov = T) %>%
    full_join(income.class %>%
        select(country.name) %>%
        mutate(inIncome = T), by = "country.name") %>%
    full_join(region.class %>%
        select(country.name) %>%
        mutate(inReg = T), by = "country.name") %>%
    mutate(inPov = !is.na(inPov), inIncome = !is.na(inIncome),
        inReg = !is.na(inReg))

d
```

```
## # A tibble: 62,759 x 4
##    country.name inPov inIncome inReg
##    <fct>        <lgl> <lgl>    <lgl>
##  1 Angola       TRUE  TRUE     TRUE
##  2 Angola       TRUE  TRUE     TRUE
##  3 Angola       TRUE  TRUE     TRUE
##  4 Angola       TRUE  TRUE     TRUE
##  5 Angola       TRUE  TRUE     TRUE
##  6 Angola       TRUE  TRUE     TRUE
##  7 Angola       TRUE  TRUE     TRUE
##  8 Angola       TRUE  TRUE     TRUE
##  9 Angola       TRUE  TRUE     TRUE
## 10 Angola       TRUE  TRUE     TRUE
## # ... with 62,749 more rows
## # i Use 'print(n = ...)' to see more rows
```

First, remove special economic groups from `poverty.headcount`. We figured these regions will not appear in `income.class` or `region.class`, so we might find something from looking at the countries **only** appear in `poverty.headcount`.

```
d %>%
    filter(inPov & (!inIncome | !inReg)) %>%
    distinct(country.name)
```

```
## # A tibble: 18 x 1
##    country.name
##    <fct>
##  1 Cote d'Ivoire
##  2 Czechia
##  3 East Asia & Pacific
##  4 Europe & Central Asia
##  5 Fragile and conflict affected situations
##  6 High income
##  7 IDA total
##  8 Latin America & Caribbean
##  9 Low income
## 10 Lower middle income
## 11 Low & middle income
## 12 Middle East & North Africa
## 13 South Asia
## 14 Sub-Saharan Africa
## 15 Sao Tome and Principe
## 16 Turkiye
## 17 Upper middle income
## 18 World
```

Lucky! We can look through these 18 results and compose a list of special regions.

```
spec.reg <- c("Fragile and conflict affected situations",
    "IDA total", "World", "East Asia & Pacific", "Europe & Central Asia",
    "Latin America & Caribbean", "Middle East & North Africa",
    "South Asia", "Sub-Saharan Africa", "Low income",
    "Low & middle income", "Lower middle income", "Upper middle income",
    "High income")
```

Then, we rename those countries with inconsistent naming convention. Since we should only care about countries whose poverty headcount is available, reusing the list generated above, we can identify:

1. Cote d'Ivoire (also Côte d'Ivoire)
2. Czechia (also Czechoslovakia or Czech Republic)
3. Curacao (also Curaçao)
4. Turkiye (formerly known as Turkey, also Türkiye)
5. Sao Tome and Principe (also São Tomé and Príncipe)

```
# mapping standard name and variation
nameMap <- tibble(standard = c("Cote d'Ivoire", "Czechia",
    "Czechia", "Curacao", "Turkiye", "Turkiye", "Sao Tome and Principe"),
    variation = c("Côte d'Ivoire", "Czechoslovakia",
        "Czech Republic", "Curaçao", "Turkey", "Türkiye",
        "São Tomé and Príncipe"))

correctName <- function(name) {
    tibble(name = name) %>%
        left_join(nameMap, by = c(name = "variation")) %>%
        mutate(standard = ifelse(is.na(standard), name,
            standard)) %>%
        select(standard) %>%
        pull()
}

orig.name <- c("Vietnam", "China", "Turkey", "Czechia Republic")
correctName(orig.name)
```

```
## [1] "Vietnam"          "China"            "Turkiye"          "Czechia Republic"
```

Let's test this out!

```
d <- poverty.headcount %>%
  # correct name here
  mutate(country.name = correctName(country.name)) %>%
  select(country.name) %>%
  mutate(inPov = T) %>%
  full_join(income.class %>%
              # correct name here
              mutate(country.name = correctName(country.name)) %>%
              select(country.name) %>%
              mutate(inIncome = T), by = "country.name") %>%
  full_join(region.class %>%
              # correct name here
              mutate(country.name = correctName(country.name)) %>%
              select(country.name) %>%
              mutate(inReg = T), by = "country.name") %>%
  filter(!(country.name %in% spec.reg)) %>%
  mutate(inPov = !is.na(inPov), inIncome = !is.na(inIncome), inReg = !is.na(inReg))

# countries not in region list, but is in Pov list
d %>%
  filter(!inReg & inPov) %>%
```

```
  distinct(country.name) %>%
  nrow()
```

```
## [1] 0
```

```
# countries not in income list, but is in Pov list
d %>%
  filter(!inIncome & inPov) %>%
  distinct(country.name) %>%
  nrow()
```

```
## [1] 0
```

We are *pretty* confident that there's no inconsistent naming left unprocessed in the data sets.

```
# Rename countries in all data sets.
poverty.headcount <- poverty.headcount %>%
    mutate(country.name = correctName(country.name))
mpi <- mpi %>%
    mutate(country.name = correctName(country.name))
education.expenditure.total <- education.expenditure.total %>%
    mutate(country.name = correctName(country.name))
education.expenditure.primary <- education.expenditure.primary %>%
    mutate(country.name = correctName(country.name))
education.expenditure.secondary <- education.expenditure.secondary %>%
    mutate(country.name = correctName(country.name))
education.expenditure.tertiary <- education.expenditure.tertiary %>%
    mutate(country.name = correctName(country.name))
health.expenditure <- health.expenditure %>%
    mutate(country.name = correctName(country.name))
military.expenditure <- military.expenditure %>%
    mutate(country.name = correctName(country.name))
fdi <- fdi %>%
    mutate(country.name = correctName(country.name))
labour.force.participation <- labour.force.participation %>%
    mutate(country.name = correctName(country.name))
unemployment.rate <- unemployment.rate %>%
    mutate(country.name = correctName(country.name))
population.growth <- population.growth %>%
    mutate(country.name = correctName(country.name))
rural.population.growth <- rural.population.growth %>%
    mutate(country.name = correctName(country.name))
urban.population.growth <- urban.population.growth %>%
    mutate(country.name = correctName(country.name))
gdp.deflator <- gdp.deflator %>%
    mutate(country.name = correctName(country.name))
gender.equality <- gender.equality %>%
    mutate(country.name = correctName(country.name))
gross.capital.formation <- gross.capital.formation %>%
    mutate(country.name = correctName(country.name))
trade <- trade %>%
    mutate(country.name = correctName(country.name))
```

```
region.class <- region.class %>%
    mutate(country.name = correctName(country.name))
income.class <- income.class %>%
    mutate(country.name = correctName(country.name))
gdp.pc <- gdp.pc %>%
    mutate(country.name = correctName(country.name))
```

Join the data

```
countries <- poverty.headcount %>%
    # We used a full join here so we can conduct
    # a separate analysis on mpi later
full_join(mpi, by = c("country.name", "country.code",
    "year")) %>%
    left_join(income.class, c("country.name", "country.code",
        "year")) %>%
    left_join(region.class, by = "country.name") %>%
    left_join(education.expenditure.total, by = c("country.name",
        "country.code", "year")) %>%
    left_join(education.expenditure.primary, by = c("country.name",
        "country.code", "year")) %>%
    left_join(education.expenditure.secondary, by = c("country.name",
        "country.code", "year")) %>%
    left_join(education.expenditure.tertiary, by = c("country.name",
        "country.code", "year")) %>%
    left_join(health.expenditure, by = c("country.name",
        "country.code", "year")) %>%
    left_join(military.expenditure, by = c("country.name",
        "country.code", "year")) %>%
    left_join(fdi, by = c("country.name", "country.code",
        "year")) %>%
    left_join(labour.force.participation, by = c("country.name",
        "country.code", "year")) %>%
    left_join(unemployment.rate, by = c("country.name",
        "country.code", "year")) %>%
    left_join(population.growth, by = c("country.name",
        "country.code", "year")) %>%
    left_join(rural.population.growth, by = c("country.name",
        "country.code", "year")) %>%
    left_join(urban.population.growth, by = c("country.name",
        "country.code", "year")) %>%
    left_join(gdp.deflator, by = c("country.name",
        "country.code", "year")) %>%
    left_join(gender.equality, by = c("country.name",
        "country.code", "year")) %>%
    left_join(gross.capital.formation, by = c("country.name",
        "country.code", "year")) %>%
    left_join(trade, by = c("country.name", "country.code",
        "year")) %>%
    left_join(gdp.pc, by = c("country.name", "country.code",
        "year")) %>%
    # filter special groups
filter(!(country.name %in% spec.reg))
```

Data preview

```
head(countries)
```

```
## # A tibble: 6 x 24
##   count~1 count~2  year   pov   mpi income reg   edu.t~3 edu.pri edu.sec edu.ter
##   <fct>   <chr>   <dbl> <dbl> <dbl> <fct>  <fct>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 AGO     Angola   2000  21.4    NA L      Sub-~    2.61      NA      NA      NA
## 2 AGO     Angola   2008  14.6    NA LM     Sub-~    2.69      NA      NA      NA
## 3 AGO     Angola   2018  31.1    NA LM     Sub-~    2.04      NA      NA      NA
## 4 ALB     Albania  1996   0.5    NA LM     Euro~    3.08      NA      NA      NA
## 5 ALB     Albania  2002   1.1    NA LM     Euro~    3.12      NA      NA      NA
## 6 ALB     Albania  2005   0.6    NA LM     Euro~    3.28      NA      NA      NA
## # ... with 13 more variables: hlth <dbl>, mil <dbl>, fdi <dbl>, lbr.part <dbl>,
## #   unemp <dbl>, pop.gwth.total <dbl>, pop.gwth.rural <dbl>,
## #   pop.gwth.urban <dbl>, gdp.dflt <dbl>, gdr.eql <dbl>, gcf <dbl>,
## #   trade <dbl>, gdp.pc <dbl>, and abbreviated variable names 1: country.code,
## #   2: country.name, 3: edu.total
## # i Use `colnames()` to see all variable names
```

```
str(countries)
```

```
## tibble [1,901 x 24] (S3: tbl_df/tbl/data.frame)
##  $ country.code  : Factor w/ 272 levels "AGO","ALB","ARE",..: 1 1 1 2 2 2 2 2 2 2 ...
##  $ country.name  : chr [1:1901] "Angola" "Angola" "Angola" "Albania" ...
##  $ year          : num [1:1901] 2000 2008 2018 1996 2002 ...
##  $ pov           : num [1:1901] 21.4 14.6 31.1 0.5 1.1 0.6 0.2 0.6 1 0.1 ...
##  $ mpi           : num [1:1901] NA NA NA NA NA NA NA NA NA NA ...
##  $ income        : Factor w/ 4 levels "H","L","LM","UM": 2 3 3 3 3 3 3 4 4 4 ...
##  $ reg           : Factor w/ 7 levels "East Asia & Pacific",..: 7 7 7 2 2 2 2 2 2 2 ...
##  $ edu.total     : num [1:1901] 2.61 2.69 2.04 3.08 3.12 ...
##  $ edu.pri       : num [1:1901] NA NA NA NA NA ...
##  $ edu.sec       : num [1:1901] NA NA NA NA NA ...
##  $ edu.ter       : num [1:1901] NA NA NA NA NA ...
##  $ hlth          : num [1:1901] 1.91 3.32 2.54 NA 6.91 ...
##  $ mil           : num [1:1901] 6.39 3.57 1.87 1.38 1.32 ...
##  $ fdi           : num [1:1901] 8.79e+08 1.68e+09 -6.46e+09 9.01e+07 1.35e+08 ...
##  $ lbr.part      : num [1:1901] NA NA NA 38.8 59.6 ...
##  $ unemp         : num [1:1901] NA NA NA 12.3 15.8 ...
##  $ pop.gwth.total: num [1:1901] 3.277 3.711 3.276 -0.622 -0.3 ...
##  $ pop.gwth.rural: num [1:1901] 0.921 1.91 1.338 -1.546 -2.169 ...
##  $ pop.gwth.urban: num [1:1901] 5.682 5.02 4.312 0.812 2.181 ...
##  $ gdp.dflt      : num [1:1901] 418.02 19.37 28.17 38.17 3.65 ...
##  $ gdr.eql       : num [1:1901] NA 3 NA NA NA 4 NA NA NA NA ...
##  $ gcf           : num [1:1901] 30.5 30.8 17.9 18.1 35.3 ...
##  $ trade         : num [1:1901] 152.5 121.4 66.4 44.9 68.5 ...
##  $ gdp.pc        : num [1:1901] 557 4081 2525 1010 1425 ...
```

```
summary(countries)
```

```
##   country.code  country.name            year           pov            mpi
##  BRA    :  36   Length:1901        Min.   :1967   Min.   : 0.0   Min.   : 2
```

```
##  CRI    :  34   Class :character   1st Qu.:2002   1st Qu.: 0.2   1st Qu.:18
##  ARG    :  32   Mode  :character   Median :2009   Median : 1.5   Median :25
##  USA    :  32                      Mean   :2007   Mean   :10.0   Mean   :27
##  DEU    :  30                      3rd Qu.:2014   3rd Qu.:11.6   3rd Qu.:33
##  HND    :  30                      Max.   :2021   Max.   :91.5   Max.   :74
##  (Other):1707                                     NA's   :58     NA's   :1446
##   income                            reg        edu.total      edu.pri
##  H   :644   East Asia & Pacific       :167   Min.   : 1    Min.   : 1
##  L   :253   Europe & Central Asia     :883   1st Qu.: 4    1st Qu.:24
##  LM  :501   Latin America & Caribbean :416   Median : 5    Median :30
##  UM  :438   Middle East & North Africa:107   Mean   : 5    Mean   :32
##  NA's: 65   North America             : 50   3rd Qu.: 5    3rd Qu.:38
##             South Asia                : 61   Max.   :16    Max.   :70
##             Sub-Saharan Africa        :217   NA's   :596   NA's   :1090
##     edu.sec        edu.ter         hlth           mil            fdi
##  Min.   : 3     Min.   : 0    Min.   : 2    Min.   : 0.0   Min.   :-3.44e+11
##  1st Qu.:30     1st Qu.:17    1st Qu.: 5    1st Qu.: 1.0   1st Qu.: 2.98e+08
##  Median :36     Median :21    Median : 7    Median : 1.5   Median : 1.71e+09
##  Mean   :36     Mean   :21    Mean   : 7    Mean   : 1.8   Mean   : 1.60e+10
##  3rd Qu.:41     3rd Qu.:25    3rd Qu.: 9    3rd Qu.: 2.1   3rd Qu.: 9.82e+09
##  Max.   :72     Max.   :50    Max.   :18    Max.   :19.4   Max.   : 7.34e+11
##  NA's   :1094   NA's   :963   NA's   :464   NA's   :105    NA's   :17
##     lbr.part       unemp      pop.gwth.total  pop.gwth.rural  pop.gwth.urban
##  Min.   :30    Min.   : 0.2   Min.   :-3.63   Min.   :-8.56   Min.   :-4.08
##  1st Qu.:56    1st Qu.: 4.5   1st Qu.: 0.27   1st Qu.:-0.86   1st Qu.: 0.51
##  Median :61    Median : 6.9   Median : 1.03   Median :-0.02   Median : 1.48
##  Mean   :61    Mean   : 8.1   Mean   : 1.06   Mean   : 0.00   Mean   : 1.69
##  3rd Qu.:65    3rd Qu.:10.1   3rd Qu.: 1.78   3rd Qu.: 0.96   3rd Qu.: 2.66
##  Max.   :93    Max.   :49.7   Max.   : 5.61   Max.   : 4.60   Max.   :13.80
##  NA's   :320   NA's   :267                    NA's   :14      NA's   :13
##     gdp.dflt       gdr.eql          gcf           trade         gdp.pc
##  Min.   : -26   Min.   :2      Min.   : 0.0   Min.   :  1   Min.   :   120
##  1st Qu.:   2   1st Qu.:3      1st Qu.:19.6   1st Qu.: 51   1st Qu.:  1928
##  Median :   4   Median :4      Median :22.7   Median : 73   Median :  6032
##  Mean   :  16   Mean   :4      Mean   :23.9   Mean   : 84   Mean   : 15220
##  3rd Qu.:   9   3rd Qu.:4      3rd Qu.:26.8   3rd Qu.:105   3rd Qu.: 21490
##  Max.   :3334   Max.   :5      Max.   :69.5   Max.   :380   Max.   :123679
##  NA's   :18     NA's   :1635   NA's   :72     NA's   :57    NA's   :8
```

There's no NA in `reg`, which is a sign that all naming in the data is remedied. There's some expected NAs in `income` and `pov`, as these data are collected by year. There's a substantial amount of missing data in `mpi`, as this is a relative new concept. We will address the nature, and processing of missing data in the next sections.

2.2. Missing values

As observed from the summary above, the data set contains a lot of missing values in some of the variables.

```
mean(is.na(countries))
```

```
## [1] 0.182
```

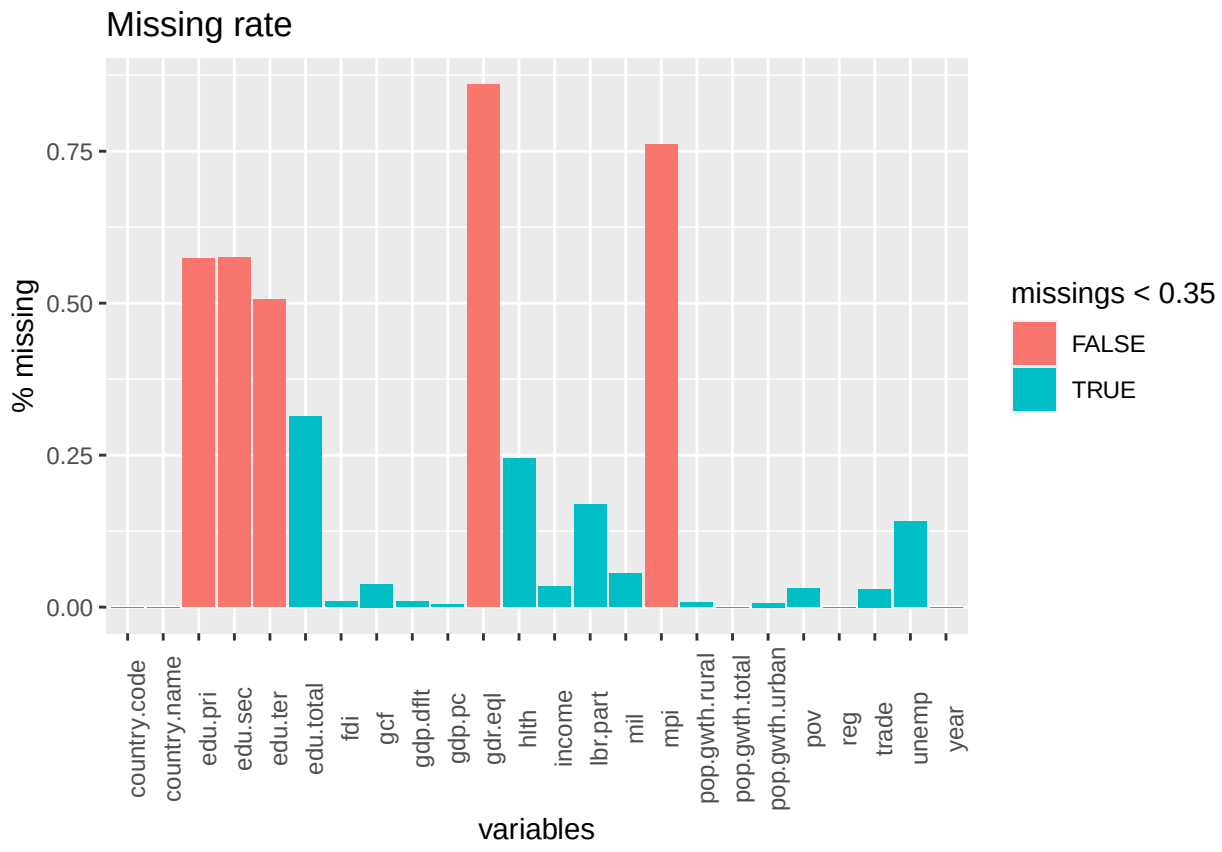
About 19% of the data set is missing.

```
nCompleteObs <- sum(complete.cases(countries))
print(paste("No. of complete cases:", nCompleteObs))
```

```
## [1] "No. of complete cases: 3"
```

There are only 3 complete cases where all the variable is available. This is nowhere near acceptable to conduct any meaningful analysis. Therefore, we need to eliminate some variables for a more balance data set.

```
missings <- colMeans(is.na(countries))
ggplot(mapping = aes(x = names(missings), y = missings,
    fill = missings < 0.35)) + geom_bar(stat = "identity") +
    ggtitle("Missing rate") + xlab("variables") + ylab("% missing") +
    theme(axis.text.x = element_text(size = 9, angle = 90))
```




```
missings[missings > 0.35]
```

```
##     mpi edu.pri edu.sec edu.ter gdr.eql
##   0.761   0.573   0.575   0.507   0.860
```

There are 5 variables with missing rate >35%.

expenditure in primary, secondary, and tertiary edication can be very useful and relevant information to predict poverty reduction (Akbar et al. 2019). However, we would like to exclude these variables from some

first analyses to make use of the richer set of data. We can conduct a separate analysis with these variable to gain more insight.

```
# variables with high missing rate
hMiss <- names(missings[missings > 0.35])
# exclude these variables in countries1
countries1 <- countries %>%
    select(!hMiss) %>%
    filter(!is.na(pov))
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use ‘all_of(hMiss)‘ instead of ‘hMiss‘ to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
str(countries1)
```

```
## tibble [1,843 x 19] (S3: tbl_df/tbl/data.frame)
##  $ country.code  : Factor w/ 272 levels "AGO","ALB","ARE",..: 1 1 1 2 2 2 2 2 2 2 ...
##  $ country.name  : chr [1:1843] "Angola" "Angola" "Angola" "Albania" ...
##  $ year          : num [1:1843] 2000 2008 2018 1996 2002 ...
##  $ pov           : num [1:1843] 21.4 14.6 31.1 0.5 1.1 0.6 0.2 0.6 1 0.1 ...
##  $ income        : Factor w/ 4 levels "H","L","LM","UM": 2 3 3 3 3 3 3 4 4 4 ...
##  $ reg           : Factor w/ 7 levels "East Asia & Pacific",..: 7 7 7 2 2 2 2 2 2 2 ...
##  $ edu.total     : num [1:1843] 2.61 2.69 2.04 3.08 3.12 ...
##  $ hlth          : num [1:1843] 1.91 3.32 2.54 NA 6.91 ...
##  $ mil           : num [1:1843] 6.39 3.57 1.87 1.38 1.32 ...
##  $ fdi           : num [1:1843] 8.79e+08 1.68e+09 -6.46e+09 9.01e+07 1.35e+08 ...
##  $ lbr.part      : num [1:1843] NA NA NA 38.8 59.6 ...
##  $ unemp         : num [1:1843] NA NA NA 12.3 15.8 ...
##  $ pop.gwth.total: num [1:1843] 3.277 3.711 3.276 -0.622 -0.3 ...
##  $ pop.gwth.rural: num [1:1843] 0.921 1.91 1.338 -1.546 -2.169 ...
##  $ pop.gwth.urban: num [1:1843] 5.682 5.02 4.312 0.812 2.181 ...
##  $ gdp.dflt      : num [1:1843] 418.02 19.37 28.17 38.17 3.65 ...
##  $ gcf           : num [1:1843] 30.5 30.8 17.9 18.1 35.3 ...
##  $ trade         : num [1:1843] 152.5 121.4 66.4 44.9 68.5 ...
##  $ gdp.pc        : num [1:1843] 557 4081 2525 1010 1425 ...
```

Re-evaluate the `countries1` set.

```
mean(is.na(countries1))
```

```
## [1] 0.0547
```

```
sum(complete.cases(countries1))
```

```
## [1] 937
```

```
mean(complete.cases(countries1))
```

```
## [1] 0.508
```

On average, each column has 6% missing rate, results in 937 complete data point (i.e. 49%). This can be a sufficient number for the analysis. However, the missing data can induce loss of power due to the reduced sample size, and some other biases depending on which variables is missing.

```
# complete rate of data by regions
countries1 %>%
    mutate(isComplete = complete.cases(.)) %>%
    group_by(reg) %>%
    summarise(complete.rate = mean(isComplete)) %>%
    arrange(desc(complete.rate))
```

```
## # A tibble: 7 x 2
##   reg                          complete.rate
##   <fct>                                <dbl>
## 1 Europe & Central Asia                0.646
## 2 Latin America & Caribbean            0.505
## 3 Middle East & North Africa           0.462
## 4 East Asia & Pacific                  0.437
## 5 South Asia                           0.245
## 6 Sub-Saharan Africa                   0.192
## 7 North America                        0.14
```

Countries from North America, Sub-Saharan Africa, and South Asia have the highest rate of missing data. We suspect that Sub-Saharan Africa, and South Asia are comparably less accessible regions. We also know that Americans don't like filling out forms, so their high rate of missing data is understandable as well.

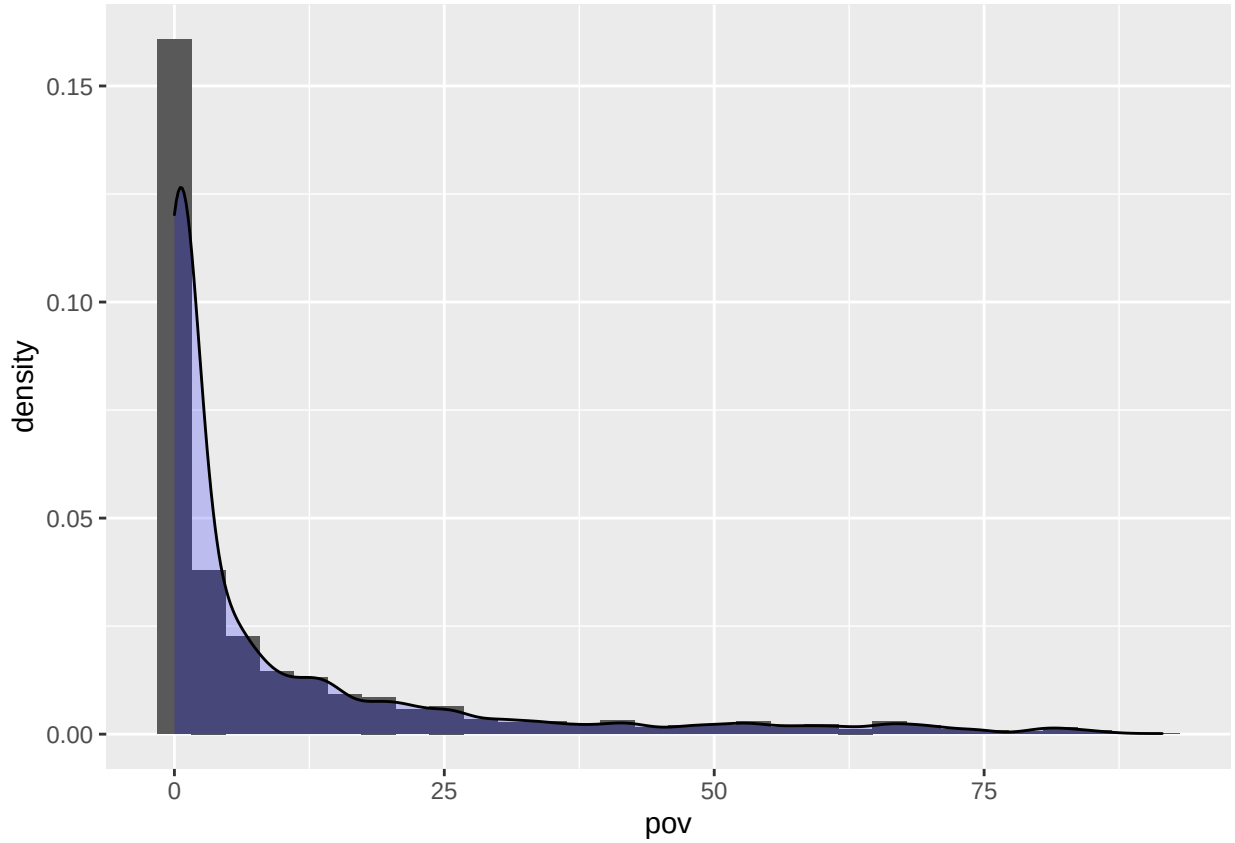
Still, we need to find a way to address this issue. we propose several approaches:

1. **Use complete cases**: Only use the complete cases for the analysis. This is a straightforward approach, but doesn't resolve the bias resulted from the massive loss of data.
2. **Selectively remove variables with high missing rate**: Start with all the variable in the data set, then step-by-step remove insignificant, or frequently missing variables to use as much data as possible.
3. **Forward variable selection**: Start with 0 variable, then selectively add potentially important variable to the data set to achieve optimal model.
4. **Imputation**: The idea is to replace the missing observations on the response or the predictors with artificial values that try to preserve the data set structure. This is a quite complex topic of its own. You can read more from Arel-Bundock and Pelc (2018).

2.3. Descriptive Analytics

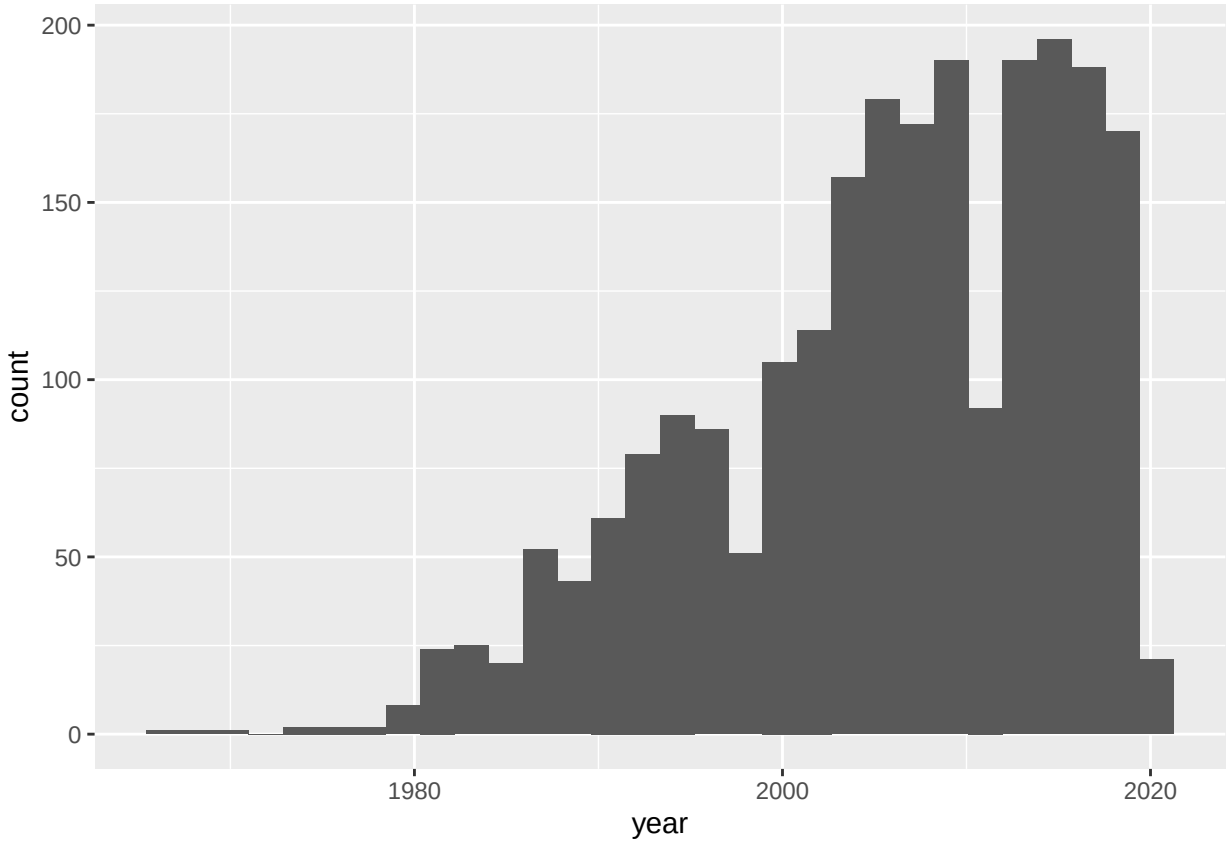
Distribution of the predicted variable `pov`

```
ggplot(countries, aes(x = pov)) + geom_histogram(aes(y = ..density..),
    bins = 30) + geom_density(alpha = 0.2, fill = "blue")
```



The graph displays a decreasing rate as poverty indicator increasing. This might not be representative of the current state of poverty in the world, but of the number presented in our data. For example, more recent data is likely to be more inclusive than ancient data, when poverty is more prevalent. We should look at data from the same period.

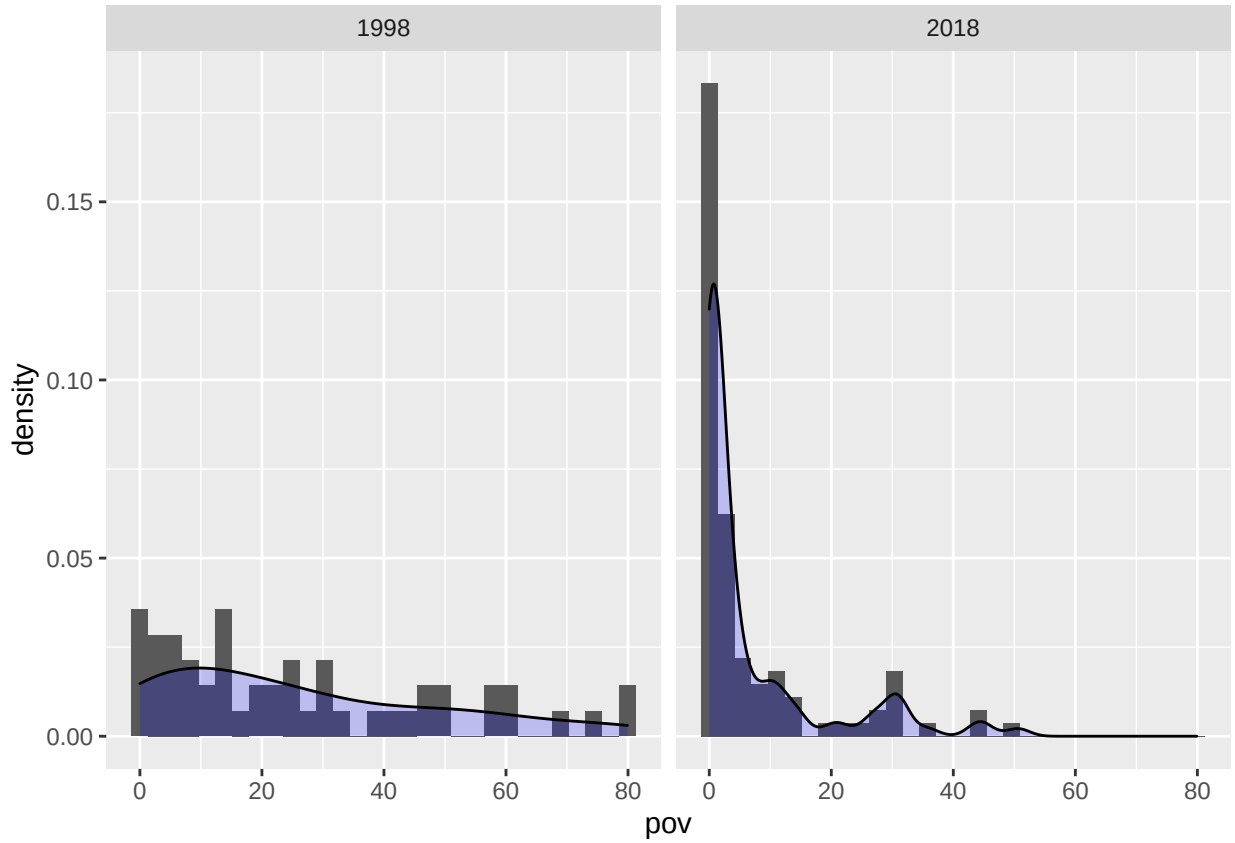
```
# number of data point available from 1967 to
# 2021
ggplot(poverty.headcount, aes(x = year)) + geom_histogram(bins = 30)
```




```
# pov data from 1998 and 2018
pov.98.18 <- poverty.headcount %>%
    filter(year == 1998 | year == 2018)
pov.98.18 %>%
    group_by(year) %>%
    summarise(sum = n())
```

```
## # A tibble: 2 x 2
##    year   sum
##   <dbl> <int>
## 1  1998    51
## 2  2018    99
```

```
ggplot(pov.98.18, aes(x = pov)) + geom_histogram(aes(y = ..density..),
    bins = 30) + geom_density(alpha = 0.2, fill = "blue") +
    facet_grid(cols = vars(year))
```



The graph for 1998 has a much gentler slope, meaning poverty was more popular during that time, as predicted from out intuition. What about the general progress of the world?

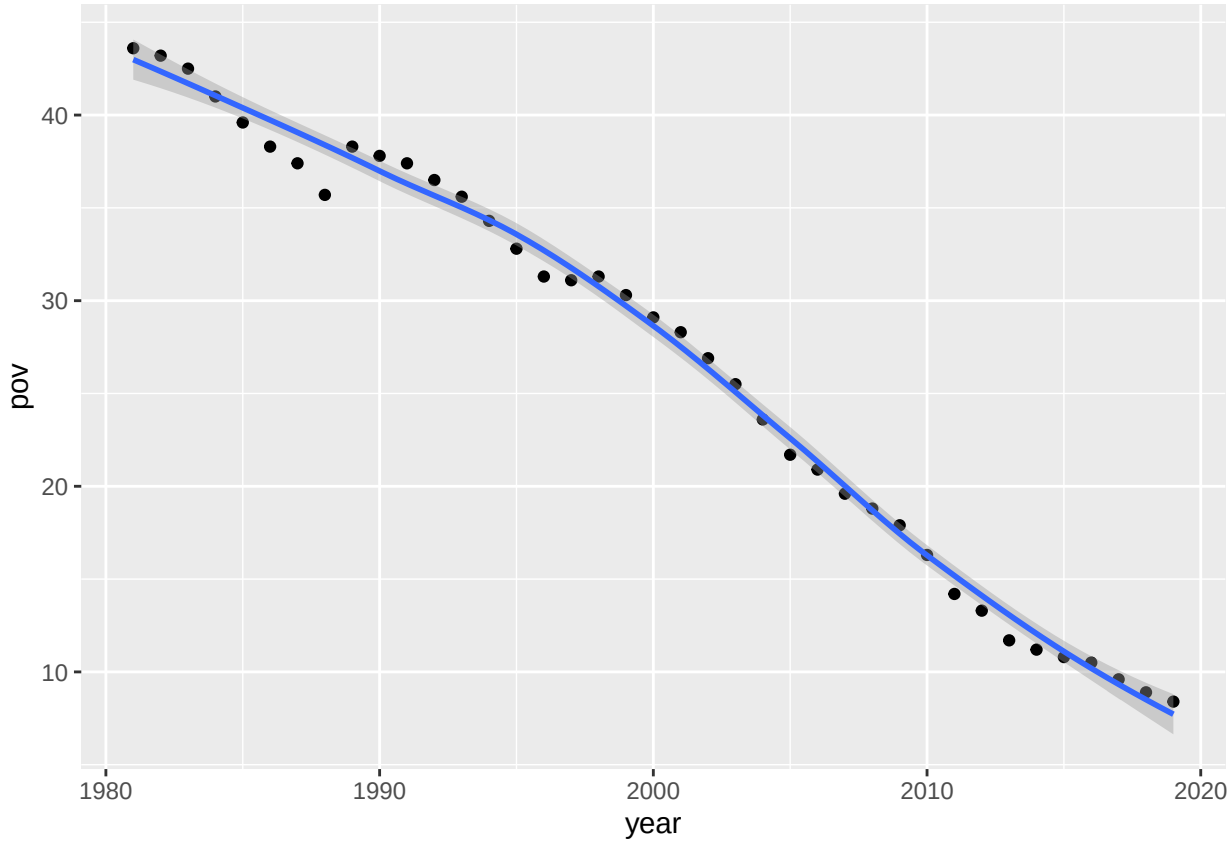
```
# Re-import pov and only take special regions
# geographic
geo.regs <- c("EAS", "ECS", "LCN", "MEA", "SAS", "SSF",
    "WLD")
# economics
eco.regs <- c("HIC", "LIC", "LMC", "LMY", "UMC")

pov.reg <- importWDI("../data/poverty.headcount.215dollar.csv",
    "pov") %>%
    filter(country.code %in% c(geo.regs, eco.regs)) %>%
    mutate(type = ifelse(country.code %in% geo.regs,
        "Geographic", "Economics"))
```

```
## New names:
## Rows: 266 Columns: 67
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (4): Country Name, Country Code, Indicator Name, Indicator Code dbl (50): 1967,
## 1969, 1971, 1974, 1975, 1977, 1978, 1979, 1980, 1981, 1982, ... lgl (13): 1960,
## 1961, 1962, 1963, 1964, 1965, 1966, 1968, 1970, 1972, 1973, ...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...67`
```

```
pov.reg %>%
    distinct(country.code, country.name, type) %>%
    arrange(type)
```

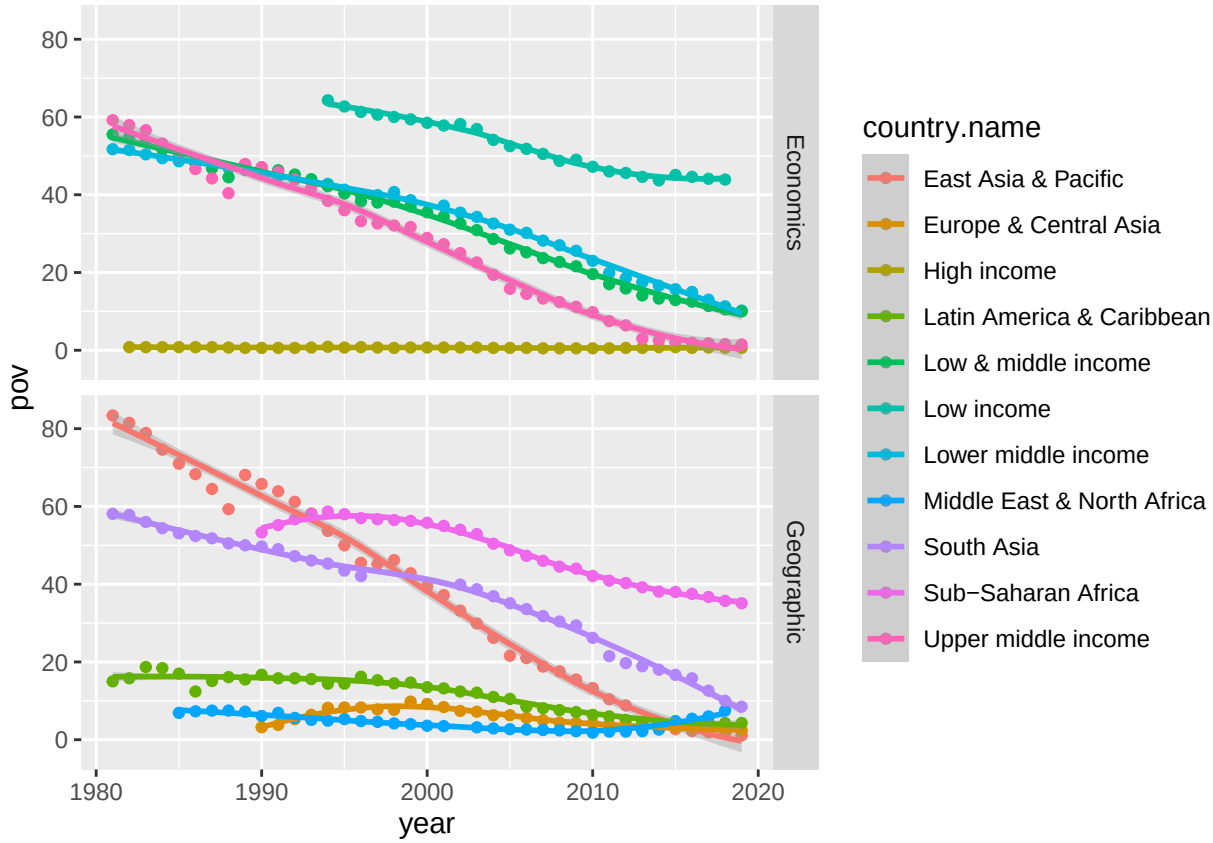
```
## # A tibble: 12 x 3
##    country.code country.name               type
##    <fct>        <fct>                      <chr>
##  1 HIC          High income                Economics
##  2 LIC          Low income                 Economics
##  3 LMC          Lower middle income        Economics
##  4 LMY          Low & middle income        Economics
##  5 UMC          Upper middle income        Economics
##  6 EAS          East Asia & Pacific        Geographic
##  7 ECS          Europe & Central Asia      Geographic
##  8 LCN          Latin America & Caribbean  Geographic
##  9 MEA          Middle East & North Africa Geographic
## 10 SAS          South Asia                 Geographic
## 11 SSF          Sub-Saharan Africa         Geographic
## 12 WLD          World                      Geographic
```

```
# World
ggplot(pov.reg %>%
    filter(country.code == "WLD"), aes(x = year, y = pov)) +
    geom_point() + geom_smooth(formula = y ~ x, method = loess)
```



An overall very steady decrease of poverty. How about each region?

```
ggplot(pov.reg %>%
    filter(country.code != "WLD"), aes(x = year, y = pov,
    color = country.name)) + geom_point() + geom_smooth(formula = y ~
    x, method = loess) + facet_grid(rows = vars(type))
```



There's a general steady, but distinct decline of poverty over time in each type region of respective type. Latin America & Caribbean, Europe & Central Asia, Middle East & North Africa, and High Income group has a more gradual decline as they are not very poor to begin with.

Among the income groups, Low & Middle Income, Lower & Middle Income, and Upper Middle Income have quite similar in term of poverty indicator and slope over the year. While these values vary greatly among different geographical regions.

Let's see some important statistics

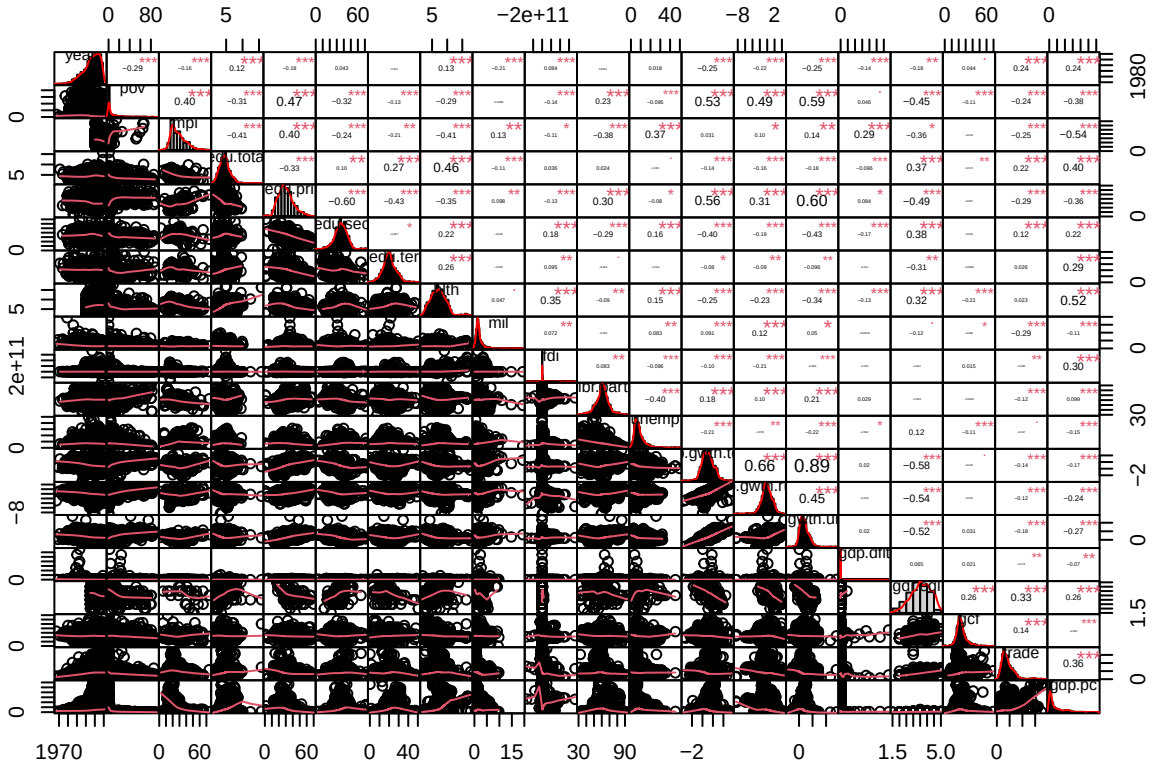
```
stats <- poverty.headcount %>%
    summarise(count = n(), skewness = skewness(pov),
        kurtosis = kurtosis(pov), std.deviation = sd(pov))

kable(stats)
```

count	skewness	kurtosis	std.deviation
2322	1.6	1.66	19.5

2.4. Assumption Check

We can conduct some preliminary checks on linearity and correlation of predictors to have a better picture of the data. Checklist **1. Linear relationship:**
Use correlation matrix to check linearity

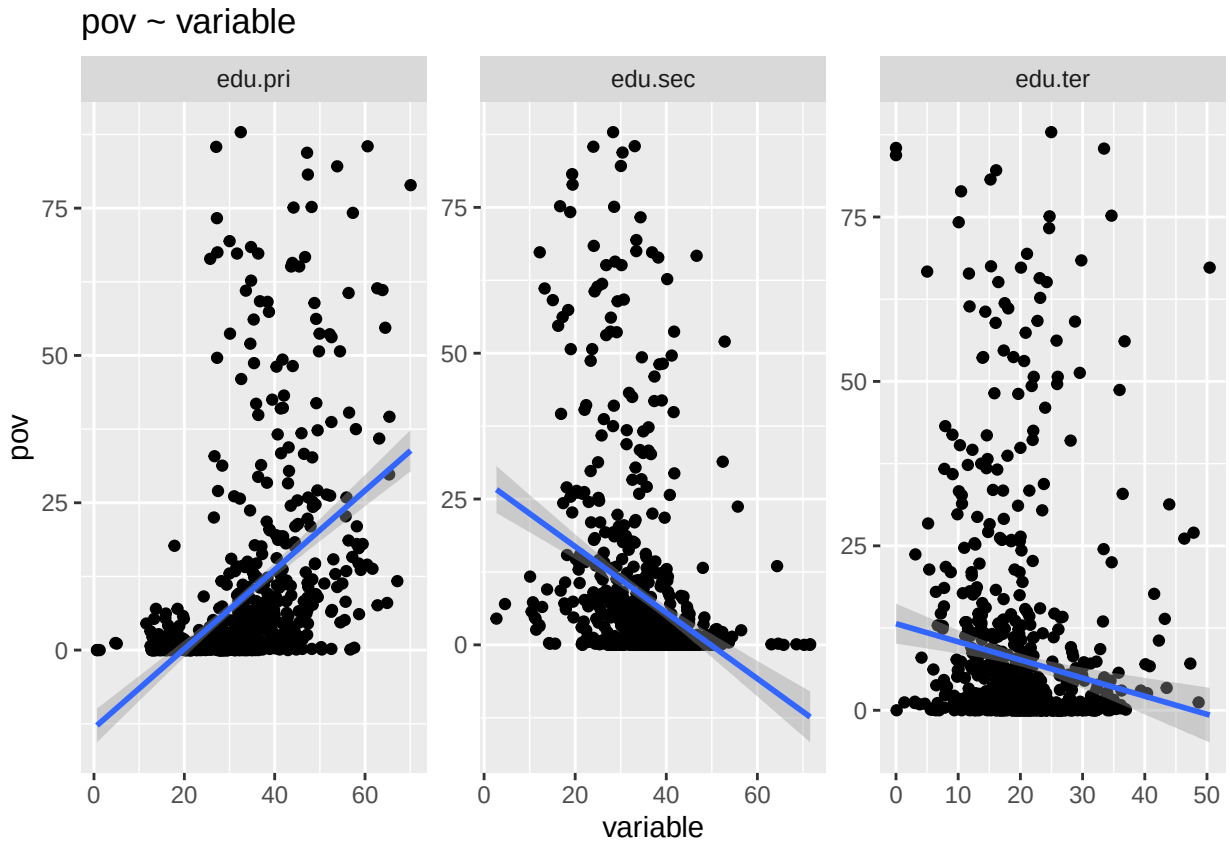
```
# select only numeric data
countries.num <- countries %>%
    select(where(is.numeric))
chart.Correlation(countries.num, histogram = TRUE,
    pch = 19)
```

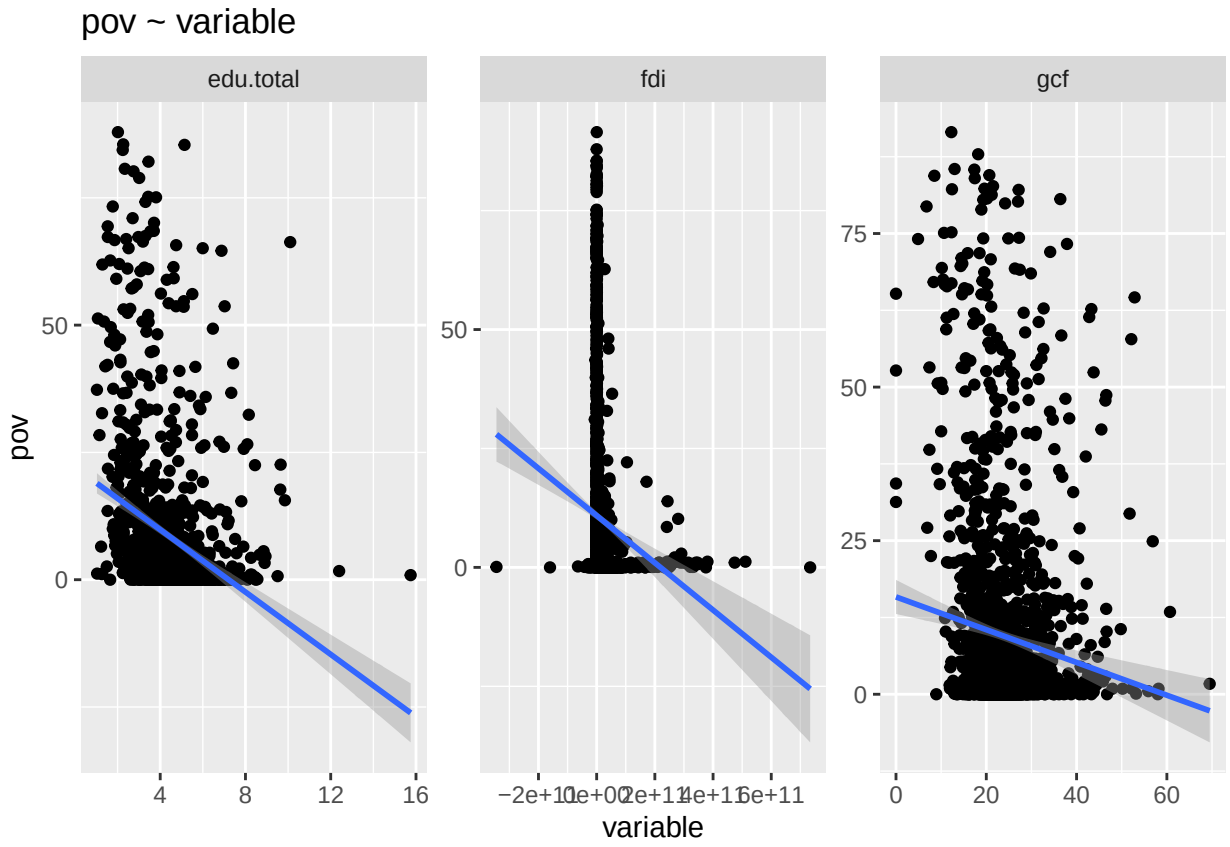


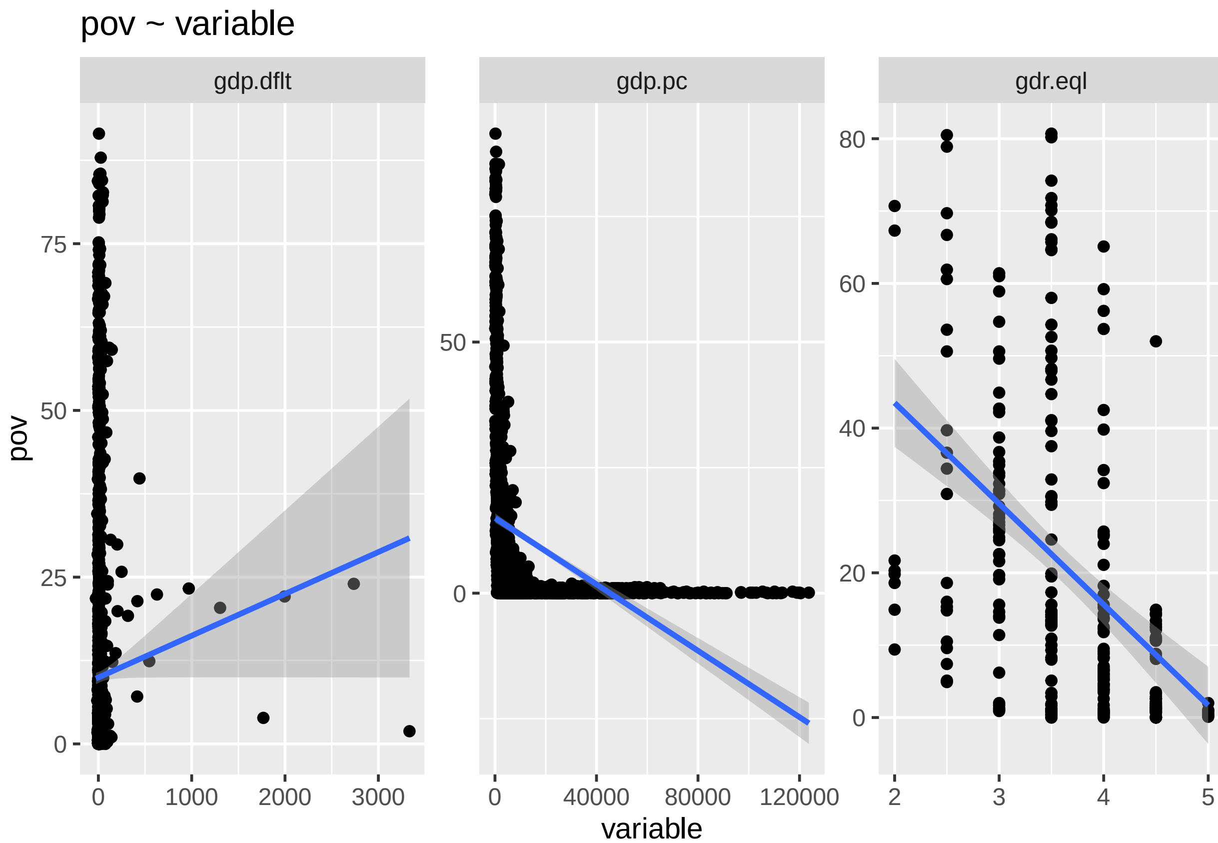
Which is utterly intelligible, but a majority of the fitted lines are linear at first glance. We should render separate graphs for the relationship between `pov` & other variables.

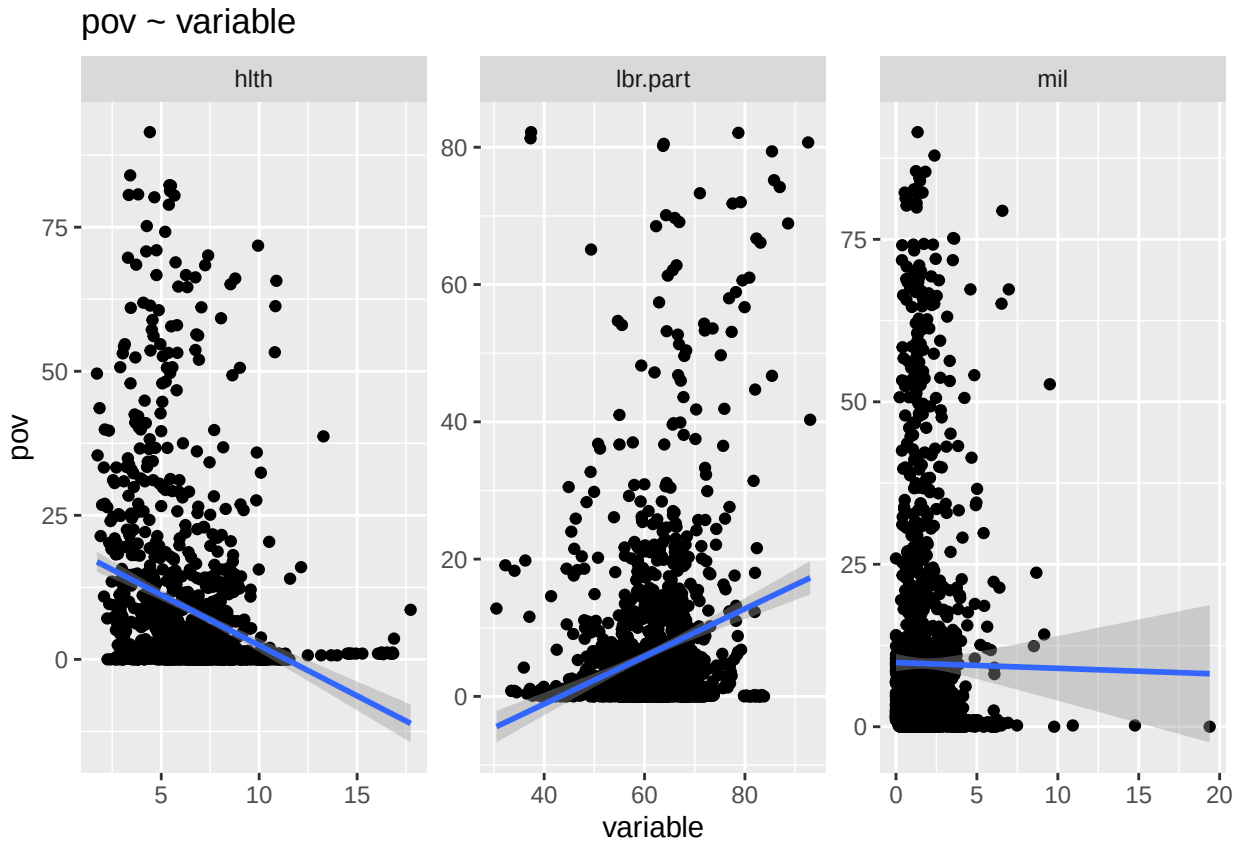
```
# lengthen data table to variable-value pairs,
# with pov as predicted variable and others as
# predictive
countries.num2 <- countries.num %>%
    pivot_longer(cols = 3:ncol(.), names_to = "variable",
        values_to = "value") %>%
```

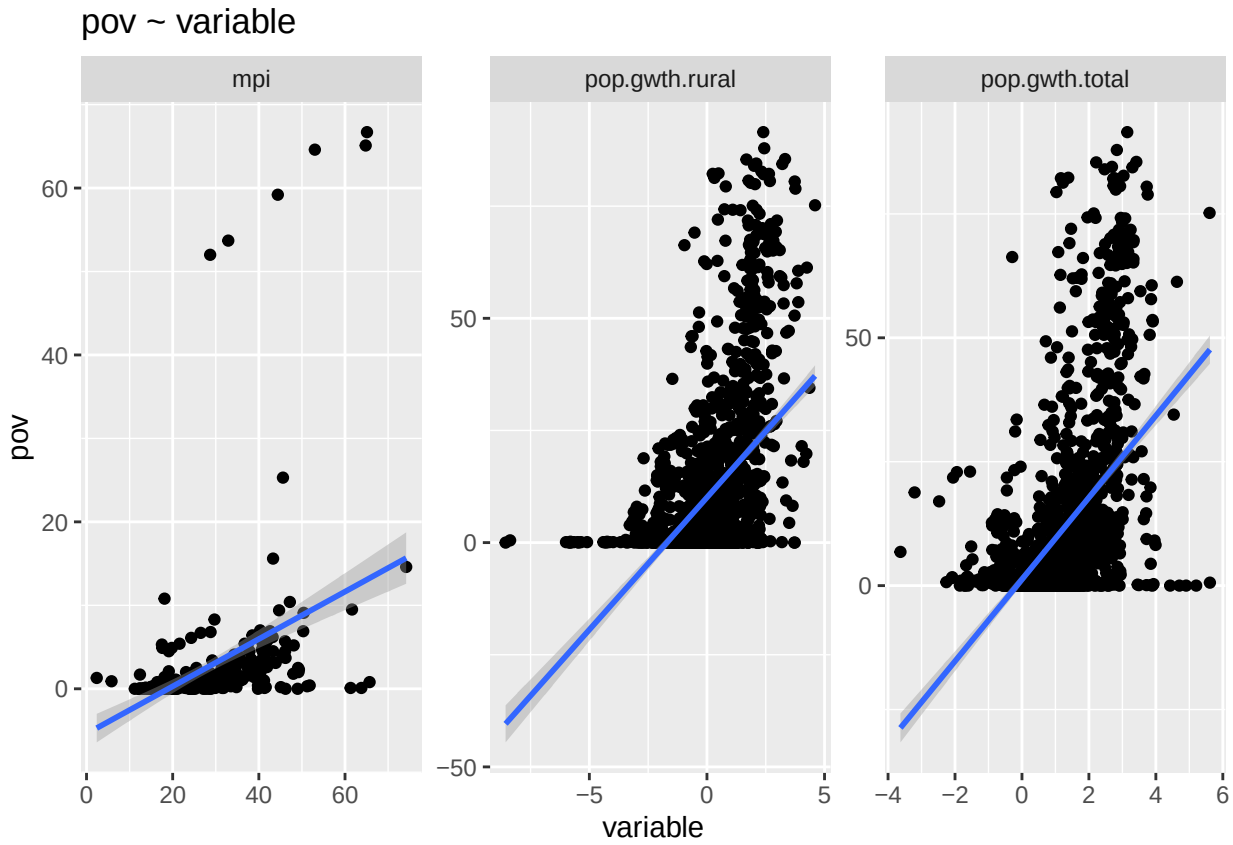
```
    filter(!is.na(value) & !is.na(pov))

drawGraph <- function(indvar, data) {
    ggplot(data %>%
        filter(variable == indvar), aes(x = value,
        y = pov)) + geom_point() + geom_smooth(method = lm) +
        labs(title = paste("Relationship pov ~", indvar),
            x = indvar, y = "pov")
}

ivs <- distinct(countries.num2, variable)[[1]]

# for (indvar in ivs) { print(
# ggplot(countries.num2 %>% filter(variable ==
# indvar), aes(x = value, y = pov)) +
# geom_point() + geom_smooth(formula = y~x,
# method = lm) + labs(title = paste('Relationship
# pov ~', indvar), x = indvar, y = 'pov') ) }

p <- ggplot(countries.num2, aes(x = value, y = pov)) +
    geom_point() + geom_smooth(formula = y ~ x, method = lm) +
    labs(title = paste("pov ~ variable"), x = "variable",
        y = "pov") + facet_wrap_paginate(~variable,
    ncol = 3, nrow = 1, scales = "free")

requiredPages <- n_pages(p)

for (i in 1:requiredPages) {
    ggplot(countries.num2, aes(x = value, y = pov)) +
        geom_point() + geom_smooth(formula = y ~ x,
        method = lm) + labs(title = paste("pov ~ variable"),
        x = "variable", y = "pov") + facet_wrap_paginate(~variable,
        ncol = 3, nrow = 1, scales = "free", page = i) ->
        p
    print(p)
}
```

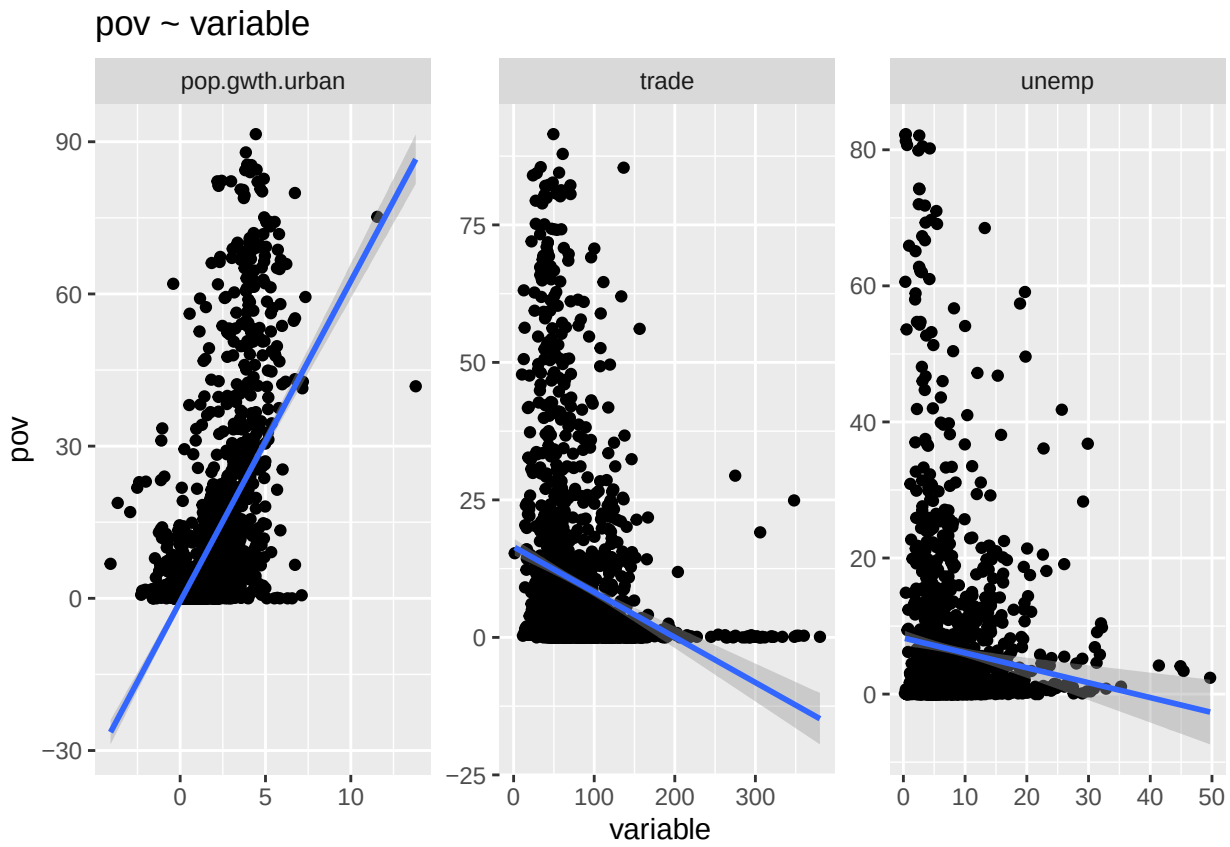

pov ~ variable
edu.pri
edu.sec
edu.ter
pov
variable
0
25
50
75
0
20
40
60
0
10
20
30
40
50


pov ~ variable
edu.total
fdi
gcf
pov
50
0
75
25
4
8
12
16
−2e+11
0e+00
2e+11
4e+11
6e+11
0
20
40
60
variable


pov ~ variable
gdp.dflt
gdp.pc
gdr.eql
pov
variable
75
50
25
0
0
1000
2000
3000
50
0
0
40000
80000
120000
80
60
40
20
0
2
3
4
5

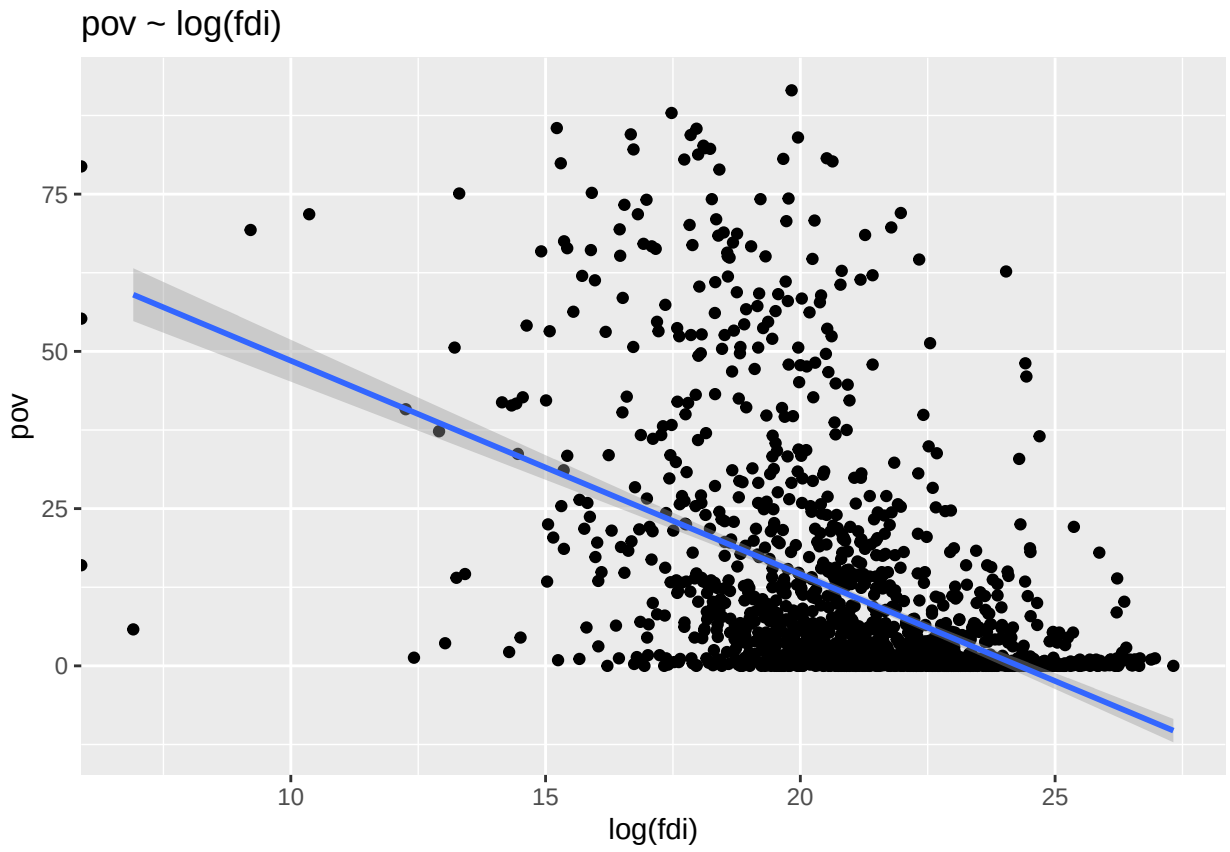

pov ~ variable
hlth
lbr.part
mil
pov
variable
0
25
50
75
0
20
40
60
80
0
5
10
15
20
40
60
80

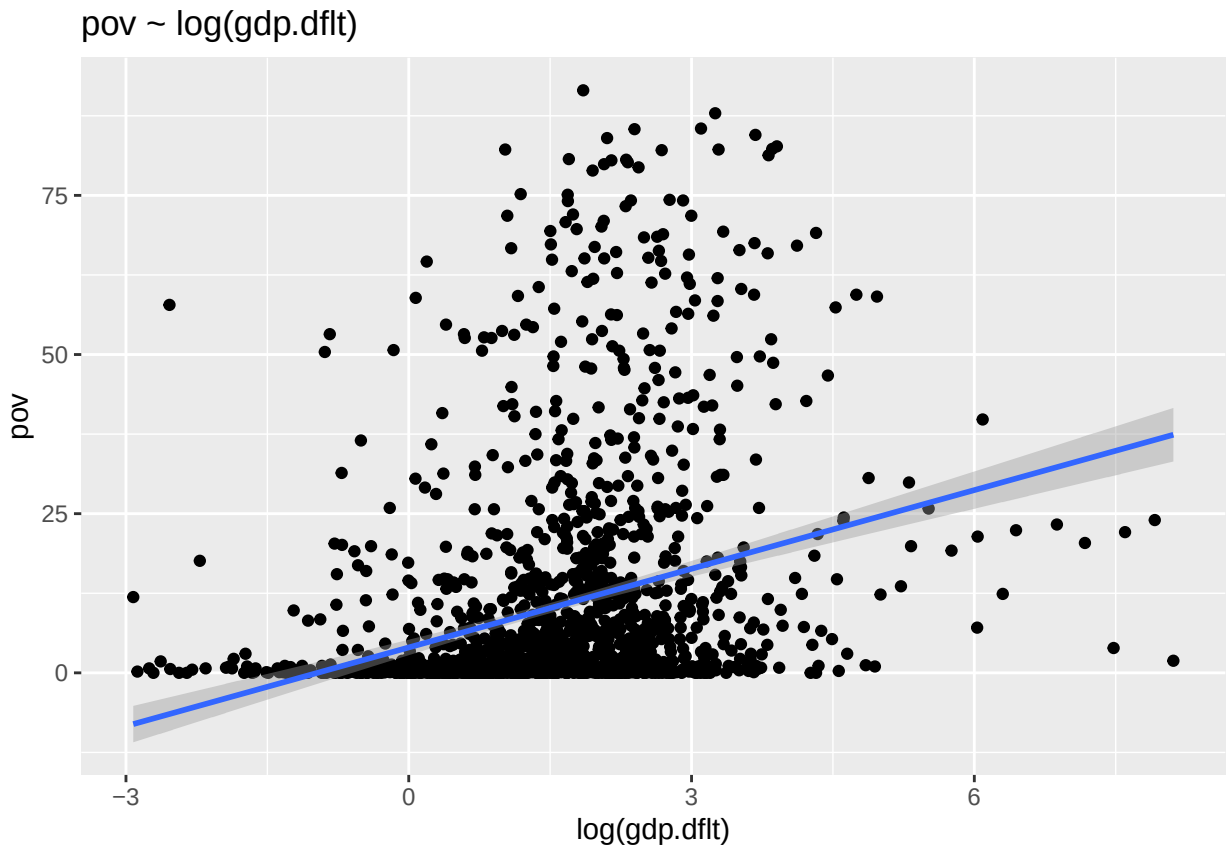

pov ~ variable
mpi
pop.gwth.rural
pop.gwth.total
pov
variable
0
20
40
60
−50
50
−5
5
−4
−2
2
4
6

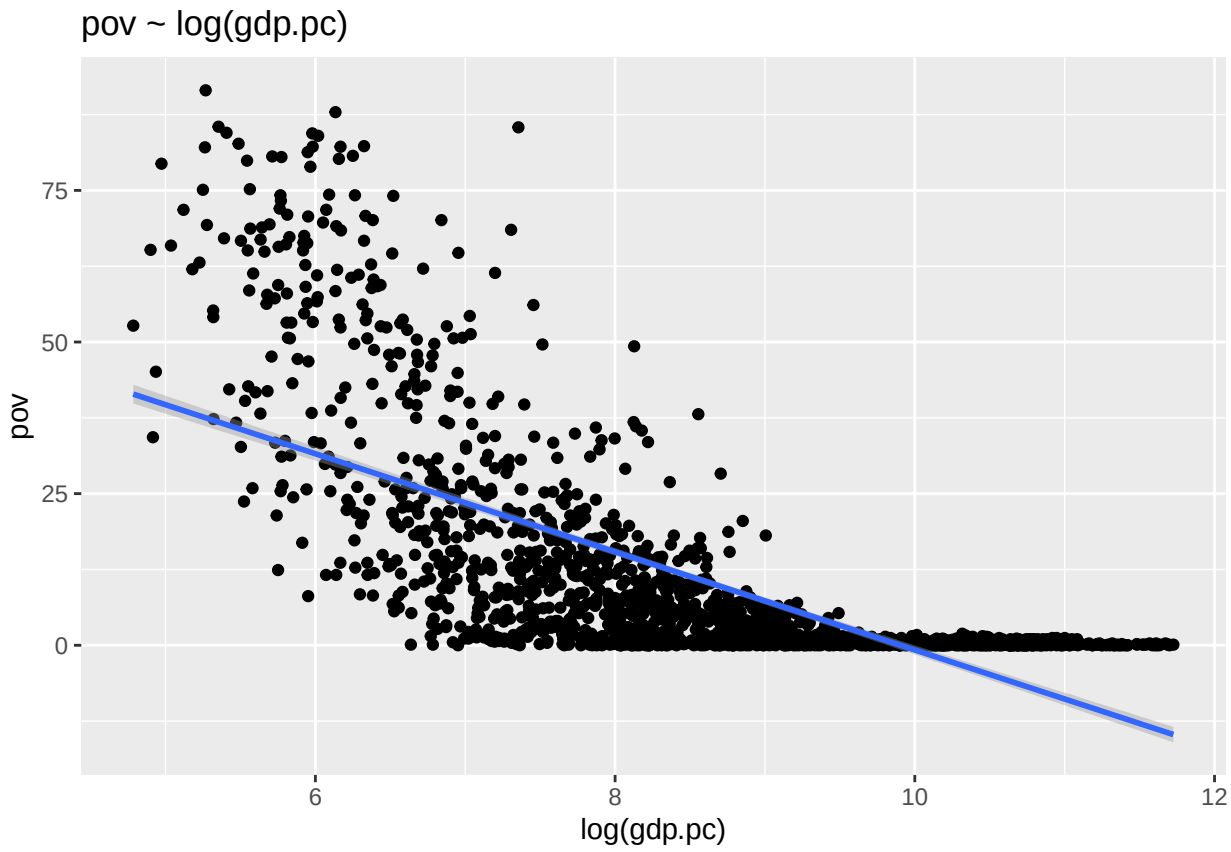


Most variables display linear relationship with some exceptions, which are more appropriate to assume a logarithmic relationship. Gender equality should be viewed as a categorical data.

```
logIvs <- c("fdi", "gdp.dflt", "gdp.pc")

for (indvar in logIvs) {
    print(ggplot(countries.num2 %>%
        filter(variable == indvar), aes(x = log(value),
        y = pov)) + geom_point() + geom_smooth(formula = y ~
        x, method = lm) + labs(title = paste0("pov ~ log(",
        indvar, ")"), x = paste0("log(", indvar, ")"),
        y = "pov"))
}
```

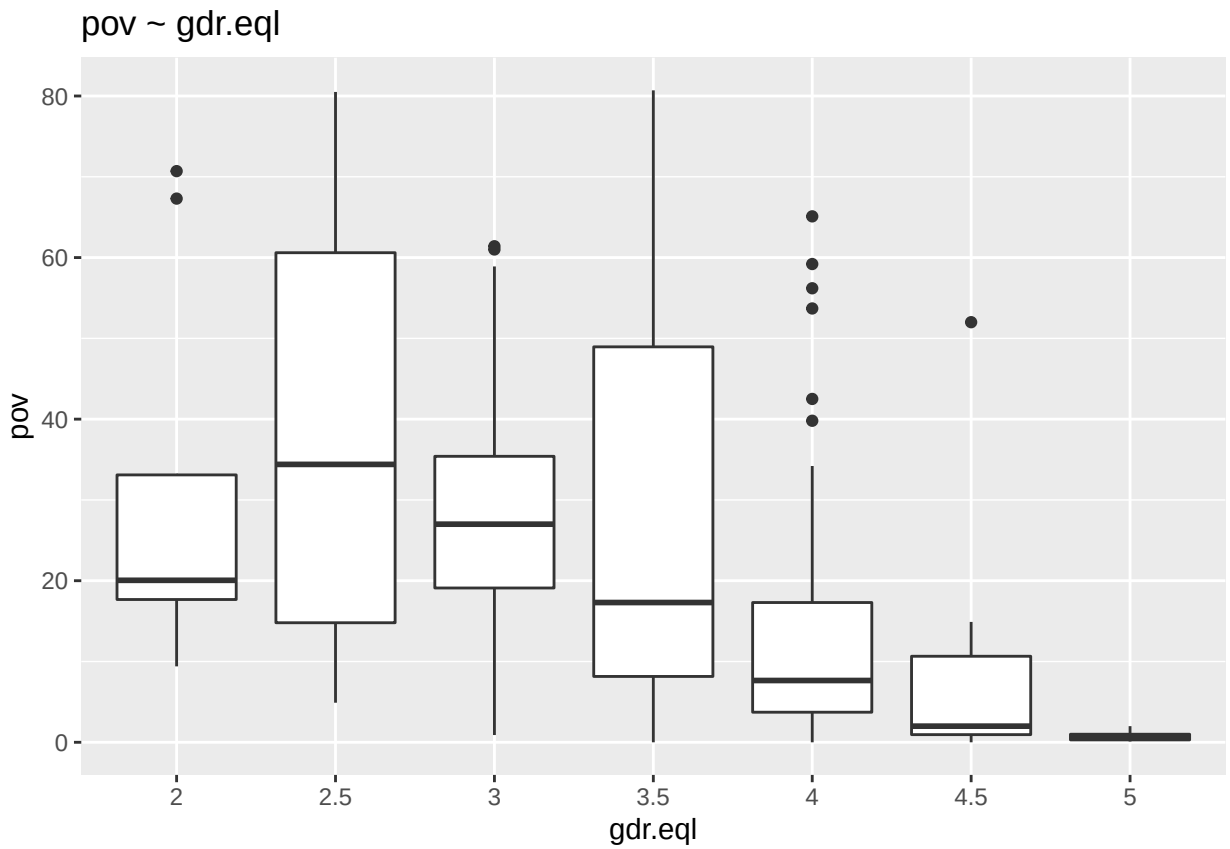

pov ~ log(fdi)
75
50
25
0
pov
10
15
20
25
log(fdi)


pov ~ log(gdp.dflt)
75
50
25
0
pov
−3
0
3
6
log(gdp.dflt)



The relationship are more linear after the transformation.

```
gdr.eql <- countries.num2 %>%
    filter(variable == "gdr.eql") %>%
    mutate(value = factor(value))

ggplot(gdr.eql, aes(x = value, y = pov)) + geom_boxplot() +
    labs(title = paste("pov ~ gdr.eql"), x = "gdr.eql",
        y = "pov")
```



There's a significant correlation between gender equality and poverty.
Correlation coefficients between pov and predictors.

```
# transform some variables
countries.num3 <- countries.num %>%
    mutate(lfdi = log(fdi), lgdp.dflt = log(gdp.dflt),
        lgdp.pc = log(gdp.pc)) %>%
    mutate(lfdi = ifelse(is.nan(lfdi) | lfdi == -Inf,
        NA, lfdi))

# name of the independent variables
cn <- colnames(countries.num3)[-c(1, 2)]

# correlation of independent variable and pov
corWithPov <- function(indevar, data) {
    cor(data[[indevar]], data[["pov"]], use = "complete.obs")
}

cor.pov <- sapply(cn, corWithPov, data = countries.num3)
kable(cor.pov)
```

	x
mpi	0.404
edu.total	-0.310
edu.pri	0.471

	x
edu.sec	-0.317
edu.ter	-0.128
hlth	-0.294
mil	-0.007
fdi	-0.144
lbr.part	0.232
unemp	-0.095
pop.gwth.total	0.529
pop.gwth.rural	0.494
pop.gwth.urban	0.587
gdp.dflt	0.046
gdr.eql	-0.451
gcf	-0.111
trade	-0.239
gdp.pc	-0.377
lfdi	-0.488
lgdp.dflt	0.299
lgdp.pc	-0.705

`edu.ter`, `mil`, `fdi`, `lbr.part`, `unemp`, `gdp.dflt`, `gcf` have negligible correlation with `pov`. `lgdp.pc`, `lfdi`, and `lgdp.dflt` have stronger linear relationship with `pov` than their un-transformed counterparts.

```
countries1 <- countries1 %>%
    mutate(lfdi = log(fdi), lgdp.dflt = log(gdp.dflt),
        lgdp.pc = log(gdp.pc)) %>%
    mutate(lfdi = ifelse(is.nan(lfdi) | lfdi == -Inf,
        NA, lfdi))
```

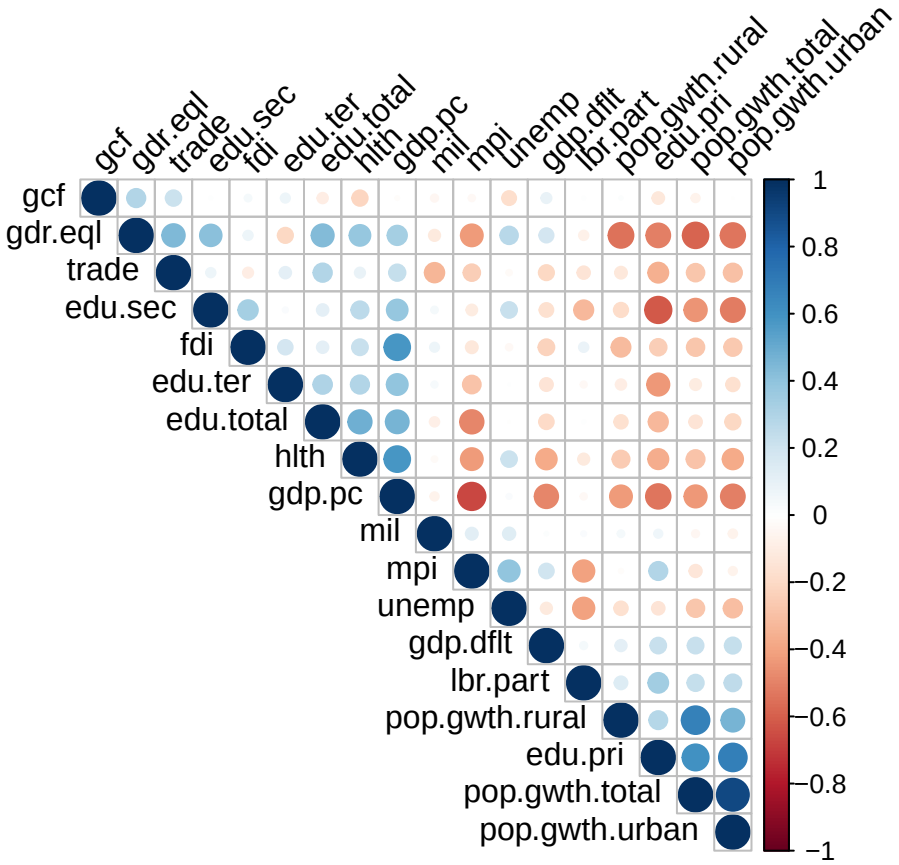
2. Predictors are independent There should be no correlation/multicolinearity between each pair of predictors.

```
countries.arr <- simplify2array(countries.num %>%
    select(-c("year", "pov")))

cor.mtx <- rcorr(countries.arr, type = "spearman")

round(cor.mtx$r, 2)
```

```
##                  mpi edu.total edu.pri edu.sec edu.ter  hlth   mil   fdi
## mpi             1.00     -0.49    0.29   -0.10   -0.28 -0.43  0.13 -0.13
## edu.total      -0.49      1.00   -0.33    0.12    0.31  0.48 -0.08  0.12
## edu.pri         0.29     -0.33    1.00   -0.62   -0.44 -0.36  0.06 -0.24
## edu.sec        -0.10      0.12   -0.62    1.00    0.02  0.26  0.04  0.34
## edu.ter        -0.28      0.31   -0.44    0.02    1.00  0.30  0.04  0.19
## hlth           -0.43      0.48   -0.36    0.26    0.30  1.00 -0.03  0.22
## mil             0.13     -0.08    0.06    0.04    0.04 -0.03  1.00  0.07
## fdi            -0.13      0.12   -0.24    0.34    0.19  0.22  0.07  1.00
## lbr.part       -0.41      0.01    0.35   -0.33   -0.04 -0.11  0.02  0.08
## unemp           0.40      0.00   -0.14    0.22    0.00  0.22  0.14 -0.04
## pop.gwth.total -0.14     -0.14    0.60   -0.45   -0.11 -0.29 -0.05 -0.27
## pop.gwth.rural -0.02     -0.16    0.29   -0.18   -0.10 -0.26  0.04 -0.32
```

```
## pop.gwth.urban -0.06     -0.21    0.68   -0.51   -0.16 -0.37 -0.07 -0.27
## gdp.dflt        0.19     -0.19    0.22   -0.17   -0.15 -0.38  0.01 -0.22
## gdr.eql        -0.43      0.44   -0.50    0.41   -0.21  0.39 -0.11  0.08
## gcf            -0.04     -0.10   -0.12    0.01    0.08 -0.21 -0.05  0.04
## trade          -0.24      0.30   -0.35    0.08    0.12  0.09 -0.34 -0.09
## gdp.pc         -0.66      0.47   -0.53    0.38    0.39  0.59 -0.06  0.59
##                lbr.part unemp pop.gwth.total pop.gwth.rural pop.gwth.urban
## mpi               -0.41  0.40          -0.14          -0.02          -0.06
## edu.total          0.01  0.00          -0.14          -0.16          -0.21
## edu.pri            0.35 -0.14           0.60           0.29           0.68
## edu.sec           -0.33  0.22          -0.45          -0.18          -0.51
## edu.ter           -0.04  0.00          -0.11          -0.10          -0.16
## hlth              -0.11  0.22          -0.29          -0.26          -0.37
## mil                0.02  0.14          -0.05           0.04          -0.07
## fdi                0.08 -0.04          -0.27          -0.32          -0.27
## lbr.part           1.00 -0.41           0.24           0.15           0.25
## unemp             -0.41  1.00          -0.27          -0.16          -0.30
## pop.gwth.total     0.24 -0.27           1.00           0.68           0.91
## pop.gwth.rural     0.15 -0.16           0.68           1.00           0.46
## pop.gwth.urban     0.25 -0.30           0.91           0.46           1.00
## gdp.dflt           0.05 -0.11           0.22           0.11           0.23
## gdr.eql           -0.08  0.27          -0.58          -0.55          -0.53
## gcf                0.01 -0.17          -0.06           0.01           0.00
## trade             -0.14 -0.03          -0.28          -0.13          -0.29
## gdp.pc            -0.04  0.02          -0.44          -0.43          -0.50
##                gdp.dflt gdr.eql   gcf trade gdp.pc
## mpi                0.19   -0.43 -0.04 -0.24  -0.66
## edu.total         -0.19    0.44 -0.10  0.30   0.47
## edu.pri            0.22   -0.50 -0.12 -0.35  -0.53
## edu.sec           -0.17    0.41  0.01  0.08   0.38
## edu.ter           -0.15   -0.21  0.08  0.12   0.39
## hlth              -0.38    0.39 -0.21  0.09   0.59
## mil                0.01   -0.11 -0.05 -0.34  -0.06
## fdi               -0.22    0.08  0.04 -0.09   0.59
## lbr.part           0.05   -0.08  0.01 -0.14  -0.04
## unemp             -0.11    0.27 -0.17 -0.03   0.02
## pop.gwth.total     0.22   -0.58 -0.06 -0.28  -0.44
## pop.gwth.rural     0.11   -0.55  0.01 -0.13  -0.43
## pop.gwth.urban     0.23   -0.53  0.00 -0.29  -0.50
## gdp.dflt           1.00    0.18  0.09 -0.20  -0.49
## gdr.eql            0.18    1.00  0.30  0.45   0.33
## gcf                0.09    0.30  1.00  0.21  -0.01
## trade             -0.20    0.45  0.21  1.00   0.24
## gdp.pc            -0.49    0.33 -0.01  0.24   1.00
```

```
corrplot(cor.mtx$r, type = "upper", order = "hclust",
    tl.col = "black", tl.srt = 45)
```



Some significant correlations can be found between `gdr.eql` with `edu.pri`, `pop.gwth.total`, and `fdi` with `mil`, etc. We expect these variables to be eliminated in the VIF test. This might be a good property for imputation (3.2.4)

2.5. Data Source

- poverty.headcount
- mpi
- education.expenditure.primary
- education.expenditure.secondary
- education.expenditure.tertiary
- education.expenditure.total
- health.expenditure
- military.expenditure
- fdi
- unemployment.rate
- labour.force.participation
- gender.equality
- population.growth
- urban.population.growth
- rural.population.growth
- gdp.deflator
- gross capital formation
- trade
- region.class
- income.class

- gross.capital.formation

3. Model Selection and Interpretation

We can now conduct linear regressions following the approaches mentioned above to address missing values issues.

3.1. Use Complete Cases

```
needs("tidyverse", "ggpubr", "gridExtra", "e1071",
    "ggplot2", "dplyr", "tidyr", "readxl", "corrplot",
    "car", "DescTools", "glmnet")
```

First we will write the function "eval_results(true,predict) to evaluate the models

```
eval_results <- function(true, predict) {
    actual <- data.matrix(true)
    SSE <- sum((predict - actual)^2)
    SST <- sum((actual - mean(actual))^2)
    R_square <- 1 - SSE/SST
    data.frame(MSE = MSE(predict, true), MAE = MAE(predict,
        true), RMSE = RMSE(predict, true), Rsquare = R_square)
}
```

First we read the csv file for the countries containing the dataset. We will then put the variables we need in to subset1.

```
df <- read.csv("../data/countries1.csv")
head(df)
```

```
##   X country.code country.name year  pov income                  reg edu.total
## 1 1          AGO       Angola 2000 21.4      L    Sub-Saharan Africa      2.61
## 2 2          AGO       Angola 2008 14.6     LM    Sub-Saharan Africa      2.69
## 3 3          AGO       Angola 2018 31.1     LM    Sub-Saharan Africa      2.04
## 4 4          ALB      Albania 1996  0.5     LM Europe & Central Asia      3.08
## 5 5          ALB      Albania 2002  1.1     LM Europe & Central Asia      3.12
## 6 6          ALB      Albania 2005  0.6     LM Europe & Central Asia      3.28
##   hlth  mil       fdi lbr.part unemp pop.gwth.total pop.gwth.rural
## 1 1.91 6.39  8.79e+08       NA    NA          3.277          0.921
## 2 3.32 3.57  1.68e+09       NA    NA          3.711          1.910
## 3 2.54 1.87 -6.46e+09       NA    NA          3.276          1.338
## 4   NA 1.38  9.01e+07     38.8  12.3         -0.622         -1.546
## 5 6.91 1.32  1.35e+08     59.6  15.8         -0.300         -2.169
## 6 6.34 1.35  2.62e+08     34.5  14.1         -0.512         -2.519
##   pop.gwth.urban gdp.dflt  gcf trade gdp.pc lfdi lgdp.dflt lgdp.pc
## 1          5.682   418.02 30.5 152.5    557 20.6      6.04    6.32
## 2          5.020    19.37 30.8 121.4   4081 21.2      2.96    8.31
## 3          4.312    28.17 17.9  66.4   2525   NA      3.34    7.83
## 4          0.812    38.17 18.1  44.9   1010 18.3      3.64    6.92
## 5          2.181     3.65 35.3  68.5   1425 18.7      1.29    7.26
## 6          1.826     3.31 36.9  70.9   2674 19.4      1.20    7.89
```

```
subset1 <- df %>%
    select("pov", "country.code", "year", "income",
        "edu.total", "hlth", "mil", "fdi", "lbr.part",
        "unemp", "pop.gwth.total", "pop.gwth.rural",
        "gdp.dflt", "gcf", "trade", "gdp.pc", "lfdi",
        "lgdp.dflt", "lgdp.pc")
head(subset1)
```

```
##    pov country.code year income edu.total hlth  mil       fdi lbr.part unemp
## 1 21.4          AGO 2000      L      2.61 1.91 6.39  8.79e+08       NA    NA
## 2 14.6          AGO 2008     LM      2.69 3.32 3.57  1.68e+09       NA    NA
## 3 31.1          AGO 2018     LM      2.04 2.54 1.87 -6.46e+09       NA    NA
## 4  0.5          ALB 1996     LM      3.08   NA 1.38  9.01e+07     38.8  12.3
## 5  1.1          ALB 2002     LM      3.12 6.91 1.32  1.35e+08     59.6  15.8
## 6  0.6          ALB 2005     LM      3.28 6.34 1.35  2.62e+08     34.5  14.1
##   pop.gwth.total pop.gwth.rural gdp.dflt  gcf trade gdp.pc lfdi lgdp.dflt
## 1          3.277          0.921   418.02 30.5 152.5    557 20.6      6.04
## 2          3.711          1.910    19.37 30.8 121.4   4081 21.2      2.96
## 3          3.276          1.338    28.17 17.9  66.4   2525   NA      3.34
## 4         -0.622         -1.546    38.17 18.1  44.9   1010 18.3      3.64
## 5         -0.300         -2.169     3.65 35.3  68.5   1425 18.7      1.29
## 6         -0.512         -2.519     3.31 36.9  70.9   2674 19.4      1.20
##   lgdp.pc
## 1    6.32
## 2    8.31
## 3    7.83
## 4    6.92
## 5    7.26
## 6    7.89
```

Next we will build a SLR model. We will put 80% of dataset to train and the rest to test.

```
set.seed(10)
index <- sort(sample(x = nrow(subset1), size = nrow(subset1) *
    0.8))
train <- df[index, ]
test <- df[-index, ]
```

The model output will be placed in lm.slr and can be viewed in summary(lm.slr).

```
lm.slr <- lm(pov ~ country.code + year + income + edu.total +
    hlth + mil + fdi + lbr.part + unemp + pop.gwth.total +
    pop.gwth.rural + gdp.dflt + gcf + trade + gdp.pc +
    lfdi + lgdp.dflt + lgdp.pc, data = train)
summary(lm.slr)
```

```
##
## Call:
## lm(formula = pov ~ country.code + year + income + edu.total +
##     hlth + mil + fdi + lbr.part + unemp + pop.gwth.total + pop.gwth.rural +
##     gdp.dflt + gcf + trade + gdp.pc + lfdi + lgdp.dflt + lgdp.pc,
##     data = train)
```

```
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.820 -0.731  0.000  0.754 14.036
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.01e+02   6.16e+01    6.51  1.7e-10 ***
## country.codeARG  2.20e+00   1.42e+00    1.56  0.12022
## country.codeARM -7.48e-01   1.48e+00   -0.50  0.61434
## country.codeAUS  1.26e+00   2.09e+00    0.60  0.54581
## country.codeAUT  2.12e+00   1.94e+00    1.09  0.27557
## country.codeAZE -8.72e+00   2.52e+00   -3.46  0.00059 ***
## country.codeBEL  4.25e+00   2.07e+00    2.06  0.04025 *
## country.codeBEN  1.36e+01   2.65e+00    5.13  3.9e-07 ***
## country.codeBFA  2.26e+01   2.76e+00    8.20  1.7e-15 ***
## country.codeBGD  1.27e+01   2.05e+00    6.21  1.0e-09 ***
## country.codeBGR  4.15e+00   1.40e+00    2.96  0.00316 **
## country.codeBLR  3.13e+00   1.39e+00    2.24  0.02521 *
## country.codeBOL  3.63e+00   1.87e+00    1.94  0.05334 .
## country.codeBRA  7.23e+00   1.46e+00    4.95  9.8e-07 ***
## country.codeBWA  2.08e+01   2.49e+00    8.33  6.2e-16 ***
## country.codeCAN  1.33e+00   2.66e+00    0.50  0.61716
## country.codeCHE -5.63e-01   2.36e+00   -0.24  0.81180
## country.codeCHL  3.37e+00   1.49e+00    2.27  0.02383 *
## country.codeCHN  1.72e+01   1.61e+00   10.71  < 2e-16 ***
## country.codeCMR  2.11e+01   2.10e+00   10.08  < 2e-16 ***
## country.codeCOG  4.29e+01   2.53e+00   16.97  < 2e-16 ***
## country.codeCOL  7.15e+00   1.37e+00    5.22  2.5e-07 ***
## country.codeCPV  4.42e+00   2.44e+00    1.81  0.07127 .
## country.codeCRI  5.33e+00   1.36e+00    3.93  9.5e-05 ***
## country.codeCYP  1.90e+00   1.91e+00    0.99  0.32106
## country.codeCZE  4.23e+00   1.54e+00    2.75  0.00609 **
## country.codeDEU  1.58e+00   1.91e+00    0.83  0.40825
## country.codeDNK  7.38e-01   2.10e+00    0.35  0.72561
## country.codeDOM  4.49e+00   1.30e+00    3.46  0.00058 ***
## country.codeECU  5.42e+00   1.47e+00    3.68  0.00026 ***
## country.codeEGY -2.46e+00   1.79e+00   -1.37  0.17110
## country.codeESP  2.89e+00   1.53e+00    1.89  0.05953 .
## country.codeEST  4.63e+00   1.58e+00    2.94  0.00347 **
## country.codeFIN  1.61e+00   1.77e+00    0.91  0.36375
## country.codeFJI  1.48e+00   2.38e+00    0.62  0.53393
## country.codeFRA  3.63e+00   2.11e+00    1.72  0.08583 .
## country.codeGBR -2.28e-02   1.82e+00   -0.01  0.99004
## country.codeGEO  7.44e+00   1.26e+00    5.88  7.1e-09 ***
## country.codeGIN  3.39e+01   2.60e+00   13.08  < 2e-16 ***
## country.codeGNB  1.15e+01   2.63e+00    4.37  1.5e-05 ***
## country.codeGRC  1.06e+00   1.71e+00    0.62  0.53441
## country.codeGTM  7.33e+00   1.97e+00    3.71  0.00023 ***
## country.codeHND  1.30e+01   1.50e+00    8.68  < 2e-16 ***
## country.codeHRV  2.58e+00   1.60e+00    1.61  0.10697
## country.codeHUN  4.70e+00   1.59e+00    2.95  0.00326 **
## country.codeIDN  1.32e+01   1.39e+00    9.52  < 2e-16 ***
## country.codeIND  9.62e+00   2.09e+00    4.60  5.1e-06 ***
```

```
## country.codeIRL  2.89e+00   2.06e+00    1.40  0.16113
## country.codeIRN  3.53e+00   1.41e+00    2.50  0.01260 *
## country.codeISR  1.71e-01   1.84e+00    0.09  0.92622
## country.codeITA  2.16e+00   1.70e+00    1.27  0.20457
## country.codeJAM -6.86e-01   1.85e+00   -0.37  0.71140
## country.codeJOR  1.82e+00   2.73e+00    0.67  0.50546
## country.codeKAZ  2.25e+00   1.39e+00    1.62  0.10608
## country.codeKEN  2.27e+01   2.71e+00    8.39  3.9e-16 ***
## country.codeKGZ -1.21e+00   1.60e+00   -0.75  0.45131
## country.codeKOR  3.79e+00   2.46e+00    1.54  0.12481
## country.codeLKA  6.20e-01   1.86e+00    0.33  0.73848
## country.codeLTU  3.68e+00   1.54e+00    2.39  0.01723 *
## country.codeLUX -4.30e+00   3.39e+00   -1.27  0.20457
## country.codeLVA  4.20e+00   1.52e+00    2.76  0.00605 **
## country.codeMDA  1.19e+00   1.59e+00    0.75  0.45279
## country.codeMDG  6.74e+01   2.34e+00   28.80  < 2e-16 ***
## country.codeMEX  7.52e+00   1.50e+00    5.03  6.7e-07 ***
## country.codeMLI  5.55e+00   2.73e+00    2.03  0.04285 *
## country.codeMLT  7.22e+00   2.39e+00    3.02  0.00264 **
## country.codeMMR -1.13e+00   2.07e+00   -0.55  0.58407
## country.codeMNG  4.05e+00   1.51e+00    2.68  0.00752 **
## country.codeMUS  3.29e+00   1.92e+00    1.71  0.08694 .
## country.codeMYS  4.43e+00   1.80e+00    2.46  0.01428 *
## country.codeNER  4.94e+01   2.90e+00   17.07  < 2e-16 ***
## country.codeNLD  3.41e+00   2.23e+00    1.53  0.12705
## country.codeNOR -3.28e+00   2.29e+00   -1.43  0.15214
## country.codePAK  7.37e-01   2.01e+00    0.37  0.71368
## country.codePAN  1.03e+01   1.36e+00    7.55  1.8e-13 ***
## country.codePER  7.79e+00   1.41e+00    5.53  5.0e-08 ***
## country.codePHL  4.56e+00   1.68e+00    2.71  0.00688 **
## country.codePOL  1.99e+00   1.45e+00    1.37  0.17007
## country.codePRT  2.21e+00   1.53e+00    1.44  0.15018
## country.codePRY  2.35e+00   1.36e+00    1.73  0.08457 .
## country.codeROU  7.13e+00   1.34e+00    5.33  1.4e-07 ***
## country.codeRUS -2.66e+00   1.51e+00   -1.76  0.07864 .
## country.codeSDN  1.14e+01   2.52e+00    4.52  7.7e-06 ***
## country.codeSLE  1.55e+01   2.77e+00    5.61  3.2e-08 ***
## country.codeSLV  3.39e+00   1.30e+00    2.61  0.00942 **
## country.codeSRB  6.32e+00   1.43e+00    4.41  1.3e-05 ***
## country.codeSVK  3.65e+00   1.65e+00    2.21  0.02776 *
## country.codeSVN  3.79e+00   1.64e+00    2.31  0.02150 *
## country.codeSWE  1.46e+00   1.89e+00    0.77  0.44099
## country.codeSWZ  3.27e+01   2.40e+00   13.63  < 2e-16 ***
## country.codeSYC  6.39e+00   2.53e+00    2.52  0.01203 *
## country.codeTCD  2.06e+01   2.73e+00    7.54  2.0e-13 ***
## country.codeTGO  4.79e+01   2.27e+00   21.11  < 2e-16 ***
## country.codeTHA  1.76e+00   1.42e+00    1.24  0.21590
## country.codeTJK  5.66e+00   1.99e+00    2.84  0.00472 **
## country.codeTUN  6.59e-01   1.89e+00    0.35  0.72738
## country.codeTUR  1.38e+00   1.56e+00    0.88  0.37855
## country.codeTZA  3.61e+01   2.73e+00   13.22  < 2e-16 ***
## country.codeUGA  3.00e+01   2.44e+00   12.32  < 2e-16 ***
## country.codeUKR -3.45e+00   1.50e+00   -2.30  0.02173 *
## country.codeURY  1.52e+00   1.45e+00    1.05  0.29434
```

```
## country.codeUSA  1.19e+00   2.73e+00    0.44  0.66154
## country.codeVEN  6.94e+00   2.37e+00    2.93  0.00351 **
## country.codeZAF  2.36e+01   1.53e+00   15.44  < 2e-16 ***
## year            -1.76e-01   3.17e-02   -5.54  4.8e-08 ***
## incomeL          6.10e-01   1.14e+00    0.54  0.59118
## incomeLM        -1.40e+00   6.94e-01   -2.02  0.04396 *
## incomeUM        -1.58e+00   4.62e-01   -3.42  0.00066 ***
## edu.total       -3.10e-01   1.68e-01   -1.84  0.06565 .
## hlth            -1.62e-01   1.41e-01   -1.16  0.24825
## mil              4.52e-01   1.83e-01    2.47  0.01390 *
## fdi              2.37e-13   3.09e-12    0.08  0.93884
## lbr.part         5.25e-02   3.18e-02    1.65  0.09893 .
## unemp            4.81e-02   4.20e-02    1.15  0.25211
## pop.gwth.total  -5.72e-01   3.80e-01   -1.50  0.13310
## pop.gwth.rural   3.09e-01   2.49e-01    1.24  0.21554
## gdp.dflt         3.32e-02   2.19e-02    1.52  0.12921
## gcf             -2.06e-01   2.84e-02   -7.23  1.6e-12 ***
## trade           -1.62e-02   7.30e-03   -2.22  0.02691 *
## gdp.pc           2.14e-04   2.23e-05    9.61  < 2e-16 ***
## lfdi            -1.27e-01   1.24e-01   -1.02  0.30693
## lgdp.dflt       -4.52e-02   1.49e-01   -0.30  0.76114
## lgdp.pc         -4.74e+00   4.94e-01   -9.59  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.97 on 555 degrees of freedom
##   (796 observations deleted due to missingness)
## Multiple R-squared:  0.962,  Adjusted R-squared:  0.953
## F-statistic:  115 on 122 and 555 DF,  p-value: <2e-16
```

Next we will be performing standardisation.However not all the variables are numerical. Therefore to overcome it we build the function below. This will allow us to perform standardisation to the dataset.

```
subset1$income <- factor(subset1$income)
subset1$country.code <- factor(subset1$country.code)



sd0 <- function(vct) {
    if (!is.numeric(vct)) {
        return(NA)
    }

    return(sd(vct, na.rm = T))
}


std0 <- function(vct, scl) {
    if (!is.numeric(vct)) {
        return(vct)
    }

    return(vct/scl)
}
```

```
scale <- function(data) {
    unlist(lapply(data, sd0))
}

standardise <- function(data) {
    scaler <- scale(data)

    numCols <- which(unlist(lapply(data, is.numeric)))
    num <- as.data.frame(mapply(std0, vct = data[,
        numCols], scl = scaler[numCols], SIMPLIFY = T))
    fct <- data[, -numCols]
    return(cbind(fct, num))
}
```

Next we will perform standardistation to the dataset.

```
str(subset1)
```

```
## 'data.frame':    1843 obs. of  19 variables:
##  $ pov           : num  21.4 14.6 31.1 0.5 1.1 0.6 0.2 0.6 1 0.1 ...
##  $ country.code  : Factor w/ 168 levels "AGO","ALB","ARE",..: 1 1 1 2 2 2 2 2 2 2 ...
##  $ year          : int  2000 2008 2018 1996 2002 2005 2008 2012 2014 2015 ...
##  $ income        : Factor w/ 4 levels "H","L","LM","UM": 2 3 3 3 3 3 3 4 4 4 ...
##  $ edu.total     : num  2.61 2.69 2.04 3.08 3.12 ...
##  $ hlth          : num  1.91 3.32 2.54 NA 6.91 ...
##  $ mil           : num  6.39 3.57 1.87 1.38 1.32 ...
##  $ fdi           : num  8.79e+08 1.68e+09 -6.46e+09 9.01e+07 1.35e+08 ...
##  $ lbr.part      : num  NA NA NA 38.8 59.6 ...
##  $ unemp         : num  NA NA NA 12.3 15.8 ...
##  $ pop.gwth.total: num  3.277 3.711 3.276 -0.622 -0.3 ...
##  $ pop.gwth.rural: num  0.921 1.91 1.338 -1.546 -2.169 ...
##  $ gdp.dflt      : num  418.02 19.37 28.17 38.17 3.65 ...
##  $ gcf           : num  30.5 30.8 17.9 18.1 35.3 ...
##  $ trade         : num  152.5 121.4 66.4 44.9 68.5 ...
##  $ gdp.pc        : num  557 4081 2525 1010 1425 ...
##  $ lfdi          : num  20.6 21.2 NA 18.3 18.7 ...
##  $ lgdp.dflt     : num  6.04 2.96 3.34 3.64 1.29 ...
##  $ lgdp.pc       : num  6.32 8.31 7.83 6.92 7.26 ...
```

```
df0 <- subset1
head(subset1)
```

```
##    pov country.code year income edu.total hlth  mil       fdi lbr.part unemp
## 1 21.4          AGO 2000      L      2.61 1.91 6.39  8.79e+08       NA    NA
## 2 14.6          AGO 2008     LM      2.69 3.32 3.57  1.68e+09       NA    NA
## 3 31.1          AGO 2018     LM      2.04 2.54 1.87 -6.46e+09       NA    NA
## 4  0.5          ALB 1996     LM      3.08   NA 1.38  9.01e+07     38.8  12.3
## 5  1.1          ALB 2002     LM      3.12 6.91 1.32  1.35e+08     59.6  15.8
## 6  0.6          ALB 2005     LM      3.28 6.34 1.35  2.62e+08     34.5  14.1
##   pop.gwth.total pop.gwth.rural gdp.dflt  gcf trade gdp.pc lfdi lgdp.dflt
## 1          3.277          0.921   418.02 30.5 152.5    557 20.6      6.04
## 2          3.711          1.910    19.37 30.8 121.4   4081 21.2      2.96
```

```
## 3          3.276          1.338    28.17 17.9  66.4   2525   NA      3.34
## 4         -0.622         -1.546    38.17 18.1  44.9   1010 18.3      3.64
## 5         -0.300         -2.169     3.65 35.3  68.5   1425 18.7      1.29
## 6         -0.512         -2.519     3.31 36.9  70.9   2674 19.4      1.20
##   lgdp.pc
## 1    6.32
## 2    8.31
## 3    7.83
## 4    6.92
## 5    7.26
## 6    7.89
```

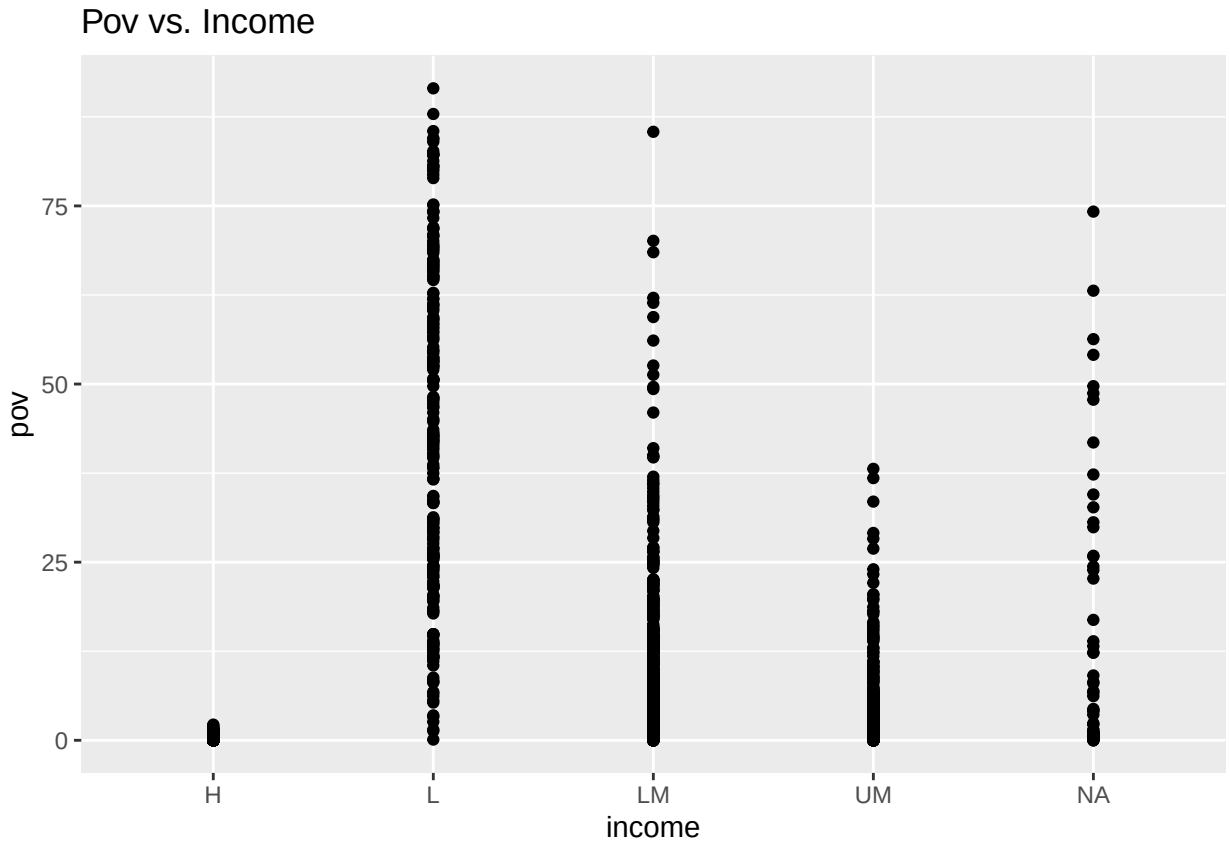
```
apply(subset1, 2, sd)
```

```
##            pov   country.code           year         income      edu.total
##          17.54             NA           9.26             NA             NA
##           hlth            mil            fdi       lbr.part         unemp
##             NA             NA             NA             NA             NA
## pop.gwth.total pop.gwth.rural       gdp.dflt            gcf          trade
##           1.12             NA             NA             NA             NA
##         gdp.pc           lfdi      lgdp.dflt        lgdp.pc
##             NA             NA             NA             NA
```

Next we will plot a pov vs income scatter plot to see the effect of income on pov. From the scatter plot, we can deduce that the low income countries have a higher pov while the higher income countries have low pov. This is consistent with our intuition that countries that are richer do not have as much poverty issue as the poorer countries.

```
ggplot(data = subset1, aes(x = income, y = pov)) +
    geom_point() + geom_smooth(method = "lm") + labs(title = "Pov vs. Income ",
    x = "income", y = "pov")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Next we will build a MLR model with data from subset 1 to see the relationship between dependent variable:pov and the other variables

```
model1 <- lm(pov ~ ., data = subset1)
summary(model1)
```

```
##
## Call:
## lm(formula = pov ~ ., data = subset1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.411 -0.723 -0.025  0.702 15.150
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3.41e+02   5.43e+01    6.28  6.0e-10 ***
## country.codeARG  3.26e+00   1.27e+00    2.56  0.01056 *
## country.codeARM -5.55e-01   1.31e+00   -0.42  0.67290
## country.codeAUS  2.51e+00   1.86e+00    1.35  0.17685
## country.codeAUT  3.39e+00   1.73e+00    1.96  0.04981 *
## country.codeAZE -5.55e+00   1.56e+00   -3.55  0.00041 ***
## country.codeBEL  5.28e+00   1.82e+00    2.90  0.00391 **
## country.codeBEN  1.46e+01   2.51e+00    5.82  9.1e-09 ***
## country.codeBFA  2.35e+01   2.59e+00    9.05  < 2e-16 ***
## country.codeBGD  1.36e+01   1.94e+00    7.01  5.5e-12 ***
```

```
## country.codeBGR  4.87e+00   1.22e+00    4.01  6.8e-05 ***
## country.codeBLR  3.43e+00   1.30e+00    2.65  0.00818 **
## country.codeBOL  4.89e+00   1.67e+00    2.93  0.00355 **
## country.codeBRA  8.40e+00   1.32e+00    6.38  3.3e-10 ***
## country.codeBWA  2.13e+01   2.42e+00    8.80  < 2e-16 ***
## country.codeCAN  2.67e+00   2.06e+00    1.29  0.19657
## country.codeCHE  8.59e-01   2.02e+00    0.42  0.67157
## country.codeCHL  4.26e+00   1.30e+00    3.27  0.00111 **
## country.codeCHN  1.50e+01   1.43e+00   10.49  < 2e-16 ***
## country.codeCMR  2.23e+01   1.98e+00   11.26  < 2e-16 ***
## country.codeCOG  4.29e+01   2.44e+00   17.60  < 2e-16 ***
## country.codeCOL  7.96e+00   1.23e+00    6.47  1.8e-10 ***
## country.codeCPV  4.22e+00   2.36e+00    1.79  0.07378 .
## country.codeCRI  6.10e+00   1.24e+00    4.91  1.1e-06 ***
## country.codeCYP  3.61e+00   1.65e+00    2.18  0.02951 *
## country.codeCZE  4.57e+00   1.38e+00    3.32  0.00093 ***
## country.codeDEU  2.96e+00   1.70e+00    1.74  0.08250 .
## country.codeDNK  2.72e+00   1.81e+00    1.50  0.13349
## country.codeDOM  5.11e+00   1.15e+00    4.44  1.0e-05 ***
## country.codeECU  6.06e+00   1.31e+00    4.65  4.0e-06 ***
## country.codeEGY -2.06e+00   1.64e+00   -1.25  0.21000
## country.codeESP  3.70e+00   1.39e+00    2.66  0.00806 **
## country.codeEST  4.84e+00   1.41e+00    3.44  0.00061 ***
## country.codeFIN  2.99e+00   1.58e+00    1.89  0.05918 .
## country.codeFJI  1.36e+00   2.30e+00    0.59  0.55576
## country.codeFRA  4.59e+00   1.98e+00    2.32  0.02075 *
## country.codeGBR  1.68e+00   1.63e+00    1.03  0.30413
## country.codeGEO  7.38e+00   1.15e+00    6.40  2.9e-10 ***
## country.codeGIN  3.59e+01   2.47e+00   14.55  < 2e-16 ***
## country.codeGNB  1.25e+01   2.47e+00    5.06  5.2e-07 ***
## country.codeGRC  1.87e+00   1.58e+00    1.19  0.23530
## country.codeGTM  8.20e+00   1.86e+00    4.42  1.1e-05 ***
## country.codeHND  1.35e+01   1.37e+00    9.82  < 2e-16 ***
## country.codeHRV  2.90e+00   1.38e+00    2.10  0.03641 *
## country.codeHUN  4.93e+00   1.43e+00    3.46  0.00058 ***
## country.codeIDN  1.40e+01   1.27e+00   11.05  < 2e-16 ***
## country.codeIND  9.52e+00   1.97e+00    4.83  1.7e-06 ***
## country.codeIRL  3.55e+00   1.83e+00    1.94  0.05310 .
## country.codeIRN  2.99e+00   1.22e+00    2.45  0.01459 *
## country.codeISR  1.38e+00   1.65e+00    0.84  0.40300
## country.codeITA  3.33e+00   1.48e+00    2.25  0.02467 *
## country.codeJAM  1.05e-02   1.77e+00    0.01  0.99526
## country.codeJOR  9.68e-01   2.37e+00    0.41  0.68351
## country.codeKAZ  2.79e+00   1.29e+00    2.17  0.03045 *
## country.codeKEN  2.46e+01   2.56e+00    9.63  < 2e-16 ***
## country.codeKGZ -5.64e-01   1.44e+00   -0.39  0.69488
## country.codeKOR  4.42e+00   1.88e+00    2.35  0.01894 *
## country.codeLKA -2.25e-01   1.59e+00   -0.14  0.88756
## country.codeLTU  4.25e+00   1.38e+00    3.09  0.00210 **
## country.codeLUX -4.08e+00   3.08e+00   -1.33  0.18475
## country.codeLVA  4.80e+00   1.33e+00    3.60  0.00034 ***
## country.codeMDA  1.49e+00   1.38e+00    1.08  0.27978
## country.codeMDG  6.53e+01   2.02e+00   32.33  < 2e-16 ***
## country.codeMEX  8.39e+00   1.31e+00    6.42  2.5e-10 ***
```

```
## country.codeMLI  6.73e+00   2.58e+00    2.61  0.00921 **
## country.codeMLT  7.26e+00   2.12e+00    3.42  0.00066 ***
## country.codeMMR -1.07e+00   1.97e+00   -0.54  0.58673
## country.codeMNG  4.09e+00   1.38e+00    2.95  0.00325 **
## country.codeMUS  2.90e+00   1.63e+00    1.77  0.07671 .
## country.codeMYS  4.65e+00   1.64e+00    2.84  0.00464 **
## country.codeNER  5.07e+01   2.70e+00   18.76  < 2e-16 ***
## country.codeNIC  2.81e+00   2.30e+00    1.22  0.22173
## country.codeNLD  4.36e+00   1.96e+00    2.22  0.02664 *
## country.codeNOR -9.63e-01   2.02e+00   -0.48  0.63329
## country.codePAK  9.39e-01   1.74e+00    0.54  0.58952
## country.codePAN  1.02e+01   1.25e+00    8.12  2.1e-15 ***
## country.codePER  8.66e+00   1.28e+00    6.75  3.1e-11 ***
## country.codePHL  5.26e+00   1.55e+00    3.39  0.00073 ***
## country.codePOL  2.66e+00   1.29e+00    2.06  0.04005 *
## country.codePRT  3.25e+00   1.39e+00    2.34  0.01959 *
## country.codePRY  3.21e+00   1.26e+00    2.55  0.01091 *
## country.codeROU  7.72e+00   1.20e+00    6.43  2.4e-10 ***
## country.codeRUS -1.01e+00   1.37e+00   -0.74  0.46204
## country.codeSDN  1.13e+01   2.42e+00    4.69  3.3e-06 ***
## country.codeSEN  3.58e+01   2.38e+00   15.05  < 2e-16 ***
## country.codeSLE  1.70e+01   2.60e+00    6.53  1.3e-10 ***
## country.codeSLV  4.60e+00   1.17e+00    3.93  9.1e-05 ***
## country.codeSRB  6.55e+00   1.29e+00    5.08  4.9e-07 ***
## country.codeSVK  3.87e+00   1.49e+00    2.60  0.00952 **
## country.codeSVN  4.47e+00   1.47e+00    3.04  0.00245 **
## country.codeSWE  3.18e+00   1.68e+00    1.89  0.05867 .
## country.codeSWZ  3.32e+01   2.31e+00   14.41  < 2e-16 ***
## country.codeSYC  6.49e+00   2.44e+00    2.65  0.00813 **
## country.codeTCD  2.14e+01   2.57e+00    8.33  4.0e-16 ***
## country.codeTGO  4.88e+01   2.11e+00   23.16  < 2e-16 ***
## country.codeTHA  1.83e+00   1.32e+00    1.39  0.16550
## country.codeTJK  6.78e+00   1.84e+00    3.69  0.00024 ***
## country.codeTUN  1.64e+00   1.57e+00    1.05  0.29465
## country.codeTUR  1.60e+00   1.38e+00    1.16  0.24836
## country.codeTZA  3.72e+01   2.57e+00   14.47  < 2e-16 ***
## country.codeUGA  3.16e+01   2.23e+00   14.17  < 2e-16 ***
## country.codeUKR -2.44e+00   1.33e+00   -1.83  0.06752 .
## country.codeURY  2.45e+00   1.33e+00    1.84  0.06572 .
## country.codeUSA  3.25e+00   2.42e+00    1.34  0.17986
## country.codeVEN  7.76e+00   2.30e+00    3.38  0.00076 ***
## country.codeZAF  2.41e+01   1.43e+00   16.83  < 2e-16 ***
## country.codeZMB  6.40e+01   2.37e+00   26.98  < 2e-16 ***
## year            -1.46e-01   2.80e-02   -5.21  2.5e-07 ***
## incomeL         -1.22e-02   9.74e-01   -0.01  0.99000
## incomeLM        -1.42e+00   6.15e-01   -2.30  0.02157 *
## incomeUM        -1.54e+00   4.17e-01   -3.70  0.00023 ***
## edu.total       -4.16e-01   1.47e-01   -2.83  0.00478 **
## hlth            -1.63e-01   1.26e-01   -1.30  0.19503
## mil              4.44e-01   1.70e-01    2.60  0.00938 **
## fdi             -1.04e-12   2.55e-12   -0.41  0.68494
## lbr.part         2.90e-02   2.79e-02    1.04  0.29968
## unemp            6.97e-02   3.73e-02    1.87  0.06228 .
## pop.gwth.total  -4.40e-01   3.31e-01   -1.33  0.18313
```

```
## pop.gwth.rural   2.30e-01   2.17e-01    1.06  0.28932
## gdp.dflt         2.19e-02   1.96e-02    1.11  0.26580
## gcf             -1.69e-01   2.38e-02   -7.12  2.7e-12 ***
## trade           -1.10e-02   6.58e-03   -1.68  0.09371 .
## gdp.pc           2.05e-04   1.99e-05   10.30  < 2e-16 ***
## lfdi            -8.85e-02   1.11e-01   -0.79  0.42741
## lgdp.dflt        5.58e-02   1.34e-01    0.42  0.67743
## lgdp.pc         -4.81e+00   4.43e-01  -10.87  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.97 on 709 degrees of freedom
##   (1008 observations deleted due to missingness)
## Multiple R-squared:  0.961,  Adjusted R-squared:  0.954
## F-statistic:  138 on 125 and 709 DF,  p-value: <2e-16
```

```
vif(model1)
```

```
##                    GVIF  Df GVIF^(1/(2*Df))
## country.code   4.67e+12 106            1.15
## year           4.59e+00   1            2.14
## income         1.47e+02   3            2.30
## edu.total      8.90e+00   1            2.98
## hlth           1.65e+01   1            4.07
## mil            8.44e+00   1            2.90
## fdi            4.80e+00   1            2.19
## lbr.part       9.43e+00   1            3.07
## unemp          5.79e+00   1            2.41
## pop.gwth.total 2.15e+01   1            4.64
## pop.gwth.rural 1.80e+01   1            4.24
## gdp.dflt       4.02e+00   1            2.00
## gcf            4.89e+00   1            2.21
## trade          2.06e+01   1            4.54
## gdp.pc         3.16e+01   1            5.62
## lfdi           9.51e+00   1            3.08
## lgdp.dflt      4.59e+00   1            2.14
## lgdp.pc        6.50e+01   1            8.06
```

Next we will perform regularisation. We will split the data to predictor variable "train.x" and response variables "train.y"

```
subset2 <- subset(subset1, select = c("pov", "country.code",
    "year", "income", "edu.total", "hlth", "mil", "fdi",
    "lbr.part", "unemp", "pop.gwth.total", "pop.gwth.rural",
    "gdp.dflt", "gcf", "trade", "gdp.pc", "lfdi", "lgdp.dflt",
    "lgdp.pc"))
subset2 <- na.omit(subset2)
head(subset2)
```

```
##    pov country.code year income edu.total hlth  mil      fdi lbr.part unemp
## 5  1.1          ALB 2002     LM      3.12 6.91 1.32 1.35e+08     59.6  15.8
## 6  0.6          ALB 2005     LM      3.28 6.34 1.35 2.62e+08     34.5  14.1
## 8  0.6          ALB 2012     UM      2.93 5.06 1.49 9.18e+08     57.0  13.4
```

```
## 9  1.0          ALB 2014     UM      3.05 5.50 1.35 1.15e+09     53.4  18.0
## 10 0.1          ALB 2015     UM      3.44 4.90 1.16 9.90e+08     55.5  17.2
## 12 0.4          ALB 2017     UM      3.61 5.01 1.11 1.02e+09     58.1  13.6
##    pop.gwth.total pop.gwth.rural gdp.dflt  gcf trade gdp.pc lfdi lgdp.dflt
## 5          -0.300          -2.17    3.647 35.3  68.5   1425 18.7    1.2940
## 6          -0.512          -2.52    3.307 36.9  70.9   2674 19.4    1.1960
## 8          -0.165          -2.51    1.043 28.3  76.5   4248 20.6    0.0418
## 9          -0.207          -2.56    1.550 25.7  75.4   4579 20.9    0.4382
## 10         -0.291          -2.64    0.564 25.8  71.8   3953 20.7   -0.5727
## 12         -0.092          -2.43    1.451 25.1  78.2   4531 20.7    0.3723
##    lgdp.pc
## 5     7.26
## 6     7.89
## 8     8.35
## 9     8.43
## 10    8.28
## 12    8.42
```

```
train.x <- data.matrix(subset(subset2, select = -pov))
train.y = data.matrix(subset2$pov)
```

Next we will perform splitting of data to train and test.

```
split_train_test <- function(data) {
    set.seed(1984)
    # split data
    seeds <- data %>%
        group_by(country.code) %>%
        filter(row_number() == 1)

    plants <- data %>%
        group_by(country.code) %>%
        filter(row_number() != 1)

    isComplete <- which(complete.cases(plants))
    idx <- sample(isComplete, replace = F, 0.2 * nrow(plants))

    training <- plants[-idx, ] %>%
        rbind(seeds)
    test <- plants[idx, ]

    return(list(training, test))
}

training <- split_train_test(subset2)[[1]]
test <- split_train_test(subset2)[[2]]

ntrain <- 1:nrow(training)
combine <- rbind(training, test)
combine.mtrx <- model.matrix(pov ~ ., data = combine)
train.mtrx <- combine.mtrx[ntrain, ]
test.mtrx <- combine.mtrx[-ntrain, ]
```

```r
train.x <- train.mtrx[, -1]
train.y <- training$pov

test.x <- test.mtrx[, -1]
test.y <- test$pov
```

We then build a MLR model with the training model to train it.

```r
model3 <- lm(pov ~ ., data = training)
summary(model3)
```

```
##
## Call:
## lm(formula = pov ~ ., data = training)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -9.213 -0.704 -0.018  0.621 14.165
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.38e+02   6.39e+01    5.30  1.7e-07 ***
## country.codeARG   3.92e+00   1.39e+00    2.83  0.00485 **
## country.codeARM -1.77e-01   1.41e+00   -0.13  0.90019
## country.codeAUS   2.95e+00   1.99e+00    1.48  0.14016
## country.codeAUT   3.76e+00   1.94e+00    1.94  0.05276 .
## country.codeAZE -6.13e+00   1.72e+00   -3.57  0.00039 ***
## country.codeBEL   5.77e+00   1.98e+00    2.91  0.00371 **
## country.codeBEN   1.42e+01   2.69e+00    5.29  1.8e-07 ***
## country.codeBFA   2.32e+01   2.78e+00    8.36  4.9e-16 ***
## country.codeBGD   1.34e+01   2.06e+00    6.51  1.6e-10 ***
## country.codeBGR   4.70e+00   1.30e+00    3.62  0.00032 ***
## country.codeBLR   3.00e+00   1.44e+00    2.08  0.03801 *
## country.codeBOL   4.91e+00   1.91e+00    2.57  0.01037 *
## country.codeBRA   8.80e+00   1.41e+00    6.23  9.0e-10 ***
## country.codeBWA   2.10e+01   2.53e+00    8.31  7.0e-16 ***
## country.codeCAN   3.22e+00   2.20e+00    1.46  0.14398
## country.codeCHE   1.24e+00   2.21e+00    0.56  0.57567
## country.codeCHL   4.47e+00   1.39e+00    3.21  0.00141 **
## country.codeCHN   1.50e+01   1.55e+00    9.71  < 2e-16 ***
## country.codeCMR   2.24e+01   2.11e+00   10.59  < 2e-16 ***
## country.codeCOG   4.32e+01   2.56e+00   16.86  < 2e-16 ***
## country.codeCOL   8.15e+00   1.33e+00    6.14  1.5e-09 ***
## country.codeCPV   4.59e+00   2.47e+00    1.86  0.06307 .
## country.codeCRI   6.21e+00   1.35e+00    4.61  5.0e-06 ***
## country.codeCYP   3.51e+00   1.84e+00    1.90  0.05738 .
## country.codeCZE   4.37e+00   1.52e+00    2.87  0.00420 **
## country.codeDEU   3.36e+00   1.90e+00    1.77  0.07810 .
## country.codeDNK   2.83e+00   2.05e+00    1.38  0.16909
## country.codeDOM   4.99e+00   1.24e+00    4.02  6.6e-05 ***
## country.codeECU   5.90e+00   1.49e+00    3.96  8.5e-05 ***
## country.codeEGY -1.62e+00   1.76e+00   -0.92  0.35915
## country.codeESP   4.31e+00   1.52e+00    2.83  0.00480 **
```

```
## country.codeEST  4.65e+00   1.53e+00    3.04  0.00247 **
## country.codeFIN  3.20e+00   1.74e+00    1.84  0.06700 .
## country.codeFJI  1.07e+00   2.40e+00    0.45  0.65477
## country.codeFRA  5.12e+00   2.10e+00    2.44  0.01489 *
## country.codeGBR  2.12e+00   1.82e+00    1.16  0.24454
## country.codeGEO  7.28e+00   1.23e+00    5.92  5.6e-09 ***
## country.codeGIN  3.55e+01   2.60e+00   13.69  < 2e-16 ***
## country.codeGNB  1.26e+01   2.61e+00    4.83  1.8e-06 ***
## country.codeGRC  2.05e+00   1.77e+00    1.16  0.24662
## country.codeGTM  8.61e+00   1.96e+00    4.39  1.3e-05 ***
## country.codeHND  1.35e+01   1.61e+00    8.37  4.5e-16 ***
## country.codeHRV  3.01e+00   1.47e+00    2.05  0.04053 *
## country.codeHUN  5.06e+00   1.53e+00    3.31  0.00099 ***
## country.codeIDN  1.38e+01   1.41e+00    9.84  < 2e-16 ***
## country.codeIND  9.53e+00   2.10e+00    4.53  7.1e-06 ***
## country.codeIRL  3.97e+00   2.03e+00    1.96  0.05035 .
## country.codeIRN  3.12e+00   1.32e+00    2.37  0.01816 *
## country.codeISR  1.35e+00   1.78e+00    0.76  0.45013
## country.codeITA  3.66e+00   1.63e+00    2.24  0.02543 *
## country.codeJAM  2.41e-01   1.85e+00    0.13  0.89641
## country.codeJOR  1.35e+00   2.56e+00    0.53  0.59791
## country.codeKAZ  2.97e+00   1.39e+00    2.14  0.03252 *
## country.codeKEN  2.44e+01   2.71e+00    9.00  < 2e-16 ***
## country.codeKGZ  3.40e-02   1.59e+00    0.02  0.98293
## country.codeKOR  4.43e+00   1.97e+00    2.24  0.02521 *
## country.codeLKA -2.50e-01   1.67e+00   -0.15  0.88155
## country.codeLTU  4.11e+00   1.50e+00    2.73  0.00646 **
## country.codeLUX -4.07e+00   3.35e+00   -1.22  0.22476
## country.codeLVA  4.64e+00   1.46e+00    3.18  0.00158 **
## country.codeMDA  1.58e+00   1.49e+00    1.06  0.29058
## country.codeMDG  6.52e+01   2.18e+00   29.89  < 2e-16 ***
## country.codeMEX  8.55e+00   1.48e+00    5.78  1.2e-08 ***
## country.codeMLI  6.38e+00   2.76e+00    2.31  0.02105 *
## country.codeMLT  7.39e+00   2.32e+00    3.19  0.00151 **
## country.codeMMR -1.14e+00   2.09e+00   -0.54  0.58654
## country.codeMNG  3.64e+00   1.51e+00    2.41  0.01613 *
## country.codeMUS  2.73e+00   1.72e+00    1.59  0.11347
## country.codeMYS  4.26e+00   1.75e+00    2.43  0.01541 *
## country.codeNER  5.06e+01   2.89e+00   17.49  < 2e-16 ***
## country.codeNIC  3.20e+00   2.40e+00    1.33  0.18259
## country.codeNLD  4.58e+00   2.21e+00    2.07  0.03899 *
## country.codeNOR -7.14e-01   2.24e+00   -0.32  0.74962
## country.codePAK  2.00e+00   2.00e+00    1.00  0.31788
## country.codePAN  1.04e+01   1.33e+00    7.83  2.5e-14 ***
## country.codePER  9.09e+00   1.39e+00    6.56  1.2e-10 ***
## country.codePHL  4.85e+00   1.72e+00    2.82  0.00496 **
## country.codePOL  2.76e+00   1.40e+00    1.96  0.05019 .
## country.codePRT  3.61e+00   1.52e+00    2.38  0.01767 *
## country.codePRY  3.17e+00   1.36e+00    2.33  0.02039 *
## country.codeROU  8.39e+00   1.36e+00    6.19  1.2e-09 ***
## country.codeRUS -1.12e+00   1.46e+00   -0.77  0.44166
## country.codeSDN  1.16e+01   2.53e+00    4.58  5.6e-06 ***
## country.codeSEN  3.60e+01   2.51e+00   14.36  < 2e-16 ***
## country.codeSLE  1.70e+01   2.76e+00    6.17  1.3e-09 ***
```

```
## country.codeSLV  4.50e+00   1.29e+00    3.49  0.00053 ***
## country.codeSRB  6.79e+00   1.36e+00    4.98  8.6e-07 ***
## country.codeSVK  3.89e+00   1.61e+00    2.41  0.01622 *
## country.codeSVN  4.48e+00   1.61e+00    2.78  0.00560 **
## country.codeSWE  3.38e+00   1.87e+00    1.81  0.07077 .
## country.codeSWZ  3.36e+01   2.41e+00   13.95  < 2e-16 ***
## country.codeSYC  6.21e+00   2.56e+00    2.43  0.01547 *
## country.codeTCD  2.13e+01   2.76e+00    7.71  5.6e-14 ***
## country.codeTGO  4.85e+01   2.28e+00   21.25  < 2e-16 ***
## country.codeTHA  1.64e+00   1.41e+00    1.17  0.24446
## country.codeTJK  6.66e+00   1.98e+00    3.36  0.00083 ***
## country.codeTUN  1.81e+00   1.64e+00    1.10  0.27090
## country.codeTUR  1.07e+00   1.63e+00    0.66  0.51214
## country.codeTZA  3.72e+01   2.72e+00   13.66  < 2e-16 ***
## country.codeUGA  3.16e+01   2.40e+00   13.17  < 2e-16 ***
## country.codeUKR -2.49e+00   1.45e+00   -1.72  0.08649 .
## country.codeURY  2.63e+00   1.47e+00    1.79  0.07380 .
## country.codeUSA  4.63e+00   2.77e+00    1.67  0.09454 .
## country.codeVEN  7.94e+00   2.38e+00    3.33  0.00092 ***
## country.codeZAF  2.63e+01   1.86e+00   14.14  < 2e-16 ***
## country.codeZMB  6.44e+01   2.50e+00   25.79  < 2e-16 ***
## year            -1.44e-01   3.29e-02   -4.38  1.4e-05 ***
## incomeL          7.52e-02   1.09e+00    0.07  0.94494
## incomeLM        -1.60e+00   6.92e-01   -2.31  0.02131 *
## incomeUM        -1.45e+00   4.50e-01   -3.23  0.00133 **
## edu.total       -3.72e-01   1.67e-01   -2.23  0.02629 *
## hlth            -2.48e-01   1.41e-01   -1.75  0.08019 .
## mil              5.20e-01   1.84e-01    2.82  0.00499 **
## fdi             -1.54e-12   2.87e-12   -0.54  0.59207
## lbr.part         3.07e-02   3.03e-02    1.01  0.31114
## unemp            5.28e-02   4.20e-02    1.26  0.20974
## pop.gwth.total  -5.21e-01   3.75e-01   -1.39  0.16464
## pop.gwth.rural   2.20e-01   2.45e-01    0.90  0.36877
## gdp.dflt         2.87e-02   2.23e-02    1.29  0.19864
## gcf             -1.64e-01   2.66e-02   -6.17  1.3e-09 ***
## trade           -1.00e-02   7.23e-03   -1.38  0.16727
## gdp.pc           2.06e-04   2.17e-05    9.51  < 2e-16 ***
## lfdi            -1.27e-01   1.21e-01   -1.05  0.29306
## lgdp.dflt       -1.29e-02   1.54e-01   -0.08  0.93334
## lgdp.pc         -4.85e+00   5.01e-01   -9.68  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.02 on 564 degrees of freedom
## Multiple R-squared:  0.965,  Adjusted R-squared:  0.957
## F-statistic:  124 on 125 and 564 DF,  p-value: <2e-16
```

```
fit <- model3$fitted.values
true <- training$pov
summary_MLR3_train <- eval_results(true, fit)
```

We then evaluate the model between train and test

```
fit <- predict(model3, newdata = test)
true <- test$pov
summary_MLR3_test <- eval_results(true, fit)
```

We can see the evaluation of the model from the function below. From the train model, the R^squared value will be 0.965 and for the test model it will be 0.881.

```
summary_MLR <- rbind(summary_MLR3_train[-5], summary_MLR3_test[-5])
rownames(summary_MLR) <- c("Baseline MLR_Train", "Baseline MLR_Test")
knitr::kable(summary_MLR, digits = 3)
```

	MSE	MAE	RMSE	Rsquare
Baseline MLR_Train	3.34	1.14	1.83	0.965
Baseline MLR_Test	3.60	1.29	1.90	0.881

Next we will be performing ridge and lasso regression model using the following code. We will split the data to train and test.

```
# Training Data
data <- training
train.x <- data.matrix(data[, -ncol(data)])
train.y <- data.matrix(data[, ncol(data)])

# Test Data
data <- test
test.x <- data.matrix(data[, -ncol(data)])
test.y <- data.matrix(data[, ncol(data)])
```

Next we will perform the Ridge regression model.

```
# Fit Ridge Regression Model
set.seed(123)
cv_ridge <- cv.glmnet(train.x, train.y, alpha = 0)
glm_ridge <- glmnet(train.x, train.y, alpha = 0, lambda = cv_ridge$lambda.min)

# Evaluate Ridge Regression Model on Training Set
fit <- predict(glm_ridge, newx = train.x, type = "response")
true <- train.y
summary_ridge_train <- eval_results(true, fit)


# Evaluate Ridge Regression Model on Test Set
fit <- predict(glm_ridge, newx = test.x, type = "response")
true <- test.y
summary_ridge_test <- eval_results(true, fit)

# Summarise Ridge Regression Model Performance
# into Table
summary_ridge <- rbind(summary_ridge_train, summary_ridge_test)
rownames(summary_ridge) <- c("Ridge Model_Train", "Ridge Model_Test")
```

Next we will perform Lasso Regression model

```
# Fit LASSO Model
set.seed(123)
cv_lasso <- cv.glmnet(train.x, train.y, alpha = 1)
glm_lasso <- glmnet(train.x, train.y, alpha = 1, lambda = cv_lasso$lambda.min)

# Evaluate LASSO Model on Training Set
fit <- predict(glm_lasso, newx = train.x)
true <- train.y
summary_lasso_train <- eval_results(true, fit)

# Evaluate LASSO Model on Test Set
fit <- predict(glm_lasso, newx = test.x)
true <- test.y
summary_lasso_test <- eval_results(true, fit)

# Summarise LASSO Model Performance into Table
summary_lasso <- rbind(summary_lasso_train, summary_lasso_test)
rownames(summary_lasso) <- c("LASSO Model_Train", "LASSO Model_Test")
```

We will compare the 3 models that we have build, Multi linear ,Ridge and Lasso regression models. We evaluate the 3 models together and compare both train and test models of each of the 3 regression model.

```
summary_3models <- rbind(summary_MLR3_train, summary_ridge_train,
    summary_lasso_train, summary_MLR3_test, summary_ridge_test,
    summary_lasso_test)

rownames(summary_3models) <- c("Baseline MLR_Train",
    "Ridge Model_Train", "LASSO Model_Train", "Baseline MLR_Test",
    "Ridge Model_Test", "LASSO Model_Test")

knitr::kable(summary_3models[1:3, -5], digits = 3)
```

	MSE	MAE	RMSE	Rsquare
Baseline MLR_Train	3.336	1.137	1.826	0.965
Ridge Model_Train	0.221	0.356	0.470	0.862
LASSO Model_Train	0.216	0.352	0.465	0.865

```
knitr::kable(summary_3models[4:6, -5], digits = 3)
```

	MSE	MAE	RMSE	Rsquare
Baseline MLR_Test	3.598	1.293	1.897	0.881
Ridge Model_Test	0.171	0.318	0.414	0.863
LASSO Model_Test	0.159	0.311	0.398	0.873

From the 3 models we can see that the original multi linear regression model is the best one with the highest R^squared value. However the Lasso Regression Model has the lowest MSE, MAE and RMSE. We can conclude that the reason for the first model having the highest R^squared value might be due to overfitting as many values will get removed. Next we will build other models to see if they work better.

3.2. Selectively remove variables with high missing rate

```
needs("knitr", "readr", "tidyr", "dplyr", "ggplot2",
    "stats", "tidyverse", "e1071", "car", "glmnet",
    "corrplot", "plotmo")
```

```
countries <- read.csv("../data/countries.csv")
```

3.2.1. Model Fitting

A. Ordinary Linear Regression Previous analysis showed that of all the variables in the countries.csv dataset, many of them had a high missing rate of above 35%, an arbitrary percentage that our group has chosen to filter the predictor variables. This section attempts to build a model on variables that have missing rates of below 35%.

The previous analysis showed that the relations between the predicted variable and gdp.pc, gdp.dflt, and fdi are better fitted to logarithmic models. Hence, we transformed the data with the mutate() function before building the machine learning model.

```
subset0 <- subset(countries, select = c("country.code",
    "year", "edu.total", "fdi", "gcf", "gdp.dflt",
    "hlth", "income", "lbr.part", "mil", "pop.gwth.rural",
    "pop.gwth.urban", "pov", "trade", "unemp", "year",
    "gdp.pc"))

subset1_0 <- subset0 %>%
    mutate(lfdi = log(subset0$fdi)) %>%
    mutate(lgdp.pc = log(subset0$gdp.pc)) %>%
    mutate(lgdp.dflt = log(subset0$gdp.dflt))
```

subset1 contains the variables below 35% of missing values.

```
subset1 <- subset(subset1_0, select = c("edu.total",
    "lfdi", "gcf", "lgdp.dflt", "hlth", "income", "lbr.part",
    "mil", "pop.gwth.rural", "pop.gwth.urban", "pov",
    "trade", "unemp", "year", "lgdp.pc"))

subset1 <- na.omit(subset1)
```

We also performed standardisation to reduce the variance of our machine learning models. Below are the functions used to aid our standardisation step.

```
sd0 <- function(vct) {
    if (!is.numeric(vct)) {
        return(NA)
    }

    return(sd(vct, na.rm = T))
}
```

```
std0 <- function(vct, scl) {
    if (!is.numeric(vct)) {
        return(vct)
    }

    return(vct/scl)
}

scale <- function(data) {
    unlist(lapply(data, sd0))
}

standardise <- function(data) {
    scaler <- scale(data)

    numCols <- which(unlist(lapply(data, is.numeric)))
    num <- as.data.frame(mapply(std0, vct = data[,
        numCols], scl = scaler[numCols], SIMPLIFY = T))
    fct <- data[, -numCols]
    return(list(cbind(fct, num), scaler))
}
```

```
df.countries0 <- standardise(subset1)[[1]]
scaler2 <- standardise(subset1)[[2]]
df.countries <- df.countries0
```

```
# Divide each feature/target by its standard
# deviation
head(df.countries)
```

```
##   fct edu.total  lfdi  gcf lgdp.dflt hlth lbr.part   mil pop.gwth.rural
## 1  LM      2.25  9.91 5.55    1.1870 3.13     7.94 1.135          -1.62
## 2  LM      2.37 10.26 5.80    1.0971 2.87     4.59 1.161          -1.89
## 3  UM      2.11 10.92 4.46    0.0384 2.29     7.59 1.279          -1.88
## 4  UM      2.20 11.04 4.04    0.4020 2.49     7.11 1.158          -1.91
## 5  UM      2.48 10.96 4.06   -0.5254 2.22     7.39 0.999          -1.98
## 6  UM      2.60 10.98 3.94    0.3415 2.27     7.73 0.953          -1.82
##   pop.gwth.urban    pov trade unemp year lgdp.pc
## 1           1.70 0.1202  1.45  3.59  383    5.84
## 2           1.43 0.0656  1.50  3.20  384    6.34
## 3           1.44 0.0656  1.62  3.04  385    6.72
## 4           1.28 0.1093  1.60  4.10  385    6.78
## 5           1.16 0.0109  1.52  3.91  386    6.66
## 6           1.20 0.0437  1.66  3.10  386    6.77
```

First, we split the data into training and testing data.

```
set.seed(123)
index <- sort(sample(x = nrow(df.countries), size = nrow(df.countries) *
    0.8))
train_1 <- subset1[index, ]
test_1 <- subset1[-index, ]
```

Our first attempt

```
# Everything
lm_1 <- lm(data = train_1, formula = pov ~ .)
summary(lm_1)
```

```
##
## Call:
## lm(formula = pov ~ ., data = train_1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -24.10  -2.65  -0.36   2.09  50.94
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    205.02815   92.92122    2.21  0.02770 *
## edu.total       -0.80558    0.19166   -4.20  3.0e-05 ***
## lfdi             0.13301    0.17075    0.78  0.43626
## gcf             -0.10351    0.03776   -2.74  0.00628 **
## lgdp.dflt       -0.06451    0.24743   -0.26  0.79438
## hlth             0.22709    0.14684    1.55  0.12246
## incomeL          6.84962    2.36679    2.89  0.00393 **
## incomeLM        -3.45936    1.48782   -2.33  0.02037 *
## incomeUM        -3.45134    1.00494   -3.43  0.00063 ***
## lbr.part         0.22187    0.03333    6.66  5.9e-11 ***
## mil             -0.93851    0.21070   -4.45  9.9e-06 ***
## pop.gwth.rural   0.63592    0.19567    3.25  0.00121 **
## pop.gwth.urban   1.92366    0.20369    9.44  < 2e-16 ***
## trade           -0.01476    0.00553   -2.67  0.00776 **
## unemp            0.28941    0.05994    4.83  1.7e-06 ***
## year            -0.08979    0.04615   -1.95  0.05213 .
## lgdp.pc         -3.51559    0.60014   -5.86  7.4e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.86 on 651 degrees of freedom
## Multiple R-squared:  0.588,  Adjusted R-squared:  0.578
## F-statistic: 58.1 on 16 and 651 DF,  p-value: <2e-16
```

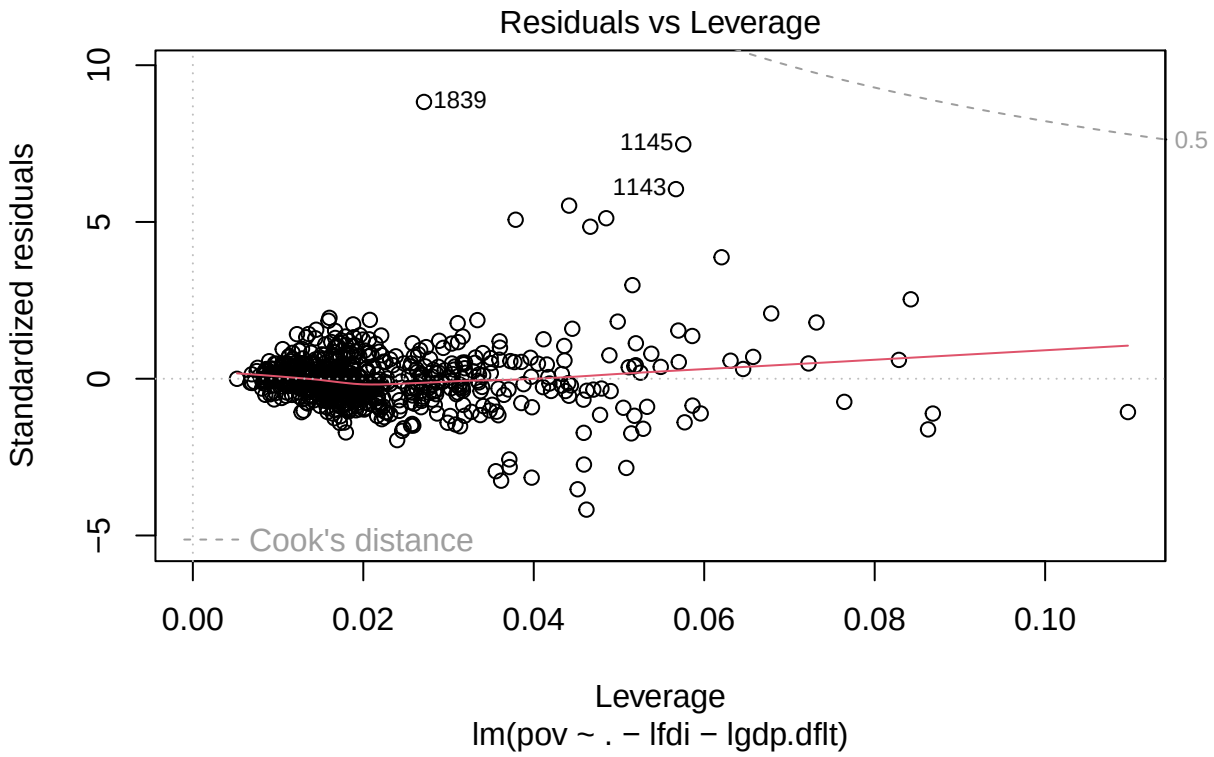
The summary of `lm_1` suggests that the model still contains predictor variables with $\text{pr}(>|t|) > 0.05$. We can imply that these variables do not provide a high significance to our model and are thus able to be removed. To avoid removing significant variables, we shall conduct the removal of the variables deliberately. In `lm_1`, we notice that the variable `lgdp.dflt` has a high $\text{Pr}(>|t|)$. Hence, we remove `lgdp.dflt`.

```
# remove hlth
lm_2 <- lm(data = train_1, formula = pov ~ . - lfdi -
    lgdp.dflt)
summary(lm_2)
```

```
##
## Call:
## lm(formula = pov ~ . - lfdi - lgdp.dflt, data = train_1)
```

```
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -23.88  -2.65  -0.37   2.10  50.98
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    206.1259    90.5962    2.28  0.02322 *
## edu.total       -0.8282     0.1892   -4.38  1.4e-05 ***
## gcf             -0.1015     0.0375   -2.71  0.00697 **
## hlth             0.2235     0.1466    1.52  0.12777
## incomeL          6.8791     2.3514    2.93  0.00356 **
## incomeLM        -3.4336     1.4653   -2.34  0.01942 *
## incomeUM        -3.4478     0.9778   -3.53  0.00045 ***
## lbr.part         0.2239     0.0331    6.77  2.8e-11 ***
## mil             -0.9289     0.2099   -4.42  1.1e-05 ***
## pop.gwth.rural   0.6157     0.1938    3.18  0.00156 **
## pop.gwth.urban   1.9411     0.2022    9.60  < 2e-16 ***
## trade           -0.0156     0.0054   -2.89  0.00394 **
## unemp            0.2851     0.0591    4.82  1.8e-06 ***
## year            -0.0897     0.0451   -1.99  0.04716 *
## lgdp.pc         -3.3417     0.5584   -5.98  3.6e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.86 on 653 degrees of freedom
## Multiple R-squared:  0.588,  Adjusted R-squared:  0.579
## F-statistic: 66.5 on 14 and 653 DF,  p-value: <2e-16
```

```
plot(lm_2, which = 5)
```

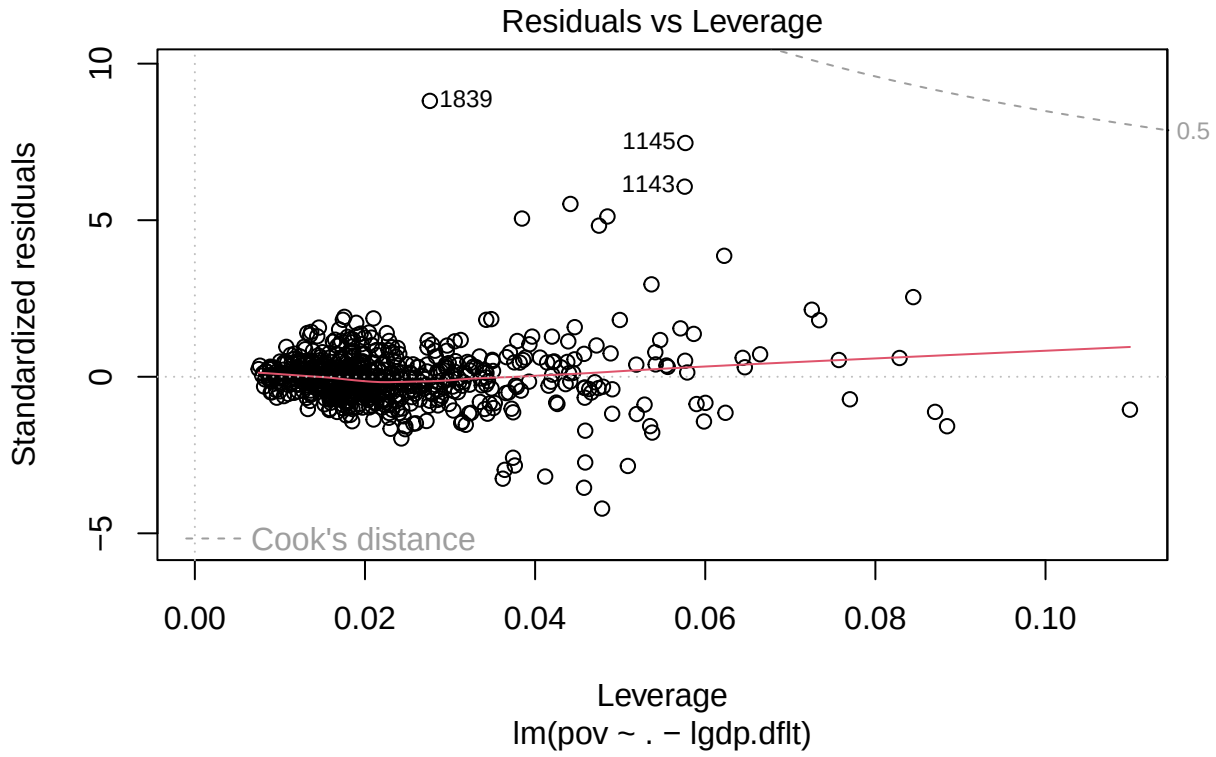



```
# Remove lgdp.dflt
lm_3 <- lm(data = train_1, formula = pov ~ . - lgdp.dflt)
summary(lm_3)
```

```
##
## Call:
## lm(formula = pov ~ . - lgdp.dflt, data = train_1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -24.06  -2.71  -0.37   2.10  50.88
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    200.16324   90.96351    2.20  0.02812 *
## edu.total       -0.80991    0.19080   -4.24  2.5e-05 ***
## lfdi             0.12942    0.17007    0.76  0.44694
## gcf             -0.10409    0.03767   -2.76  0.00588 **
## hlth             0.22762    0.14672    1.55  0.12129
## incomeL          6.79249    2.35494    2.88  0.00405 **
## incomeLM        -3.51750    1.46997   -2.39  0.01700 *
## incomeUM        -3.50707    0.98125   -3.57  0.00038 ***
## lbr.part         0.22282    0.03310    6.73  3.7e-11 ***
## mil             -0.93973    0.21050   -4.46  9.5e-06 ***
## pop.gwth.rural   0.63483    0.19548    3.25  0.00122 **
## pop.gwth.urban   1.92396    0.20354    9.45  < 2e-16 ***
```

```
## trade           -0.01474    0.00552   -2.67  0.00781 **
## unemp            0.29089    0.05962    4.88  1.3e-06 ***
## year            -0.08742    0.04521   -1.93  0.05361 .
## lgdp.pc         -3.50487    0.59830   -5.86  7.4e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.86 on 652 degrees of freedom
## Multiple R-squared:  0.588,  Adjusted R-squared:  0.579
## F-statistic:   62 on 15 and 652 DF,  p-value: <2e-16
```

```
plot(lm_3, which = 5)
```

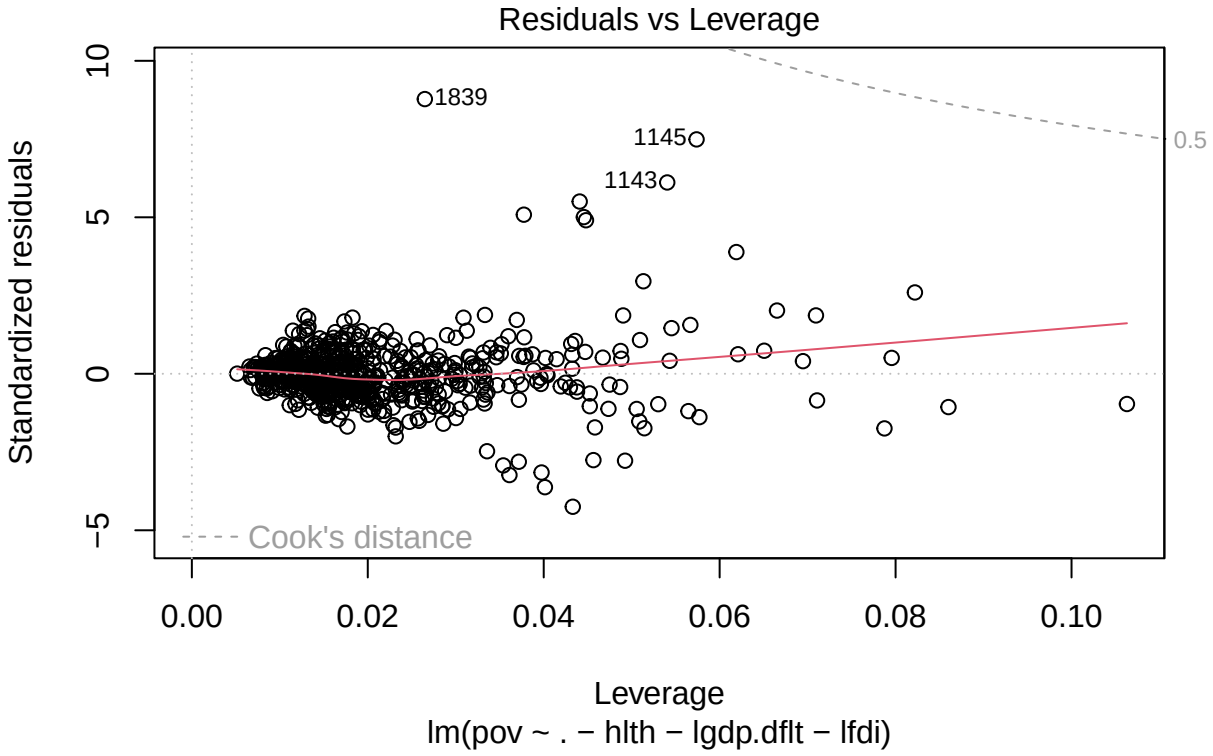



```
# remove lfdi
lm_4 <- lm(data = train_1, formula = pov ~ . - hlth -
    lgdp.dflt - lfdi)
summary(lm_4)
```

```
##
## Call:
## lm(formula = pov ~ . - hlth - lgdp.dflt - lfdi, data = train_1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -24.36  -2.61  -0.36   1.99  50.76
```

```
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    186.36637   89.75554    2.08  0.03825 *
## edu.total       -0.73952    0.18026   -4.10  4.6e-05 ***
## gcf             -0.11112    0.03701   -3.00  0.00278 **
## incomeL          7.31879    2.33605    3.13  0.00181 **
## incomeLM        -3.33101    1.46529   -2.27  0.02333 *
## incomeUM        -3.57242    0.97541   -3.66  0.00027 ***
## lbr.part         0.21730    0.03280    6.62  7.3e-11 ***
## mil             -0.93100    0.21016   -4.43  1.1e-05 ***
## pop.gwth.rural   0.57354    0.19201    2.99  0.00292 **
## pop.gwth.urban   1.90680    0.20117    9.48  < 2e-16 ***
## trade           -0.01696    0.00533   -3.18  0.00153 **
## unemp            0.30422    0.05783    5.26  1.9e-07 ***
## year            -0.07992    0.04469   -1.79  0.07416 .
## lgdp.pc         -3.13114    0.54157   -5.78  1.1e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.86 on 654 degrees of freedom
## Multiple R-squared:  0.586,  Adjusted R-squared:  0.578
## F-statistic: 71.3 on 13 and 654 DF,  p-value: <2e-16
```

```
plot(lm_4, which = 5)
```



Our model shows that among all the 14 predictor variables that we started with, only 11 of them showed

significance in building the model.

We then evaluated the results of the model.

```
# evaluating lm_4
eval.metrics.linreg <- function(actual, predicted) {
    residual <- actual - predicted
    mse <- mean(residual^2)
    mae <- mean(abs(residual))
    rmse <- sqrt(mse)
    mape <- mean(abs(residual/actual)) * 100

    data.frame(MSE = mse, MAE = mae, RMSE = rmse, MAPE = mape)
}
```

```
actual <- train_1$pov
predicted <- predict(lm_4, newdata = train_1)
eval.metrics.linreg(actual, predicted)
```

```
##    MSE  MAE RMSE MAPE
## 1 33.6 3.56  5.8  Inf
```

The MAPE shows `Inf` due to the presence of 0 values in the Y variable.

B. Regularization Manual multiple regression models are prone to standard error. Hence, Ridge regression and LASSO regression are also used to build our model.

Next, we will perform Ridge regression and LASSO regression in a dataset that contains `country.code` and `year` in addition to the previous dataset that we used for the previous linear regression model.

```
# Subsetting the data

data0 <- subset(subset1_0, select = c("country.code",
    "year", "edu.total", "lfdi", "gcf", "lgdp.dflt",
    "hlth", "income", "lbr.part", "mil", "pop.gwth.rural",
    "pop.gwth.urban", "pov", "trade", "unemp", "year",
    "lgdp.pc"))
data <- na.omit(data0)
train.x <- data.matrix(subset(data, select = -pov))
train.y <- data.matrix(data$pov)
```

Splitting the aforementioned dataset into training and testing data.

```
split_train_test <- function(data) {
    set.seed(1984)
    # split data
    seeds <- data %>%
        group_by(country.code) %>%
        filter(row_number() == 1)

    plants <- data %>%
        group_by(country.code) %>%
        filter(row_number() != 1)
```

```
    isComplete <- which(complete.cases(plants))
    idx <- sample(isComplete, replace = F, 0.2 * nrow(plants))

    training <- plants[-idx, ] %>%
        rbind(seeds)
    test <- plants[idx, ]

    return(list(training, test))
}
```

```
training <- split_train_test(data)[[1]]
test <- split_train_test(data)[[2]]
```

Further splitting the training and testing data into `train.x` and `train.y`

```
ntrain <- 1:nrow(training)
combine <- rbind(training, test)
combine.mtrx <- model.matrix(pov ~ ., data = combine)
train.mtrx <- combine.mtrx[ntrain, ]
test.mtrx <- combine.mtrx[-ntrain, ]

train.x <- train.mtrx[, -1]
train.y <- training$pov

test.x <- test.mtrx[, -1]
test.y <- test$pov
```

Building a model with the lambda that outputs the smallest MSE.

```
cv_ridge <- cv.glmnet(train.x, train.y, alpha = 0)
glm_Ridge <- glmnet(train.x, train.y, alpha = 0, lambda = cv_ridge$lambda.min)

head(t(coef(glm_Ridge)))
```

```
## 1 x 124 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 124 column names '(Intercept)', 'country.codeARG', 'country.codeARM' ... ]]

##
## s0 321 -1.51 -3.41 -0.797 -0.142 -9.69 1.76 2.02 11.9 6.16 1.25 -1.41 -1.41
##
## s0 2.44 11.5 -2.1 -1.1 -1.15 6.28 12.5 30.2 2.46 -3.11 -1.42 -2.58 -1.26 0.328
##
## s0 0.411 -1.77 -0.63 -6.11 -1.3 -0.178 -0.297 -2.88 1.56 -1.16 3.21 26.5 2.65
##
## s0 -1.94 0.424 5.41 -0.831 0.303 5.48 4.93 -0.823 -1.58 -2.52 -0.13 -5.76 -5.15
##
## s0 -3.91 13.3 -5.24 -0.61 -2.78 0.744 0.085 0.694 -1.19 47.5 1.15 -3.21 0.00292
##
## s0 -2.52 -3.3 -2.78 -2.71 34.2 -2.74 1.73 -0.432 -1.86 0.693 2.56 -0.0818 -1.05
##
```

```
## s0 -1.23 -3.52 3.83 -3.12 7.75 25.4 8.68 -1.19 3.06 -1.49 -1.31 -0.139 23.9
##
## s0 -1.7 10.4 33.7 -4.52 1.77 -2.79 -3.6 20.8 16.7 -3.76 -1.91 2.53 0.494 17.4
##
## s0 48.6 -0.0739 -0.452 -0.26 -0.113 0.0536 -0.143 7.12 0.747 -0.899 0.0895
##
## s0 -0.105 0.449 0.839 -0.00776 0.0675 -0.075 -1.31
```

The ridge regression model shows that some countries have a higher influence on the model than others. At first glance,it seems like the Y variable, `pov`, depends heavily on whether or not it belongs to certain countries.To further illustrate the significance of `country.code`, we will plot the ridge regression graph for different lambda values.

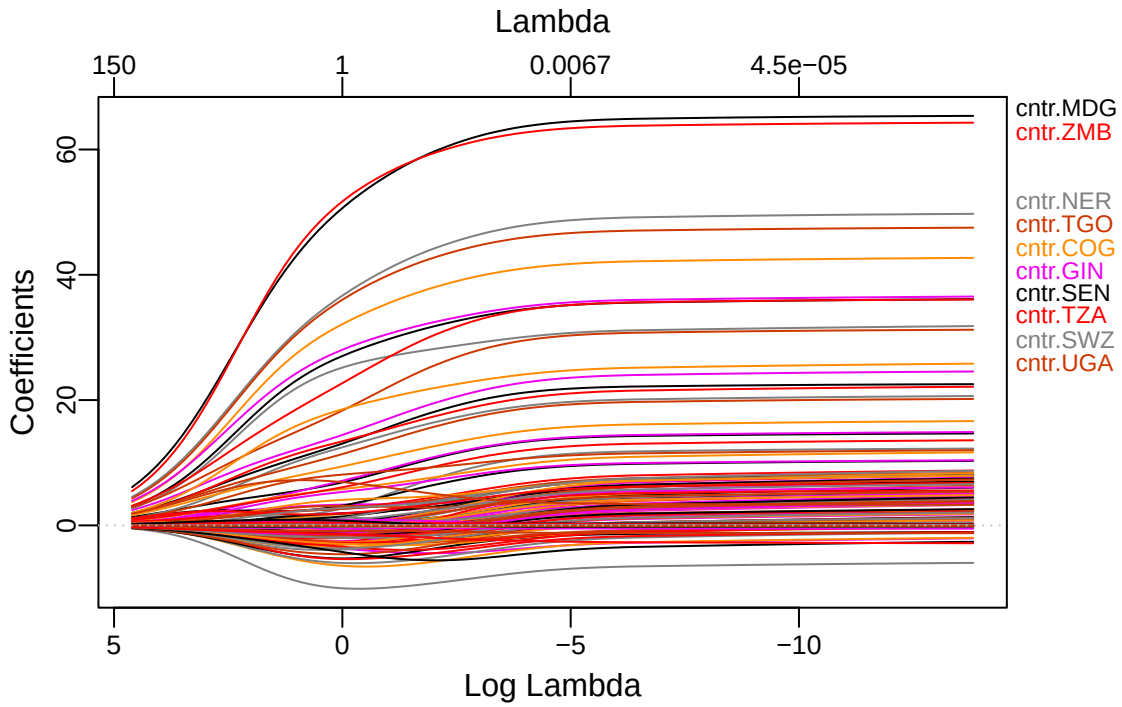
```
# generate sequence of lambda values
lambda <- 10^seq(-6, 2, length = 100)
ridge_model = glmnet(train.x, train.y, alpha = 0, lambda = lambda)

head(t(coef(ridge_model)))
```

```
## 6 x 124 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 124 column names '(Intercept)', 'country.codeARG', 'country.codeARM' ... ]]

##
## s0  89 -0.166 -0.311 -0.292 -0.269 -0.505 -0.204 1.02 2.01 1.35 -0.1018 -0.357
## s1 101 -0.195 -0.370 -0.335 -0.301 -0.637 -0.212 1.15 2.32 1.54 -0.1052 -0.415
## s2 113 -0.229 -0.438 -0.382 -0.335 -0.801 -0.213 1.28 2.67 1.75 -0.1048 -0.479
## s3 127 -0.268 -0.517 -0.432 -0.369 -1.002 -0.206 1.40 3.05 1.97 -0.0989 -0.549
## s4 140 -0.312 -0.607 -0.486 -0.401 -1.246 -0.187 1.53 3.46 2.20 -0.0858 -0.625
## s5 155 -0.362 -0.708 -0.543 -0.432 -1.537 -0.155 1.63 3.90 2.44 -0.0639 -0.705
##
## s0 -0.00386 0.237 1.16 -0.322 -0.319 -0.180 0.568 1.89 3.73 0.373 -0.0710
## s1 -0.01595 0.284 1.37 -0.372 -0.365 -0.210 0.667 2.20 4.37 0.434 -0.0982
## s2 -0.03295 0.340 1.61 -0.429 -0.415 -0.245 0.781 2.54 5.10 0.503 -0.1340
## s3 -0.05587 0.404 1.89 -0.492 -0.469 -0.285 0.911 2.92 5.92 0.580 -0.1804
## s4 -0.08573 0.478 2.21 -0.561 -0.527 -0.329 1.059 3.33 6.84 0.665 -0.2397
## s5 -0.12339 0.562 2.57 -0.638 -0.587 -0.377 1.227 3.79 7.86 0.757 -0.3142
##
## s0 -0.155 -0.374 -0.340 -0.254 -0.250 -0.0975 -0.0537 -0.387 -0.263 -0.260
## s1 -0.185 -0.439 -0.392 -0.277 -0.275 -0.1252 -0.0706 -0.476 -0.299 -0.295
## s2 -0.219 -0.512 -0.449 -0.299 -0.298 -0.1599 -0.0917 -0.583 -0.340 -0.332
## s3 -0.260 -0.595 -0.510 -0.317 -0.318 -0.2027 -0.1178 -0.712 -0.383 -0.369
## s4 -0.306 -0.688 -0.575 -0.330 -0.333 -0.2548 -0.1492 -0.865 -0.429 -0.407
## s5 -0.360 -0.792 -0.644 -0.337 -0.344 -0.3172 -0.1861 -1.044 -0.479 -0.442
##
## s0 -0.279 -0.330 -0.214 -0.286 0.543 3.91 1.17 -0.290 0.390 0.745 -0.257 -0.250
## s1 -0.312 -0.393 -0.225 -0.323 0.630 4.55 1.33 -0.335 0.437 0.869 -0.291 -0.275
## s2 -0.345 -0.466 -0.229 -0.363 0.726 5.27 1.48 -0.384 0.483 1.008 -0.327 -0.300
## s3 -0.378 -0.550 -0.226 -0.404 0.834 6.08 1.65 -0.438 0.529 1.165 -0.365 -0.321
## s4 -0.409 -0.645 -0.212 -0.447 0.951 6.96 1.80 -0.498 0.571 1.339 -0.403 -0.338
## s5 -0.439 -0.752 -0.187 -0.492 1.079 7.93 1.95 -0.563 0.609 1.530 -0.442 -0.349
##
## s0 0.847 0.496 -0.305 -0.317 -0.335 -0.211 -0.369 -0.449 -0.360 2.46 -0.0967
```

```
## s1 0.976 0.582 -0.349 -0.372 -0.391 -0.235 -0.455 -0.539 -0.438 2.83 -0.1469
## s2 1.119 0.680 -0.396 -0.436 -0.455 -0.259 -0.559 -0.645 -0.531 3.25 -0.2143
## s3 1.276 0.792 -0.447 -0.506 -0.527 -0.282 -0.684 -0.770 -0.641 3.71 -0.3027
## s4 1.446 0.920 -0.500 -0.584 -0.606 -0.303 -0.833 -0.914 -0.769 4.20 -0.4162
## s5 1.628 1.065 -0.556 -0.670 -0.695 -0.322 -1.007 -1.079 -0.917 4.72 -0.5585
##
## s0 -0.312 -0.168 -0.187 -0.270 -0.221 -0.157  6.10 0.0732 0.559 -0.261 -0.144
## s1 -0.360 -0.216 -0.204 -0.300 -0.247 -0.185  7.14 0.0852 0.600 -0.288 -0.186
## s2 -0.411 -0.274 -0.218 -0.330 -0.271 -0.216  8.31 0.0989 0.629 -0.315 -0.239
## s3 -0.466 -0.345 -0.228 -0.359 -0.294 -0.252  9.63 0.1146 0.641 -0.338 -0.302
## s4 -0.525 -0.429 -0.232 -0.385 -0.313 -0.291 11.11 0.1325 0.628 -0.358 -0.379
## s5 -0.585 -0.527 -0.229 -0.408 -0.326 -0.333 12.74 0.1530 0.585 -0.373 -0.468
##
## s0 -0.278 -0.375 -0.354 4.55 -0.139 -0.192 -0.302 0.473  0.012416 0.508 0.374
## s1 -0.346 -0.443 -0.417 5.31 -0.178 -0.195 -0.343 0.512  0.011038 0.584 0.415
## s2 -0.429 -0.520 -0.490 6.16 -0.227 -0.190 -0.387 0.545  0.008602 0.666 0.455
## s3 -0.529 -0.609 -0.573 7.12 -0.286 -0.175 -0.434 0.566  0.004916 0.754 0.493
## s4 -0.646 -0.708 -0.667 8.19 -0.359 -0.148 -0.482 0.573 -0.000151 0.849 0.526
## s5 -0.783 -0.819 -0.771 9.36 -0.445 -0.106 -0.530 0.561 -0.006624 0.949 0.553
##
## s0 -0.283 -0.275 -0.079 0.184 -0.359 1.11 3.06 1.65 0.0622 0.151 -0.307 -0.326
## s1 -0.322 -0.312 -0.116 0.227 -0.425 1.30 3.58 1.90 0.0639 0.194 -0.351 -0.373
## s2 -0.363 -0.351 -0.165 0.280 -0.501 1.50 4.19 2.18 0.0628 0.248 -0.400 -0.425
## s3 -0.407 -0.392 -0.229 0.344 -0.588 1.73 4.87 2.48 0.0582 0.315 -0.453 -0.480
## s4 -0.452 -0.435 -0.310 0.422 -0.685 1.99 5.64 2.81 0.0488 0.396 -0.509 -0.540
## s5 -0.497 -0.480 -0.410 0.517 -0.793 2.27 6.50 3.15 0.0336 0.493 -0.568 -0.602
##
## s0 -0.250 2.74 -0.333 1.97 4.31 -0.415 1.15 -0.284 -0.338 3.14 2.87 -0.377
## s1 -0.279 3.22 -0.390 2.27 5.04 -0.499 1.30 -0.337 -0.408 3.64 3.31 -0.446
## s2 -0.308 3.79 -0.454 2.60 5.87 -0.598 1.45 -0.398 -0.491 4.19 3.80 -0.524
## s3 -0.337 4.43 -0.527 2.95 6.80 -0.715 1.61 -0.468 -0.588 4.81 4.34 -0.613
## s4 -0.364 5.15 -0.607 3.33 7.84 -0.850 1.76 -0.548 -0.701 5.48 4.92 -0.714
## s5 -0.389 5.97 -0.695 3.72 8.98 -1.005 1.90 -0.638 -0.830 6.21 5.53 -0.826
##
## s0 -0.327 -0.0967 0.145 2.02  5.47 -0.0198 -0.141 -0.109 -0.00787 0.0953
## s1 -0.378 -0.0835 0.158 2.38  6.43 -0.0225 -0.160 -0.124 -0.00965 0.1052
## s2 -0.435 -0.0606 0.170 2.80  7.54 -0.0255 -0.180 -0.139 -0.01178 0.1149
## s3 -0.497 -0.0254 0.180 3.27  8.81 -0.0286 -0.202 -0.155 -0.01429 0.1237
## s4 -0.565  0.0243 0.187 3.81 10.23 -0.0318 -0.224 -0.171 -0.01722 0.1314
## s5 -0.638  0.0904 0.190 4.40 11.84 -0.0351 -0.247 -0.187 -0.02058 0.1376
##
## s0 -0.0827 1.86 0.243 -0.171 0.0223 -0.0277 0.153 0.247 -0.00299 -6.79e-05
## s1 -0.0925 2.14 0.270 -0.199 0.0256 -0.0332 0.173 0.283 -0.00342  5.60e-04
## s2 -0.1024 2.45 0.297 -0.232 0.0292 -0.0396 0.196 0.323 -0.00387  1.47e-03
## s3 -0.1122 2.79 0.323 -0.267 0.0332 -0.0470 0.219 0.365 -0.00436  2.71e-03
## s4 -0.1217 3.14 0.348 -0.307 0.0374 -0.0554 0.243 0.410 -0.00486  4.36e-03
## s5 -0.1305 3.51 0.370 -0.349 0.0418 -0.0648 0.267 0.457 -0.00537  6.46e-03
##
## s0 -0.0198 -0.309
## s1 -0.0225 -0.352
## s2 -0.0255 -0.399
## s3 -0.0286 -0.448
## s4 -0.0318 -0.499
## s5 -0.0351 -0.552
```

Plotting the coefficients against different lambda values.

```
plot_glmnet(ridge_model)
```



Interesting! For a low lambda value, the model exhibits `cntr.MDG`, which represents Madagascar, followed by `cntr.ZMB`, which represents Zimbabwe, as key predictor variables. This means that the model depends highly on whether the country is Madagascar or Zimbabwe, followed by some other significant countries. The other variables that previously defined the model is now undermined by the presence of the `country.code` variable and does not show high influence.

We then proceeded to evaluate the results

```
eval_results <- function(fit, true) {
    actual <- data.matrix(true)
    SSE <- sum((actual - fit)^2)
    SST <- sum((actual - mean(actual))^2)
    R_square <- 1 - SSE/SST
    data.frame(MSE = MSE(fit, true), MAE = MAE(fit,
        true), RMSE = RMSE(fit, true), MAPE = MAPE(fit,
        true), R2 = R_square)
}
```

```
fit <- predict(glm_Ridge, train.x)
true <- train.y
summary_Ridge_train <- eval_results(fit, true)
summary_Ridge_train
```

```
##    MSE  MAE RMSE MAPE    R2
## 1 5.37 1.47 2.32  Inf 0.943
```

The training dataset produced a much higher value compared to the regular multiple regression model shown above.

Fitting the model to the test dataset,

```
fit <- predict(glm_Ridge, test.x)
true <- test.y
summary_Ridge_test <- eval_results(fit, true)
summary_Ridge_test
```

```
##    MSE  MAE RMSE MAPE    R2
## 1 4.42 1.46  2.1  Inf 0.854
```

A slightly lower R^2 value, but still proves that the model is good enough.

Below is the summary of the results of the training and testing dataset.

```
summary_Ridge <- rbind(summary_Ridge_train, summary_Ridge_test)
rownames(summary_Ridge) <- c("Ridge_train", "Ridge_test")
knitr::kable(summary_Ridge, digits = 3)
```

	MSE	MAE	RMSE	MAPE	R2
Ridge_train	5.37	1.47	2.32	Inf	0.943
Ridge_test	4.42	1.46	2.10	Inf	0.854

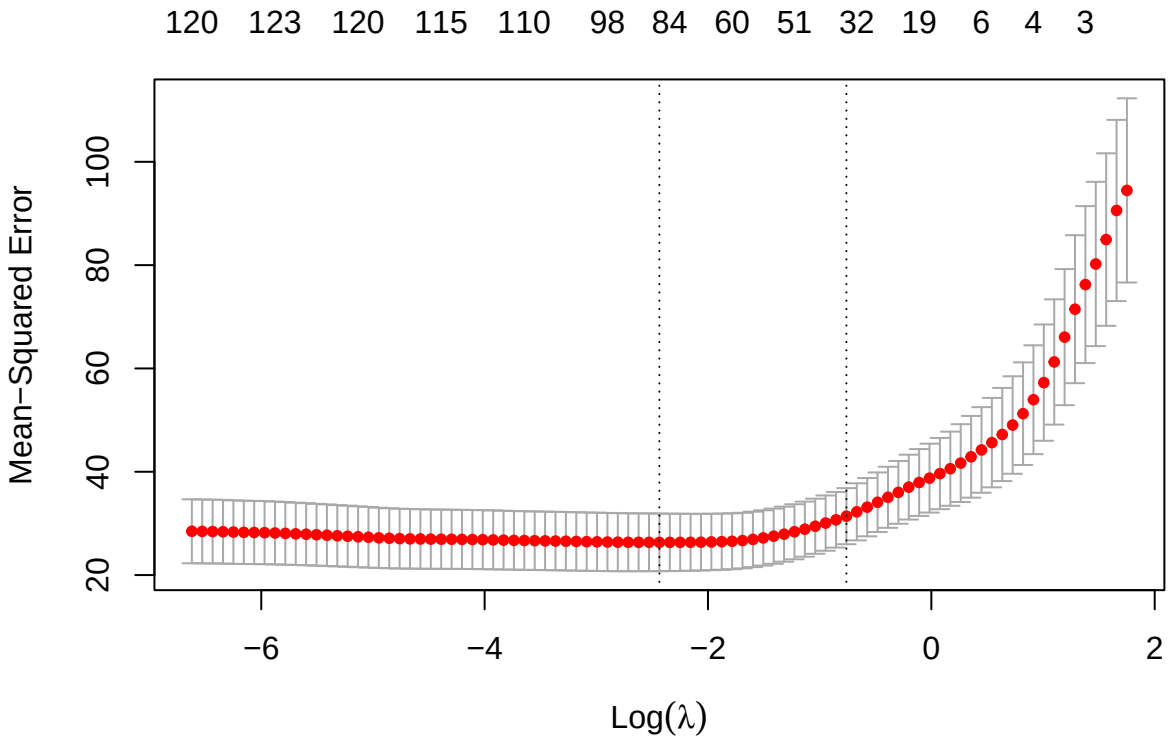
Next, we attempted LASSO regression as well. The steps are essentially similar to that of Ridge regression.

```
# Train a LASSO regression model
set.seed(123)
cv_lasso <- cv.glmnet(train.x, train.y, alpha = 1,
    type.measure = "mse")
```

```
cv_lasso
```

```
##
## Call:  cv.glmnet(x = train.x, y = train.y, type.measure = "mse", alpha = 1)
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure   SE Nonzero
## min  0.088    46    26.3 5.58      87
## 1se  0.467    28    31.4 5.43      35
```

```
plot(cv_lasso)
```




```
glm_Lasso <- glmnet(train.x, train.y, alpha = 1, lambda = cv_lasso$lambda.min)
head(t(coef(glm_Lasso)))
```

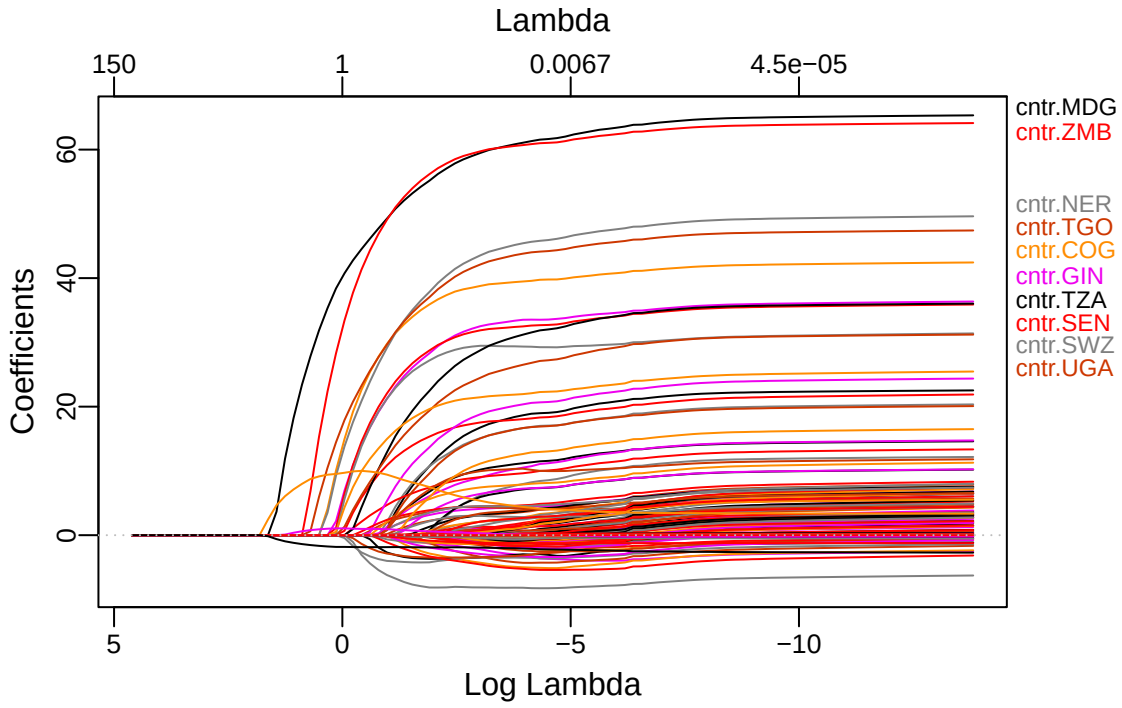
```
## 1 x 124 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 124 column names '(Intercept)', 'country.codeARG', 'country.codeARM' ... ]]

##
## s0 334 -0.186 -1.63 . . -8.04 0.257 3.91 14.4 8.92 0.288 -0.546 . 2.76 13.3
##
## s0 -0.0285 . . 7.76 16.7 37.8 3.78 . . -0.508 -0.043 . 0.0245 . 0.436 -3.59 . .
##
## s0 . . 0.0808 -0.35 4.28 31 4.01 . 1.46 6.74 . . 8.45 5.37 . . -0.762 . -2.42
##
## s0 -3.59 -1.25 16.8 -3.69 . . 0.15 . 0.668 -1.21 57.8 1.54 . . -0.0884 -0.749
##
## s0 -0.21 -1.3 41.3 . . 0.137 . 2.47 4.32 1.04 -0.338 -0.451 -0.535 4.25 -2.44
##
## s0 9.04 30.7 9.08 . 2.88 -0.195 . 0.0872 28.8 . 13 40.1 -3.46 2.34 -0.605
##
## s0 -0.878 26.2 22.5 -3.94 -0.723 0.778 1.03 20.9 58.4 -0.156 -0.299 . -0.135 .
##
## s0 . 6.27 . -1.06 0.0597 . 0.138 0.445 -0.000445 . -5.56e-05 -1.89
```

It is evident that the number of variables included in the model is less than that of ridge regression. This is in accordance with the nature of the LASSO regression model that reduces the coefficients of certain predictor variables to zero for model interpretability.

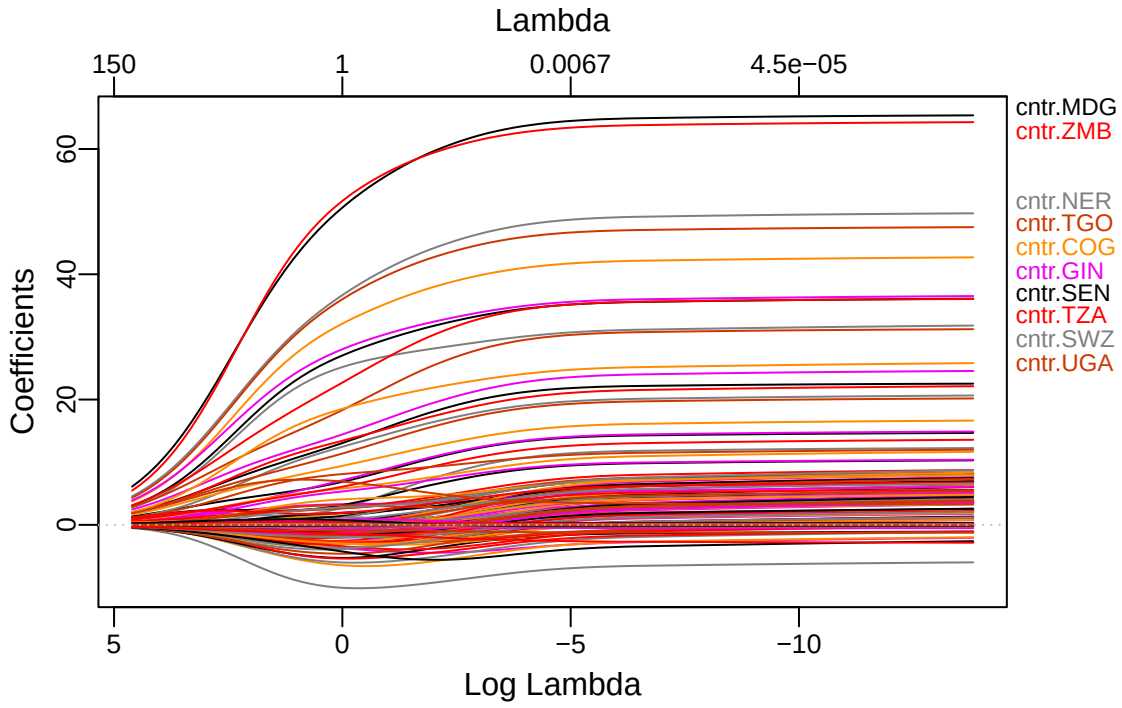
```r
lambda <- 10^seq(-6, 2, length = 100)
lasso_model = glmnet(train.x, train.y, alpha = 1, lambda = lambda)

head(t(coef(lasso_model)))
```

```
## 6 x 124 sparse Matrix of class "dgCMatrix"

##    [[ suppressing 124 column names '(Intercept)', 'country.codeARG', 'country.codeARM' ... ]]

##
## s0 4.48 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s1 4.48 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s2 4.48 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s3 4.48 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s4 4.48 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s5 4.48 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
##
## s0 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s3 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s4 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s5 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
##
## s0 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s1 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s3 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s4 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
## s5 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
##
## s0 . . . . . . . . . . .
## s1 . . . . . . . . . . .
## s2 . . . . . . . . . . .
## s3 . . . . . . . . . . .
## s4 . . . . . . . . . . .
## s5 . . . . . . . . . . .
```

```r
plot_glmnet(lasso_model)
```



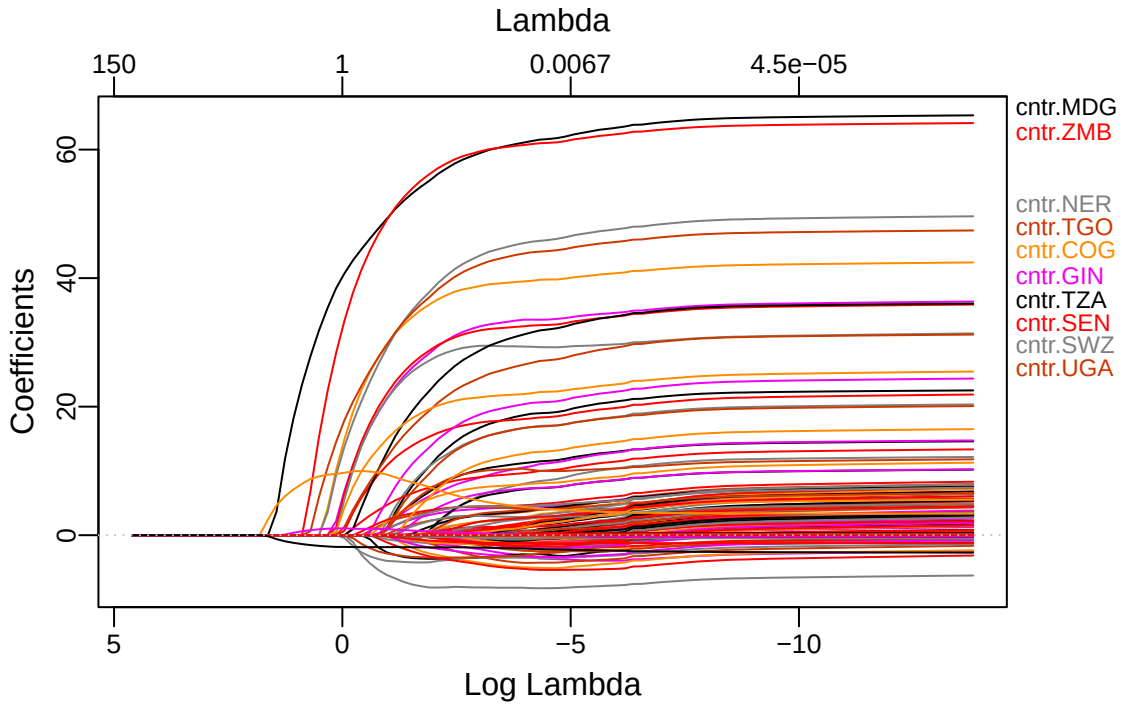

```
plot_glmnet(ridge_model)
```




```
fit2 <- predict(glm_Lasso, train.x)
true2 <- train.y
summary_Lasso_train <- eval_results(fit2, true2)
summary_Lasso_train
```

```
##    MSE  MAE RMSE MAPE    R2
## 1 4.94 1.48 2.22  Inf 0.948
```

```
plot_glmnet(lasso_model)
```




```
fit3 <- predict(glm_Lasso, test.x)
true3 <- test.y
summary_Lasso_test <- eval_results(fit3, true3)
summary_Lasso_test
```

```
##    MSE MAE RMSE MAPE    R2
## 1 4.67 1.5 2.16  Inf 0.846
```

```
summary_lasso <- rbind(summary_Lasso_train, summary_Lasso_test)
rownames(summary_lasso) <- c("LASSO_train", "LASSO_test")
knitr::kable(summary_lasso, digits = 3)
```

	MSE	MAE	RMSE	MAPE	R2
LASSO_train	4.94	1.48	2.22	Inf	0.948
LASSO_test	4.67	1.50	2.16	Inf	0.846

3.2.2. Assessment

This section recapitulates the three models that we have generated so far through the method of gradually removing variables from the original dataset.

```
summary(lm_4)
```

```
##
## Call:
## lm(formula = pov ~ . - hlth - lgdp.dflt - lfdi, data = train_1)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -24.36  -2.61  -0.36   1.99  50.76
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    186.36637   89.75554    2.08  0.03825 *
## edu.total       -0.73952    0.18026   -4.10  4.6e-05 ***
## gcf             -0.11112    0.03701   -3.00  0.00278 **
## incomeL          7.31879    2.33605    3.13  0.00181 **
## incomeLM        -3.33101    1.46529   -2.27  0.02333 *
## incomeUM        -3.57242    0.97541   -3.66  0.00027 ***
## lbr.part         0.21730    0.03280    6.62  7.3e-11 ***
## mil             -0.93100    0.21016   -4.43  1.1e-05 ***
## pop.gwth.rural   0.57354    0.19201    2.99  0.00292 **
## pop.gwth.urban   1.90680    0.20117    9.48  < 2e-16 ***
## trade           -0.01696    0.00533   -3.18  0.00153 **
## unemp            0.30422    0.05783    5.26  1.9e-07 ***
## year            -0.07992    0.04469   -1.79  0.07416 .
## lgdp.pc         -3.13114    0.54157   -5.78  1.1e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.86 on 654 degrees of freedom
## Multiple R-squared:  0.586,  Adjusted R-squared:  0.578
## F-statistic: 71.3 on 13 and 654 DF,  p-value: <2e-16
```

Our first model, `lm_4`, generated with regular multiple linear regression exhibits the lowest R^2 value. The model shows that `incomeL` is the most significant predictor variable with the highest coefficient. One interesting observation is that `incomeLM` and `incomeUM` has a negative correlation with the pov level, in contrast to `incomeL`. This shows that countries that have lower income tend to have high `pov` value. On the other hand, countries that are classified to have lower middle income and upper middle income seem to have lower `pov` values, as shown by their negative coefficients in the regression model.

Apart from income, some other variables that contribute more significantly to the model are `lgdp.pc`, with a coefficient of -3.13114, exhibiting a stronger negative correlation, and `pop.gwth.urban`, exhibiting a stronger positive correlation with a coefficient of 1.90680. This is counter-intuitive, as the notion is that population growth in cities will help drive positive economic change due to the number of productive people living in the city. The model also shows that `pop.gwth.rural` gives a less significant impact on `pov` as compared to `pop.gwth.rural`.

```
knitr::kable(summary_lasso, digits = 3)
```

	MSE	MAE	RMSE	MAPE	R2
LASSO_train	4.94	1.48	2.22	Inf	0.948

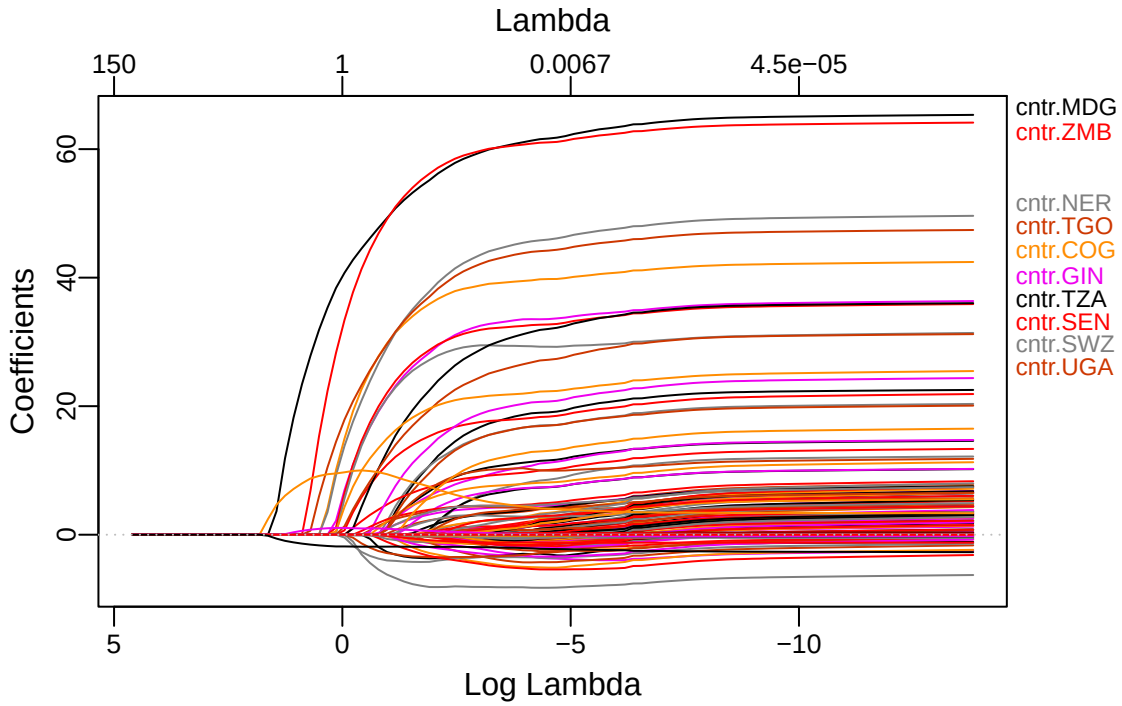
	MSE	MAE	RMSE	MAPE	R2
LASSO_test	4.67	1.50	2.16	Inf	0.846

```
knitr::kable(summary_Ridge, digits = 3)
```

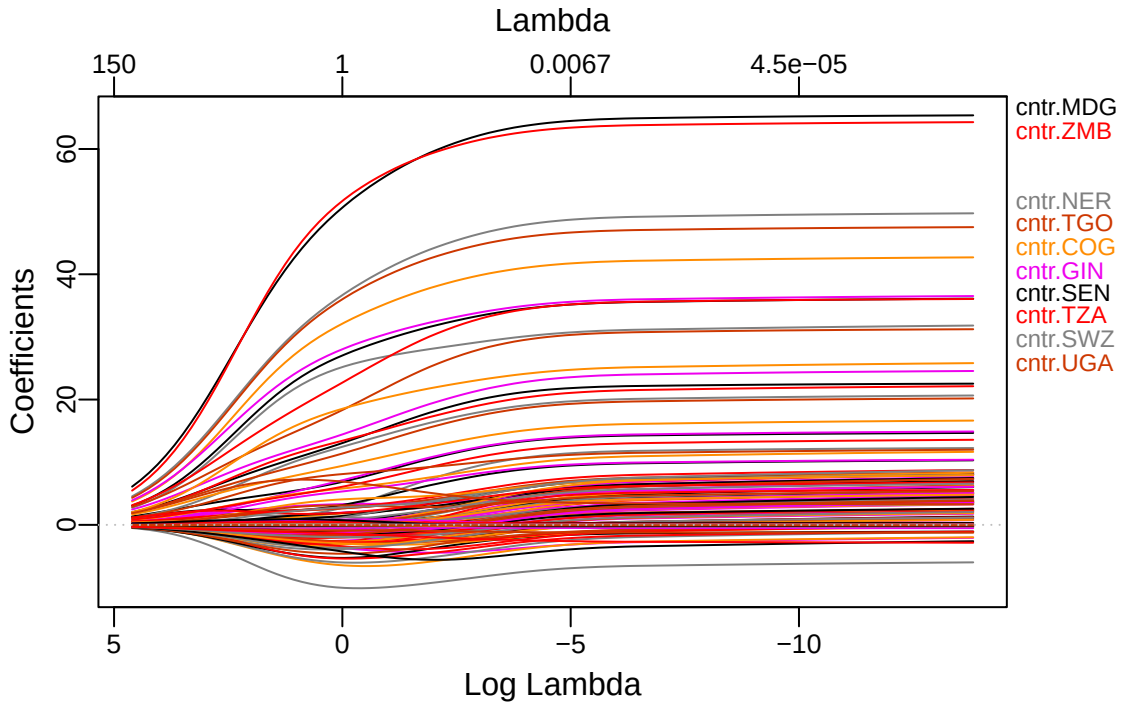
	MSE	MAE	RMSE	MAPE	R2
Ridge_train	5.37	1.47	2.32	Inf	0.943
Ridge_test	4.42	1.46	2.10	Inf	0.854

The MSE for the LASSO regression model is slightly higher than that of the Ridge regression model. This to be expected, as the Ridge regression model aims for an improved accuracy while the LASSO model aims for an improved interpretability. Both models seem to exhibit very high R^2 value for both the training and testing dataset.

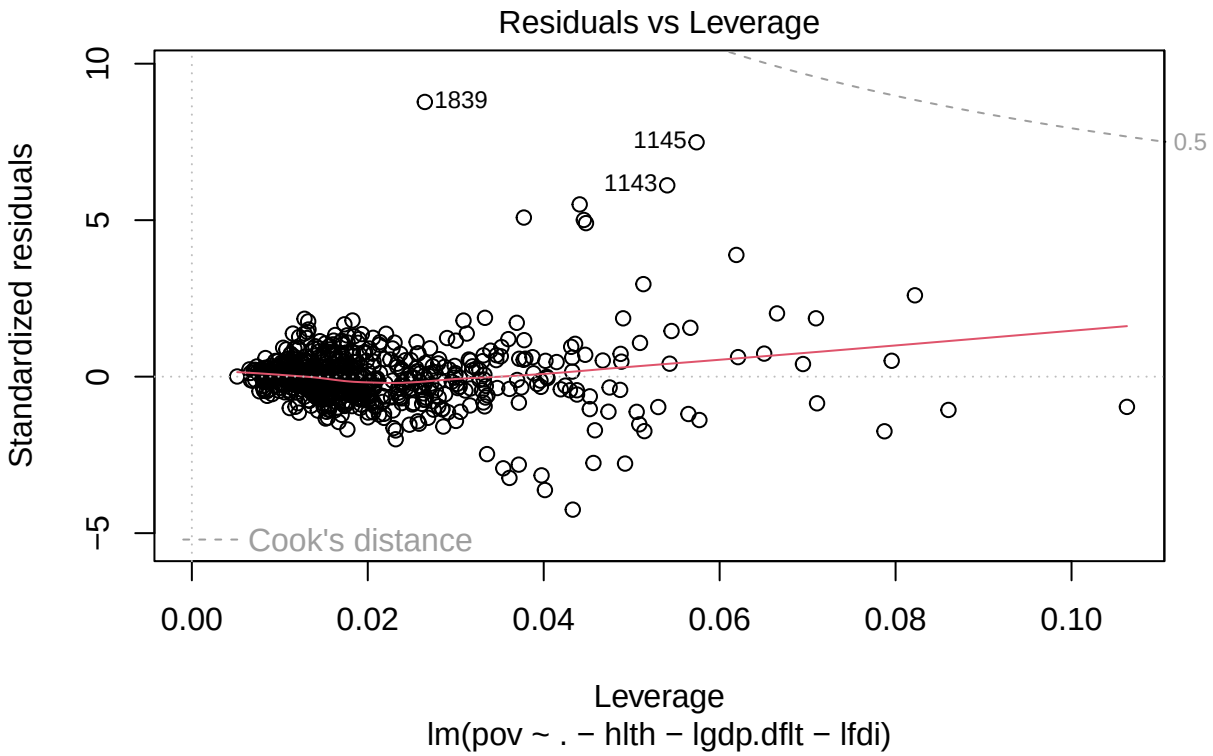
```
plot_glmnet(lasso_model)
```




```
plot_glmnet(ridge_model)
```




```
plot(lm_4, which = 5)
```



The `Residual vs Leverage` plot of `lm_4` suggests that heteroskedasticity might be present in the model. This means that the variance of the residuals is unequal over a range of observed values. This problem exists in the dataset rather than the models generated.

3.3. Forward variable selection

3.3.1. Model Fitting

```
df_count <- read.csv("../data/countries1.csv")
set.seed(10)
index <- sort(sample(x = nrow(df_count), size = nrow(df_count) *
    0.8))
train <- df_count[index, ]
test <- df_count[-index, ]
```

Through forward selection, we will do feature selection. We will stop the forward selection when the adjusted r-value>0.75.

Through iteration of different models. A linear model consisting of income as a predictor has the highest adjusted r-square value=0.6201, thus it is selected.

```
summary(lm(formula = pov ~ income, data = train))  #0.6201
```

```
##
```

```
## Call:
## lm(formula = pov ~ income, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -41.90  -3.76  -0.25   0.75  74.84
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.352      0.498    0.71     0.48
## incomeL       42.846      0.917   46.70  < 2e-16 ***
## incomeLM      10.204      0.744   13.71  < 2e-16 ***
## incomeUM       3.810      0.774    4.92  9.6e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11 on 1422 degrees of freedom
##   (48 observations deleted due to missingness)
## Multiple R-squared:  0.621,  Adjusted R-squared:  0.62
## F-statistic:  776 on 3 and 1422 DF,  p-value: <2e-16
```

Since income as a predictor brought the highest adjusted r-value, we add another predictor to income to find the predictor added to income which produced the highest adjusted r-value. reg as a predictor has the highest adjusted r-value=0.7181

```
summary(lm(formula = pov ~ reg + income, data = train))  #0.7181
```

```
##
## Call:
## lm(formula = pov ~ reg + income, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -39.80  -3.82   0.09   1.41  55.84
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                       4.794      1.005    4.77  2.0e-06 ***
## regEurope & Central Asia         -4.787      0.957   -5.00  6.4e-07 ***
## regLatin America & Caribbean      0.130      0.987    0.13    0.895
## regMiddle East & North Africa    -6.309      1.346   -4.69  3.0e-06 ***
## regNorth America                 -4.112      1.929   -2.13    0.033 *
## regSouth Asia                     0.608      1.800    0.34    0.736
## regSub-Saharan Africa            18.682      1.218   15.34  < 2e-16 ***
## incomeL                          28.128      1.070   26.29  < 2e-16 ***
## incomeLM                          6.081      0.757    8.03  2.0e-15 ***
## incomeUM                          0.600      0.755    0.79    0.427
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.46 on 1416 degrees of freedom
##   (48 observations deleted due to missingness)
## Multiple R-squared:  0.72,   Adjusted R-squared:  0.718
## F-statistic:  404 on 9 and 1416 DF,  p-value: <2e-16
```

The forward selection process continues and lgdp.pc is the new predictor added that produced highest adjusted r-square=0.7492

```
summary(lm(formula = pov ~ lgdp.pc + income + reg,
    data = train))  #0.7492
```

```
##
## Call:
## lm(formula = pov ~ lgdp.pc + income + reg, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -36.22  -3.73  -0.27   2.95  54.42
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                      70.830      5.106   13.87  < 2e-16 ***
## lgdp.pc                          -6.439      0.489  -13.16  < 2e-16 ***
## incomeL                           2.658      2.185    1.22     0.22
## incomeLM                        -10.440      1.444   -7.23  7.8e-13 ***
## incomeUM                         -8.838      1.009   -8.76  < 2e-16 ***
## regEurope & Central Asia         -3.941      0.903   -4.37  1.4e-05 ***
## regLatin America & Caribbean      0.178      0.929    0.19     0.85
## regMiddle East & North Africa    -6.795      1.271   -5.35  1.0e-07 ***
## regNorth America                 -1.918      1.822   -1.05     0.29
## regSouth Asia                    -0.544      1.695   -0.32     0.75
## regSub-Saharan Africa            17.945      1.149   15.62  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.89 on 1412 degrees of freedom
##   (51 observations deleted due to missingness)
## Multiple R-squared:  0.751,  Adjusted R-squared:  0.749
## F-statistic:  426 on 10 and 1412 DF,  p-value: <2e-16
```

The addition of predictor gdp.pc produces the highest adjusted r-square value above 0.75, adjusted r-square=0.7703

```
summary(lm(formula = pov ~ gdp.pc + income + reg +
    lgdp.pc, data = train))  #0.7703
```

```
##
## Call:
## lm(formula = pov ~ gdp.pc + income + reg + lgdp.pc, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -34.77  -3.13   0.38   2.17  53.64
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    1.11e+02   6.01e+00   18.44  < 2e-16 ***
## gdp.pc                         2.75e-04   2.40e-05   11.42  < 2e-16 ***
```

```
## incomeL                       -6.94e+00   2.25e+00   -3.08   0.0021 **
## incomeLM                      -1.39e+01   1.41e+00   -9.82  < 2e-16 ***
## incomeUM                      -8.22e+00   9.67e-01   -8.49  < 2e-16 ***
## regEurope & Central Asia      -4.06e+00   8.64e-01   -4.70  2.9e-06 ***
## regLatin America & Caribbean   4.25e-01   8.89e-01    0.48   0.6324
## regMiddle East & North Africa -6.47e+00   1.22e+00   -5.32  1.2e-07 ***
## regNorth America              -2.34e+00   1.74e+00   -1.34   0.1793
## regSouth Asia                 -1.44e+00   1.62e+00   -0.89   0.3746
## regSub-Saharan Africa          1.73e+01   1.10e+00   15.71  < 2e-16 ***
## lgdp.pc                       -1.13e+01   6.31e-01  -17.86  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.51 on 1411 degrees of freedom
##   (51 observations deleted due to missingness)
## Multiple R-squared:  0.772,  Adjusted R-squared:  0.77
## F-statistic:  435 on 11 and 1411 DF,  p-value: <2e-16
```

Fitting of multiple linear regression with predictor gdp.pc, income, reg and lgd.pc.

```
lm1 <- lm(formula = pov ~ gdp.pc + income + reg + lgdp.pc,
    data = train)
summary(lm1)
```

```
##
## Call:
## lm(formula = pov ~ gdp.pc + income + reg + lgdp.pc, data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -34.77  -3.13   0.38   2.17  53.64
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    1.11e+02   6.01e+00   18.44  < 2e-16 ***
## gdp.pc                         2.75e-04   2.40e-05   11.42  < 2e-16 ***
## incomeL                       -6.94e+00   2.25e+00   -3.08   0.0021 **
## incomeLM                      -1.39e+01   1.41e+00   -9.82  < 2e-16 ***
## incomeUM                      -8.22e+00   9.67e-01   -8.49  < 2e-16 ***
## regEurope & Central Asia      -4.06e+00   8.64e-01   -4.70  2.9e-06 ***
## regLatin America & Caribbean   4.25e-01   8.89e-01    0.48   0.6324
## regMiddle East & North Africa -6.47e+00   1.22e+00   -5.32  1.2e-07 ***
## regNorth America              -2.34e+00   1.74e+00   -1.34   0.1793
## regSouth Asia                 -1.44e+00   1.62e+00   -0.89   0.3746
## regSub-Saharan Africa          1.73e+01   1.10e+00   15.71  < 2e-16 ***
## lgdp.pc                       -1.13e+01   6.31e-01  -17.86  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.51 on 1411 degrees of freedom
##   (51 observations deleted due to missingness)
## Multiple R-squared:  0.772,  Adjusted R-squared:  0.77
## F-statistic:  435 on 11 and 1411 DF,  p-value: <2e-16
```

3.3.2. Assessment

Assessment of linear model lm1: the predicted model has some NA values which are removed. The error rates are too high, therefore, this method of variable selection may not be suitable

```
eval.metrics.linreg <- function(actual, predicted) {
    residual <- actual - predicted
    mse <- mean(residual^2)
    mae <- mean(abs(residual))
    rmse <- sqrt(mse)
    mape <- mean(abs(residual/actual)) * 100

    data.frame(MSE = mse, MAE = mae, RMSE = rmse, MAPE = mape)
}
predicted <- predict(lm1, newdata = test)
actual <- test$pov
df23 <- na.omit(data.frame(actual, predicted))
ac <- df23$actual
pr <- df23$predicted
eval.metrics.linreg(ac, pr)
```

```
##    MSE  MAE RMSE MAPE
## 1 56.8 4.76 7.53  Inf
```

3.3.3. Interpretation

gdp per capital, income class and region of the country can provide a model to predict the poverty headcount ratio in a country. The error rate of the model is high on the testing data set, showing it may not be a good model.

3.4. Imputation

What is imputation?
Imputation is a technique to handle missing values by replacing missing data with substitute values. In our study, we focus on "item imputation", which means substituting for a component of a data point (i.e. a variable). The general idea is to take a value that preserve the property of the data (e.g., distribution, mean, standard deviation), and use other variables to predict the missing values (much like regression). Since we observed some correlations during descriptive analysis, we expect the imputation algorithm to work well.

We use **Multivariate Imputation By Chained Equations (MICE)** as primary tool for this technique, because of its wide acceptance in scientific studies (Alruhaymi and Kim (2021)). In order to use this technique, we have to assume that the "missingness" of a field can be explained by the values in other columns, (e.g., If the countries is in North America, it's more likely to be missing as we have discovered in 2.2). The general ideas of the algorithm is to fill in the missing values, and improve it iteratively until predicted values converse to a stable point. The algorithmic details of MICE is very concisely (and enthrallingly) explained in Gopalan (2020).

We use relatively small parameters in the interest of time.

- `m = 3` imputed data sets
- `maxit = 20` iterations

The imputation method is `cart` (Classification and Regression Trees).

```
set.seed(1984)
# split data, make sure all the coutries
seeds <- countries1 %>%
    group_by(country.code) %>%
    filter(row_number() == 1)

plants <- countries1 %>%
    group_by(country.code) %>%
    filter(row_number() != 1)

isComplete <- which(complete.cases(plants))
idx <- sample(isComplete, replace = F, 0.2 * nrow(plants))

countries.train.4 <- plants[-idx, ] %>%
    rbind(seeds)
countries.test.4 <- plants[idx, ]

meth <- c(rep("", 4), "cart", "", rep("cart", 16))
names(meth) <- colnames(countries1)
meth
```

```
##   country.code   country.name           year            pov         income
##             ""             ""             ""             ""         "cart"
##            reg      edu.total           hlth            mil            fdi
##             ""         "cart"         "cart"         "cart"         "cart"
##       lbr.part          unemp pop.gwth.total pop.gwth.rural pop.gwth.urban
##         "cart"         "cart"         "cart"         "cart"         "cart"
##       gdp.dflt            gcf          trade         gdp.pc           lfdi
##         "cart"         "cart"         "cart"         "cart"         "cart"
##      lgdp.dflt        lgdp.pc
##         "cart"         "cart"
```

```
# run the algorithm, or read from the file we
# generated as this is a very time-consuming
# process countries1.imputed <-
# mice(countries.train.4, m = 3, maxit = 20,
# method = meth) saveRDS(countries1.imputed,
# 'countries1.imputed.RData')
countries1.imputed <- readRDS("countries1.imputed.RData")
summary(countries1.imputed)
```

```
## Class: mids
## Number of multiple imputations:  3
## Imputation methods:
##   country.code   country.name           year            pov         income
##             ""             ""             ""             ""         "cart"
##            reg      edu.total           hlth            mil            fdi
##             ""         "cart"         "cart"         "cart"         "cart"
##       lbr.part          unemp pop.gwth.total pop.gwth.rural pop.gwth.urban
##         "cart"         "cart"             ""         "cart"         "cart"
##       gdp.dflt            gcf          trade         gdp.pc           lfdi
```

```
##         "cart"         "cart"         "cart"         "cart"         "cart"
##      lgdp.dflt        lgdp.pc
##         "cart"         "cart"
## PredictorMatrix:
##              country.code country.name year pov income reg edu.total hlth mil
## country.code            0            0    1   1      1   1         1    1   1
## country.name            1            0    1   1      1   1         1    1   1
## year                    1            0    0   1      1   1         1    1   1
## pov                     1            0    1   0      1   1         1    1   1
## income                  1            0    1   1      0   1         1    1   1
## reg                     1            0    1   1      1   0         1    1   1
##              fdi lbr.part unemp pop.gwth.total pop.gwth.rural pop.gwth.urban
## country.code   1        1     1              1              1              1
## country.name   1        1     1              1              1              1
## year           1        1     1              1              1              1
## pov            1        1     1              1              1              1
## income         1        1     1              1              1              1
## reg            1        1     1              1              1              1
##              gdp.dflt gcf trade gdp.pc lfdi lgdp.dflt lgdp.pc
## country.code        1   1     1      1    1         1       1
## country.name        1   1     1      1    1         1       1
## year                1   1     1      1    1         1       1
## pov                 1   1     1      1    1         1       1
## income              1   1     1      1    1         1       1
## reg                 1   1     1      1    1         1       1
## Number of logged events:  961
##   it im       dep     meth
## 1  0  0           constant
## 2  1  1    income     cart
## 3  1  1 edu.total     cart
## 4  1  1      hlth     cart
## 5  1  1       mil     cart
## 6  1  1       fdi     cart
##
## 1
## 2
## 3 country.codeARE, country.codeBIH, country.codeCOD, country.codeDZA, country.codeEAS, country.codeEC
## 4
## 5          country.codeBTN, country.codeCOM, country.codeDJI, country.codeEAS, country.codeECS, count
## 6
```

We can take a look into one of the imputed data sets.

```
countries1.imputed.1 <- complete(countries1.imputed,
    1)
summary(countries1.imputed.1)
```

```
##   country.code  country.name           year           pov       income
##  BRA    :  32   Length:1508        Min.   :1967   Min.   : 0.0   H :484
##  USA    :  30   Class :character   1st Qu.:1999   1st Qu.: 0.3   L :247
##  HND    :  26   Mode  :character   Median :2007   Median : 1.9   LM:429
##  IDN    :  24                      Mean   :2006   Mean   :11.3   UM:348
##  ARG    :  23                      3rd Qu.:2014   3rd Qu.:13.6
```

```
##  CRI    :  23                      Max.   :2021   Max.   :91.5
##  (Other):1350
##                           reg        edu.total          hlth
##  East Asia & Pacific       :145   Min.   : 1.03   Min.   : 1.72
##  Europe & Central Asia     :664   1st Qu.: 3.41   1st Qu.: 4.90
##  Latin America & Caribbean :322   Median : 4.41   Median : 6.72
##  Middle East & North Africa: 90   Mean   : 4.51   Mean   : 6.78
##  North America             : 46   3rd Qu.: 5.43   3rd Qu.: 8.37
##  South Asia                : 46   Max.   :15.75   Max.   :17.73
##  Sub-Saharan Africa        :195
##       mil             fdi               lbr.part        unemp
##  Min.   : 0.00   Min.   :-3.44e+11   Min.   :30.5   Min.   : 0.2
##  1st Qu.: 1.05   1st Qu.: 2.11e+08   1st Qu.:56.0   1st Qu.: 4.2
##  Median : 1.49   Median : 1.28e+09   Median :61.4   Median : 6.7
##  Mean   : 1.83   Mean   : 1.45e+10   Mean   :61.3   Mean   : 8.2
##  3rd Qu.: 2.14   3rd Qu.: 8.06e+09   3rd Qu.:66.2   3rd Qu.:10.3
##  Max.   :19.38   Max.   : 7.34e+11   Max.   :93.0   Max.   :49.7
##
##  pop.gwth.total  pop.gwth.rural  pop.gwth.urban     gdp.dflt         gcf
##  Min.   :-3.63   Min.   :-8.56   Min.   :-4.08   Min.   : -26   Min.   : 0.0
##  1st Qu.: 0.31   1st Qu.:-0.79   1st Qu.: 0.55   1st Qu.:   2   1st Qu.:19.6
##  Median : 1.10   Median : 0.03   Median : 1.60   Median :   4   Median :22.7
##  Mean   : 1.13   Mean   : 0.09   Mean   : 1.79   Mean   :  21   Mean   :23.8
##  3rd Qu.: 1.90   3rd Qu.: 1.05   3rd Qu.: 2.85   3rd Qu.:   9   3rd Qu.:26.8
##  Max.   : 5.61   Max.   : 4.60   Max.   :13.80   Max.   :3334   Max.   :69.5
##
##      trade          gdp.pc             lfdi         lgdp.dflt        lgdp.pc
##  Min.   :  1   Min.   :   120   Min.   : 6.91   Min.   :-2.92   Min.   : 4.78
##  1st Qu.: 50   1st Qu.:  1579   1st Qu.:19.17   1st Qu.: 0.55   1st Qu.: 7.36
##  Median : 71   Median :  5176   Median :20.97   Median : 1.38   Median : 8.55
##  Mean   : 83   Mean   : 14378   Mean   :20.39   Mean   : 1.25   Mean   : 8.56
##  3rd Qu.:104   3rd Qu.: 19598   3rd Qu.:22.80   3rd Qu.: 2.22   3rd Qu.: 9.88
##  Max.   :380   Max.   :123679   Max.   :27.32   Max.   : 8.11   Max.   :11.73
##
```

```
sum(!complete.cases(countries1.imputed.1))
```

```
## [1] 0
```

We found no missing cases as expected.

3.4.1. Model Fitting

A. Ordinary Linear Regression We generated 3 sets of imputed (training) data. We can build separate models using each data set, and combine the estimates using pooling rule. With all the generated data sets.

```
formula.str <- "pov ~ country.code+year+income+edu.total+hlth+mil+fdi+lbr.part+unemp+pop.gwth.total+pop

# Using single imputed data set
fit1 <- lapply(1:countries1.imputed$m, function(i) lm(as.formula(formula.str),
    data = complete(countries1.imputed, i)))
```

```
# Build models with all generated data set and
# pool the estimates
fit2 <- with(data = countries1.imputed, exp = lm(as.formula(formula.str)))
fit2.combined <- pool(fit2)
# dummy lm model
fit2.dummy <- lm(as.formula(formula.str), data = complete(countries1.imputed,
    1))
# replace coefficients of dummy model
name.coef <- names(fit2.dummy$coefficients)
fit2.dummy$coefficients <- fit2.combined$pooled$estimate
names(fit2.dummy$coefficients) <- name.coef
```

Helper function to calculate R-Squared

```
r2 <- function(pred, orig) {
    RSS <- sum((pred - orig)^2)
    TSS <- sum((orig - mean(orig))^2)
    R2 <- 1 - RSS/TSS
    return(R2)
}

adjR2 <- function(pred, orig, k) {
    R2 <- r2(pred, orig)
    n <- length(pred)
    adjr2 <- 1 - (1 - R2) * (n - 1)/(n - k - 1)
    return(adjr2)
}
```

R-Squared on train and test

```
fit1.summ <- lapply(fit1, function(f) {
    # number of variables
    k <- length(f$coefficients) - 1
    # predict value of test
    pred <- predict(f, countries.test.4)
    test.resid <- pred - countries.test.4$pov
    data.frame(`Train MSE` = mean(summary(f)$residuals^2),
        `Test MSE` = mean(test.resid^2), `Train MAE` = mean(abs(summary(f)$residuals)),
        `Test MAE` = mean(abs(test.resid)), `Train R2` = summary(f)$r.squared,
        `Test R2` = r2(pred, countries.test.4$pov))
})

# number of variables
k <- length(fit2.dummy$coefficients) - 1
# number of variables
fit2.pred <- predict(fit2.dummy, countries.test.4)
test.resid <- fit2.pred - countries.test.4$pov
fit2.summ <- data.frame(`Train MSE` = mean(summary(fit2.dummy)$residuals^2),
    `Test MSE` = mean(test.resid^2), `Train MAE` = mean(abs(summary(fit2.dummy)$residuals)),
    `Test MAE` = mean(abs(test.resid)), `Train R2` = pool.r.squared(fit2)[,
        "est"], `Test R2` = r2(fit2.pred, countries.test.4$pov))

res <- do.call(rbind.data.frame, fit1.summ)
```

```
res <- rbind(res, fit2.summ)
res <- cbind(data.frame(Set = c(1:3, "Pooled")), res)

kable(res)
```

Set	Train.MSE	Test.MSE	Train.MAE	Test.MAE	Train.R2	Test.R2
1	29.3	19.3	3.36	2.46	0.916	0.772
2	27.7	20.2	3.29	2.50	0.920	0.762
3	28.8	19.7	3.33	2.45	0.917	0.768
Pooled	29.3	19.6	3.36	2.45	0.918	0.769

Pooled model slightly improve performance. Test data fitting has better MSE and MAE than train data. However, R-Squared is lower in test data. There might be some over-fitting in our models.

```
# find coef of variable other than country.code
var <- c("year", "incomeL", "incomeLM", "incomeUM",
    "edu.total", "hlth", "mil", "fdi", "lbr.part",
    "unemp", "pop.gwth.total", "pop.gwth.rural", "pop.gwth.urban",
    "gdp.dflt", "gcf", "trade", "gdp.pc", "lfdi", "lgdp.dflt",
    "lgdp.pc")

res <- do.call(rbind, lapply(fit1, function(f) f$coefficients[var]))
res <- rbind(res, fit2.dummy$coefficients[var])
res <- cbind(data.frame(Set = c(1:3, "Pooled")), res)
kable(res)
```

Set	year	incomeL	incomeLM	incomeUM	edu.total	hlth	mil	fdi	lbr.part	unemp	pop.gwth.total	pop.gwth.rural	pop.gwth.urban	gdp.dflt	gcf	trade	gdp.pc	lfdi	lgdp.dflt	lgdp.pc
1	-0.235	0.381	-5.25	-2.97	-0.420	0.248	-0.460	0	0.169	-0.099	-3.12	0.828	2.37	0.000	-0.237	-0.016	0	0.000	0.191	-6.19
2	-0.219	1.086	-4.87	-2.50	-0.319	0.416	-0.099	0	0.287	0.010	-3.02	0.789	2.14	0.001	-0.220	0.002	0	-0.022	0.116	-6.78
3	-0.176	-0.818	-6.10	-3.27	-0.677	0.305	-0.084	0	0.158	-0.012	-3.00	0.843	2.31	0.000	-0.208	-0.013	0	-0.001	0.035	-7.55
Pooled	-0.210	0.216	-5.41	-2.91	-0.472	0.323	-0.214	0	0.205	-0.034	-3.04	0.820	2.27	0.000	-0.222	-0.009	0	-0.008	0.114	-6.84

In contradiction to expectation, `hlth` is positively correlated with `pov`, and lower-middle income countries (incomeLM) are less poor than high income countries (incomeH). `pop.gwth.total` is negatively related to `pov`, while `pop.gwth.rural` and `pop.gwth.urban` are positively related. They might be indications of over-fitting model.

```
# fit model on imputed set no. 1
summary(fit1[[1]])
```

```
##
## Call:
## lm(formula = as.formula(formula.str), data = complete(countries1.imputed,
##     i))
##
```

```
## Residuals:
##    Min     1Q Median     3Q    Max
## -25.59  -2.00   0.05   1.92  35.96
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.41e+02   7.00e+01    7.73  2.1e-14 ***
## country.codeALB -1.75e+01   4.23e+00   -4.14  3.8e-05 ***
## country.codeARE -1.32e+01   5.78e+00   -2.28  0.02274 *
## country.codeARG -1.62e+01   3.98e+00   -4.06  5.2e-05 ***
## country.codeARM -1.50e+01   4.00e+00   -3.75  0.00018 ***
## country.codeAUS -1.79e+01   4.40e+00   -4.06  5.2e-05 ***
## country.codeAUT -1.75e+01   4.39e+00   -3.97  7.5e-05 ***
## country.codeAZE -2.56e+01   4.89e+00   -5.23  2.0e-07 ***
## country.codeBDI  1.98e+01   4.78e+00    4.14  3.7e-05 ***
## country.codeBEL -1.37e+01   4.39e+00   -3.12  0.00186 **
## country.codeBEN  1.72e+01   4.88e+00    3.52  0.00044 ***
## country.codeBFA  1.34e+01   4.35e+00    3.07  0.00216 **
## country.codeBGD -1.53e+01   4.28e+00   -3.58  0.00036 ***
## country.codeBGR -1.03e+01   4.29e+00   -2.39  0.01687 *
## country.codeBIH -1.47e+01   4.77e+00   -3.08  0.00214 **
## country.codeBLR -1.22e+01   4.01e+00   -3.04  0.00237 **
## country.codeBLZ  1.48e-01   4.33e+00    0.03  0.97272
## country.codeBOL -1.33e+01   3.79e+00   -3.51  0.00046 ***
## country.codeBRA -8.06e+00   3.85e+00   -2.09  0.03677 *
## country.codeBTN -1.36e+01   4.70e+00   -2.90  0.00384 **
## country.codeBWA  3.55e+00   4.44e+00    0.80  0.42399
## country.codeCAF  2.90e+01   5.44e+00    5.32  1.2e-07 ***
## country.codeCAN -1.89e+01   4.30e+00   -4.38  1.3e-05 ***
## country.codeCHE -2.35e+01   4.68e+00   -5.03  5.7e-07 ***
## country.codeCHL -1.17e+01   4.16e+00   -2.82  0.00487 **
## country.codeCHN  4.87e-01   4.08e+00    0.12  0.90495
## country.codeCIV -1.17e+01   3.97e+00   -2.96  0.00318 **
## country.codeCMR -5.74e-01   4.86e+00   -0.12  0.90604
## country.codeCOD  3.67e+01   5.47e+00    6.72  2.7e-11 ***
## country.codeCOG  2.39e+01   5.34e+00    4.47  8.3e-06 ***
## country.codeCOL -6.76e+00   3.91e+00   -1.73  0.08438 .
## country.codeCOM -1.14e+01   5.47e+00   -2.08  0.03787 *
## country.codeCPV -5.52e+00   4.90e+00   -1.13  0.25979
## country.codeCRI -1.55e+01   3.85e+00   -4.02  6.1e-05 ***
## country.codeCYP -1.43e+01   4.27e+00   -3.35  0.00084 ***
## country.codeCZE -1.46e+01   4.29e+00   -3.41  0.00066 ***
## country.codeDEU -1.84e+01   4.39e+00   -4.18  3.1e-05 ***
## country.codeDJI  1.06e+01   4.67e+00    2.26  0.02382 *
## country.codeDNK -1.90e+01   4.52e+00   -4.19  2.9e-05 ***
## country.codeDOM -1.38e+01   3.87e+00   -3.57  0.00037 ***
## country.codeDZA -1.49e+01   4.87e+00   -3.06  0.00228 **
## country.codeECU -9.32e+00   3.80e+00   -2.45  0.01439 *
## country.codeEGY -1.84e+01   4.11e+00   -4.48  8.1e-06 ***
## country.codeESP -1.49e+01   4.25e+00   -3.50  0.00048 ***
## country.codeEST -1.14e+01   4.32e+00   -2.65  0.00819 **
## country.codeETH -4.64e+00   4.49e+00   -1.03  0.30097
## country.codeFIN -1.72e+01   4.45e+00   -3.87  0.00011 ***
## country.codeFJI -1.38e+01   4.65e+00   -2.96  0.00310 **
```

```
## country.codeFRA -1.68e+01   4.32e+00   -3.89  0.00011 ***
## country.codeFSM -6.23e+00   5.64e+00   -1.11  0.26904
## country.codeGAB -8.68e+00   5.43e+00   -1.60  0.11001
## country.codeGBR -1.92e+01   4.30e+00   -4.46  8.8e-06 ***
## country.codeGEO -9.45e+00   4.04e+00   -2.34  0.01947 *
## country.codeGHA  1.16e+01   4.16e+00    2.78  0.00548 **
## country.codeGIN  5.83e+00   4.23e+00    1.38  0.16806
## country.codeGMB  2.33e+00   4.59e+00    0.51  0.61210
## country.codeGNB  1.12e+00   4.41e+00    0.25  0.79879
## country.codeGRC -1.36e+01   4.24e+00   -3.22  0.00132 **
## country.codeGTM -7.40e+00   4.25e+00   -1.74  0.08193 .
## country.codeGUY -2.52e+00   5.76e+00   -0.44  0.66194
## country.codeHND -4.52e+00   3.68e+00   -1.23  0.22001
## country.codeHRV -1.26e+01   4.38e+00   -2.88  0.00409 **
## country.codeHTI -1.24e+00   6.82e+00   -0.18  0.85533
## country.codeHUN -1.21e+01   4.22e+00   -2.86  0.00429 **
## country.codeIDN  3.78e+00   3.75e+00    1.01  0.31394
## country.codeIND  3.87e+00   3.97e+00    0.98  0.32954
## country.codeIRL -1.78e+01   4.38e+00   -4.06  5.1e-05 ***
## country.codeIRN -1.14e+01   3.94e+00   -2.90  0.00384 **
## country.codeIRQ -1.48e+01   5.39e+00   -2.75  0.00608 **
## country.codeISL -2.18e+01   4.44e+00   -4.91  1.1e-06 ***
## country.codeISR -1.53e+01   4.18e+00   -3.66  0.00026 ***
## country.codeITA -1.58e+01   4.31e+00   -3.67  0.00025 ***
## country.codeJAM -1.91e+01   4.49e+00   -4.24  2.4e-05 ***
## country.codeJOR -1.45e+01   4.30e+00   -3.36  0.00081 ***
## country.codeJPN -1.46e+01   5.62e+00   -2.59  0.00958 **
## country.codeKAZ -1.44e+01   3.96e+00   -3.64  0.00028 ***
## country.codeKEN -9.01e+00   4.40e+00   -2.05  0.04087 *
## country.codeKGZ -1.71e+01   3.84e+00   -4.44  9.7e-06 ***
## country.codeKIR -1.33e+01   5.52e+00   -2.41  0.01594 *
## country.codeKOR -1.27e+01   4.72e+00   -2.68  0.00743 **
## country.codeLAO -1.58e+01   4.27e+00   -3.69  0.00023 ***
## country.codeLBN -1.47e+01   6.82e+00   -2.15  0.03135 *
## country.codeLBR -2.63e+00   4.95e+00   -0.53  0.59500
## country.codeLCA  6.94e+00   5.59e+00    1.24  0.21449
## country.codeLKA -1.77e+01   4.27e+00   -4.16  3.4e-05 ***
## country.codeLSO  1.29e+01   4.70e+00    2.75  0.00599 **
## country.codeLTU -1.21e+01   4.38e+00   -2.76  0.00590 **
## country.codeLUX -2.37e+01   4.98e+00   -4.75  2.2e-06 ***
## country.codeLVA -1.09e+01   4.28e+00   -2.55  0.01103 *
## country.codeMAR -1.46e+01   4.27e+00   -3.42  0.00065 ***
## country.codeMDA -1.23e+01   4.08e+00   -3.01  0.00266 **
## country.codeMDG  2.63e+01   4.18e+00    6.29  4.2e-10 ***
## country.codeMDV -1.10e+01   4.63e+00   -2.37  0.01805 *
## country.codeMEX -1.05e+01   4.02e+00   -2.62  0.00879 **
## country.codeMHL -3.87e+00   7.23e+00   -0.53  0.59284
## country.codeMKD -5.39e+00   4.18e+00   -1.29  0.19800
## country.codeMLI  1.45e+01   4.61e+00    3.15  0.00166 **
## country.codeMLT -1.06e+01   4.64e+00   -2.29  0.02228 *
## country.codeMMR -1.55e+01   6.85e+00   -2.26  0.02368 *
## country.codeMNE -9.52e+00   4.33e+00   -2.20  0.02790 *
## country.codeMNG -1.17e+01   4.08e+00   -2.87  0.00416 **
## country.codeMOZ  3.65e+01   4.61e+00    7.91  5.2e-15 ***
```

```
## country.codeMRT -9.49e+00   4.07e+00   -2.33  0.01987 *
## country.codeMUS -1.36e+01   5.56e+00   -2.45  0.01431 *
## country.codeMWI  2.07e+01   4.53e+00    4.57  5.3e-06 ***
## country.codeMYS -1.76e+01   4.11e+00   -4.29  1.9e-05 ***
## country.codeNAM  6.98e+00   4.98e+00    1.40  0.16092
## country.codeNER  3.24e+01   4.33e+00    7.48  1.4e-13 ***
## country.codeNGA  9.25e+00   4.10e+00    2.26  0.02414 *
## country.codeNIC -1.15e+01   4.25e+00   -2.70  0.00712 **
## country.codeNLD -1.58e+01   4.53e+00   -3.48  0.00053 ***
## country.codeNOR -2.27e+01   4.62e+00   -4.91  1.0e-06 ***
## country.codeNPL -1.18e+01   5.03e+00   -2.35  0.01871 *
## country.codeNRU -1.16e+01   6.90e+00   -1.68  0.09363 .
## country.codePAK -5.89e+00   3.94e+00   -1.50  0.13490
## country.codePAN -6.63e+00   3.88e+00   -1.71  0.08777 .
## country.codePER -9.83e+00   3.92e+00   -2.51  0.01230 *
## country.codePHL -1.05e+01   4.38e+00   -2.41  0.01626 *
## country.codePNG  2.29e+01   5.46e+00    4.21  2.8e-05 ***
## country.codePOL -1.32e+01   4.35e+00   -3.04  0.00244 **
## country.codePRT -1.66e+01   4.30e+00   -3.86  0.00012 ***
## country.codePRY -1.69e+01   3.77e+00   -4.47  8.4e-06 ***
## country.codePSE -1.44e+01   4.10e+00   -3.52  0.00045 ***
## country.codeROU -7.85e+00   4.22e+00   -1.86  0.06300 .
## country.codeRUS -1.58e+01   3.97e+00   -3.97  7.6e-05 ***
## country.codeRWA  2.03e+01   4.44e+00    4.57  5.3e-06 ***
## country.codeSDN -2.42e+00   5.41e+00   -0.45  0.65496
## country.codeSEN  1.45e+01   4.24e+00    3.42  0.00066 ***
## country.codeSLB  4.26e+00   5.43e+00    0.78  0.43288
## country.codeSLE  6.81e+00   4.65e+00    1.46  0.14350
## country.codeSLV -1.32e+01   3.89e+00   -3.40  0.00070 ***
## country.codeSOM  4.13e+01   6.90e+00    5.98  2.9e-09 ***
## country.codeSRB -8.19e+00   4.39e+00   -1.87  0.06201 .
## country.codeSSD  2.95e+01   5.44e+00    5.42  7.2e-08 ***
## country.codeSTP -1.35e+00   4.88e+00   -0.28  0.78170
## country.codeSUR  4.59e+00   6.85e+00    0.67  0.50235
## country.codeSVK -1.20e+01   4.27e+00   -2.81  0.00506 **
## country.codeSVN -1.46e+01   4.35e+00   -3.36  0.00079 ***
## country.codeSWE -1.87e+01   4.37e+00   -4.29  1.9e-05 ***
## country.codeSWZ  4.12e+01   4.90e+00    8.42  < 2e-16 ***
## country.codeSYC -9.71e+00   5.55e+00   -1.75  0.08054 .
## country.codeSYR -1.14e+01   5.45e+00   -2.09  0.03702 *
## country.codeTCD  1.27e+01   4.87e+00    2.62  0.00898 **
## country.codeTGO  1.22e+01   4.90e+00    2.50  0.01270 *
## country.codeTHA -1.89e+01   3.85e+00   -4.92  9.9e-07 ***
## country.codeTJK -9.27e+00   4.43e+00   -2.09  0.03649 *
## country.codeTKM  1.11e+01   6.82e+00    1.62  0.10528
## country.codeTLS -1.23e+01   4.94e+00   -2.49  0.01286 *
## country.codeTON -1.48e+01   4.95e+00   -2.98  0.00289 **
## country.codeTTO -1.71e+01   5.62e+00   -3.05  0.00237 **
## country.codeTUN -1.22e+01   4.19e+00   -2.91  0.00372 **
## country.codeTUR -1.17e+01   3.81e+00   -3.08  0.00211 **
## country.codeTUV -9.63e+00   7.06e+00   -1.36  0.17268
## country.codeTZA  2.27e+01   4.61e+00    4.92  9.6e-07 ***
## country.codeUGA  7.75e+00   4.10e+00    1.89  0.05895 .
## country.codeUKR -1.90e+01   4.01e+00   -4.73  2.5e-06 ***
```

```
## country.codeURY -1.48e+01   4.14e+00   -3.57  0.00037 ***
## country.codeUSA -1.87e+01   4.46e+00   -4.19  3.0e-05 ***
## country.codeUZB  3.84e+01   4.64e+00    8.26  3.4e-16 ***
## country.codeVEN -1.05e+01   3.98e+00   -2.64  0.00841 **
## country.codeVNM -2.01e+01   4.02e+00   -5.01  6.1e-07 ***
## country.codeVUT -3.46e+00   5.37e+00   -0.64  0.51924
## country.codeWSM -1.26e+01   4.93e+00   -2.56  0.01050 *
## country.codeXKX -8.48e+00   4.20e+00   -2.02  0.04380 *
## country.codeYEM -1.51e+01   4.84e+00   -3.13  0.00181 **
## country.codeZAF  8.47e+00   4.97e+00    1.70  0.08877 .
## country.codeZMB  1.91e+01   4.07e+00    4.69  3.1e-06 ***
## country.codeZWE  1.30e+00   4.92e+00    0.26  0.79161
## year            -2.35e-01   3.58e-02   -6.57  7.4e-11 ***
## incomeL          3.81e-01   1.66e+00    0.23  0.81848
## incomeLM        -5.25e+00   1.22e+00   -4.32  1.7e-05 ***
## incomeUM        -2.97e+00   9.38e-01   -3.16  0.00159 **
## edu.total       -4.20e-01   1.48e-01   -2.83  0.00473 **
## hlth             2.48e-01   1.11e-01    2.22  0.02643 *
## mil             -4.60e-01   2.11e-01   -2.18  0.02950 *
## fdi             -8.27e-12   4.50e-12   -1.84  0.06627 .
## lbr.part         1.69e-01   2.88e-02    5.88  5.1e-09 ***
## unemp           -9.86e-02   4.65e-02   -2.12  0.03422 *
## pop.gwth.total  -3.12e+00   7.57e-01   -4.13  3.9e-05 ***
## pop.gwth.rural   8.28e-01   3.17e-01    2.61  0.00910 **
## pop.gwth.urban   2.37e+00   4.02e-01    5.89  4.8e-09 ***
## gdp.dflt        -1.56e-04   1.04e-03   -0.15  0.88032
## gcf             -2.37e-01   3.15e-02   -7.50  1.2e-13 ***
## trade           -1.57e-02   9.03e-03   -1.74  0.08294 .
## gdp.pc           2.85e-04   2.76e-05   10.35  < 2e-16 ***
## lfdi            -1.20e-04   5.61e-02    0.00  0.99829
## lgdp.dflt        1.91e-01   1.39e-01    1.37  0.17078
## lgdp.pc         -6.19e+00   5.99e-01  -10.32  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.78 on 1320 degrees of freedom
## Multiple R-squared:  0.916,  Adjusted R-squared:  0.904
## F-statistic: 76.9 on 187 and 1320 DF,  p-value: <2e-16
```

There are also some weak variable, we should perform some variable selection.

B. VIF Our data set contains a lots of variables. We can perform some variable selection to reduce over-fitting.

```
gvif <- lapply(fit1, vif)
gvif
```

```
## [[1]]
##                    GVIF  Df GVIF^(1/(2*Df))
## country.code   2.63e+08 167            1.06
## year           5.47e+00   1            2.34
## income         1.16e+02   3            2.21
## edu.total      2.72e+00   1            1.65
```

```
## hlth           4.03e+00   1            2.01
## mil            3.95e+00   1            1.99
## fdi            2.34e+00   1            1.53
## lbr.part       3.29e+00   1            1.81
## unemp          3.50e+00   1            1.87
## pop.gwth.total 3.46e+01   1            5.88
## pop.gwth.rural 1.01e+01   1            3.18
## pop.gwth.urban 2.09e+01   1            4.57
## gdp.dflt       1.31e+00   1            1.15
## gcf            2.35e+00   1            1.53
## trade          9.59e+00   1            3.10
## gdp.pc         1.39e+01   1            3.73
## lfdi           1.96e+00   1            1.40
## lgdp.dflt      2.39e+00   1            1.55
## lgdp.pc        3.99e+01   1            6.32
##
## [[2]]
##                    GVIF  Df GVIF^(1/(2*Df))
## country.code   4.42e+08 167            1.06
## year           6.02e+00   1            2.45
## income         1.13e+02   3            2.20
## edu.total      2.57e+00   1            1.60
## hlth           4.38e+00   1            2.09
## mil            3.81e+00   1            1.95
## fdi            2.34e+00   1            1.53
## lbr.part       3.53e+00   1            1.88
## unemp          3.63e+00   1            1.91
## pop.gwth.total 3.38e+01   1            5.82
## pop.gwth.rural 9.90e+00   1            3.15
## pop.gwth.urban 2.09e+01   1            4.57
## gdp.dflt       1.34e+00   1            1.16
## gcf            2.50e+00   1            1.58
## trade          1.09e+01   1            3.31
## gdp.pc         1.43e+01   1            3.78
## lfdi           2.01e+00   1            1.42
## lgdp.dflt      2.36e+00   1            1.54
## lgdp.pc        4.42e+01   1            6.65
##
## [[3]]
##                    GVIF  Df GVIF^(1/(2*Df))
## country.code   5.47e+08 167            1.06
## year           5.88e+00   1            2.42
## income         1.11e+02   3            2.19
## edu.total      2.64e+00   1            1.63
## hlth           5.46e+00   1            2.34
## mil            3.85e+00   1            1.96
## fdi            2.33e+00   1            1.53
## lbr.part       3.54e+00   1            1.88
## unemp          3.71e+00   1            1.93
## pop.gwth.total 3.45e+01   1            5.87
## pop.gwth.rural 9.96e+00   1            3.16
## pop.gwth.urban 2.06e+01   1            4.54
## gdp.dflt       1.34e+00   1            1.16
## gcf            2.56e+00   1            1.60
```

```
## trade          1.16e+01   1            3.41
## gdp.pc         1.41e+01   1            3.76
## lfdi           2.07e+00   1            1.44
## lgdp.dflt      2.29e+00   1            1.51
## lgdp.pc        4.24e+01   1            6.51
```

Some forum suggested that we should use the standard $GVIF^{1/(2 \cdot df)} < 2$ as equivalent to $GVIF < 4$ to account for high degree of freedom of some variables.

```
# adjusted gvif higher than 3
lapply(gvif, function(g) {
    g[g[, 3] > 3, 3]
})
```

```
## [[1]]
## pop.gwth.total pop.gwth.rural pop.gwth.urban          trade         gdp.pc
##           5.88           3.18           4.57           3.10           3.73
##        lgdp.pc
##           6.32
##
## [[2]]
## pop.gwth.total pop.gwth.rural pop.gwth.urban          trade         gdp.pc
##           5.82           3.15           4.57           3.31           3.78
##        lgdp.pc
##           6.65
##
## [[3]]
## pop.gwth.total pop.gwth.rural pop.gwth.urban          trade         gdp.pc
##           5.87           3.16           4.54           3.41           3.76
##        lgdp.pc
##           6.51
```

Remove the highest-ranked variables (`pop.gwth.total`, `pop.gwth.urban`, `pop.gwth.rural`, `gdp.pc`, `lgdp.pc`) and rebuild the models.

```
formula.str <- "pov ~ country.code+year+income+edu.total+hlth+mil+fdi+lbr.part+unemp+gdp.dflt+gcf+trade+

# Using single imputed data set
fit1 <- lapply(1:countries1.imputed$m, function(i) lm(as.formula(formula.str),
    data = complete(countries1.imputed, i)))
```

Re-run GVIF analysis

```
gvif <- lapply(fit1, vif)

lapply(gvif, function(g) {
    g[g[, 3] > 2, 3]
})
```

```
## [[1]]
## income  trade
##   2.05   2.98
```

```
##
## [[2]]
## income   hlth  trade
##   2.05   2.07   3.18
##
## [[3]]
## income   hlth  trade
##   2.06   2.33   3.28
```

There're still some variable with adjusted GVIF > 2. We are interested in the effects of `hlth` (expenditure in Healthcare), so we won't remove those variables. We can remove `trade`.

```
# remove trade
formula.str <- "pov ~ country.code+year+income+edu.total+hlth+mil+fdi+lbr.part+unemp+gdp.dflt+gcf+lfdi+l

# Using single imputed data set
fit1 <- lapply(1:countries1.imputed$m, function(i) lm(as.formula(formula.str),
    data = complete(countries1.imputed, i)))

# gvif
gvif <- lapply(fit1, vif)

lapply(gvif, function(g) {
    g[g[, 3] > 2, 3]
})
```

```
## [[1]]
## [1] 2.05
##
## [[2]]
## income   hlth
##   2.04   2.07
##
## [[3]]
## income   hlth
##   2.05   2.32
```

The adjusted GVIF are acceptably low.

Coefficients overview.

```
# find coef of variable other than country.code
var <- c("year", "incomeL", "incomeLM", "incomeUM",
    "edu.total", "hlth", "mil", "fdi", "lbr.part",
    "unemp", "gdp.dflt", "gcf", "lfdi", "lgdp.dflt")

res <- do.call(rbind, lapply(fit1, function(f) f$coefficients[var]))
kable(res)
```

year	incomeL	incomeLM	incomeUM	edu.total	hlth	mil	fdi	lbr.part	unemp	gdp.dflt	gcf	lfdi	lgdp.dflt
-0.350	8.88	-1.298	-3.37	-0.624	0.337	-0.526	0	0.212	-0.121	0.001	-0.287	-0.110	0.188

year	income	LincomeLM	McomeUpM	edu.total	hlth	mil	fdi	lbr.part	unemp	gdp.dflt	gcf	lfdi	lgdp.dflt
-0.355	10.61	-0.365	-2.51	-0.443	0.481	-0.251	0	0.335	-0.018	0.003	-0.287	-0.115	0.149
-0.346	8.75	-1.325	-3.33	-0.864	0.389	-0.228	0	0.215	-0.038	0.002	-0.285	-0.118	0.074

There are several sign-switching in `income` coefficients, indicating the effect of removing some multicollinearity.

```
fit1.summ <- lapply(fit1, function(f) {
    k <- length(f$coefficients) - 1
    pred <- predict(f, countries.test.4)
    data.frame(`Train Adj.R2` = summary(f)$r.squared,
        `Test Adj.R2` = r2(pred, countries.test.4$pov))
})

res <- do.call(rbind.data.frame, fit1.summ)
res <- cbind(data.frame(Set = 1:3), res)
kable(res)
```

Set	Train.Adj.R2	Test.Adj.R2
1	0.902	0.717
2	0.907	0.712
3	0.902	0.719

Directly removing variables seems to be penalising our test results. As our interest is to investigate the effect of certain variables on poverty, the yielded R-Squared is within acceptable range. We can continue variable selection on the new models.

```
formula.str <- "pov ~ country.code+year+income+edu.total+hlth+mil+fdi+lbr.part+unemp+gdp.dflt+gcf+lfdi+l
```

B. Step-wise AIC We can conduct a step-wise AIC variable selection. It is similar to the procedure we use in class, but based on a metric call AIC (Akaike Information Criterion), which is an estimator of prediction error and relative quality of statistical models. The lower AIC is, the better the model fits. \

```
# helper
fitAIC <- function(i) {
    set.seed(1984)
    fit <- lm(as.formula(formula.str), complete(countries1.imputed,
        i))
    aic.fit <- stepAIC(fit, trace = F, direction = "backward")
    return(aic.fit)
}

# build models
aic.fits <- lapply(1:countries1.imputed$m, fitAIC)

# check final terms of each model
lapply(aic.fits, function(mod) formula(mod$terms))
```

```
## [[1]]
## pov ~ country.code + year + income + edu.total + hlth + mil +
##     lbr.part + unemp + gcf + lfdi + lgdp.dflt
## <environment: 0x556de946ca18>
##
## [[2]]
## pov ~ country.code + year + income + edu.total + hlth + lbr.part +
##     gdp.dflt + gcf + lfdi
## <environment: 0x556de104e658>
##
## [[3]]
## pov ~ country.code + year + income + edu.total + hlth + lbr.part +
##     gdp.dflt + gcf + lfdi
## <environment: 0x556de3cb5898>
```

The most commonly removed variables are:

- gdp.dflt
- ldgp.dflt
- unemp
- fdi

```
# predict with each model
aic.pred <- lapply(aic.fits, predict, newdata = countries.test.4)
# calculate adj r2
aic.train.r2 <- unlist(lapply(aic.fits, function(model) summary(model)$adj.r.squared))

k <- lapply(aic.fits, function(m) length(m$coefficients))
aic.test.r2 <- unlist(mapply(r2, aic.pred, MoreArgs = list(orig = countries.test.4$pov)))

res <- data.frame(`set no.` = 1:3, train = aic.train.r2,
    test = aic.test.r2)
```

We can compare R-Squared to estimate relative over-fitting in the new models.

```
kable(res)
```

set.no.	train	test
1	0.889	0.719
2	0.895	0.709
3	0.888	0.717

Performance on train set and test set are slightly improved. However, we have simplified the model by removing some unnecessary terms. We can update our models.

```
formula.str <- "pov ~ country.code+year+income+edu.total+hlth+mil+lbr.part+gcf+lfdi"

fit3 <- lapply(1:countries1.imputed$m, function(i) lm(as.formula(formula.str),
    data = complete(countries1.imputed, i)))

# Build models with all generated data set and
```

```
# pool the estimates
fit4 <- with(data = countries1.imputed, exp = lm(as.formula(formula.str)))
fit4.combined <- pool(fit4)
# dummy lm model
fit4.dummy <- lm(as.formula(formula.str), data = complete(countries1.imputed,
    1))
# replace coefficients of dummy model & predict
fit4.dummy$coefficients <- fit4.combined$pooled$estimate

fit3.summ <- lapply(fit3, function(f) {
    k <- length(f$coefficients) - 1
    pred <- predict(f, countries.test.4)
    test.resid <- pred - countries.test.4$pov
    data.frame(`Train MSE` = mean(summary(f)$residuals^2),
        `Test MSE` = mean(test.resid^2), `Train MAE` = mean(abs(summary(f)$residuals)),
        `Test MAE` = mean(abs(test.resid)), `Train R2` = summary(f)$r.squared,
        `Test R2` = r2(pred, countries.test.4$pov))
})

k <- length(fit4.dummy$coefficients) - 1
fit4.pred <- predict(fit4.dummy, countries.test.4)
test.resid <- fit4.pred - countries.test.4$pov
fit4.summ <- data.frame(`Train MSE` = mean(summary(fit4.dummy)$residuals^2),
    `Test MSE` = mean(test.resid^2), `Train MAE` = mean(abs(summary(fit4.dummy)$residuals)),
    `Test MAE` = mean(abs(test.resid)), `Train R2` = pool.r.squared(fit4)[,
        "est"], `Test R2` = r2(fit4.pred, countries.test.4$pov))

res <- do.call(rbind.data.frame, fit3.summ)
res <- rbind(res, fit4.summ)
res <- cbind(data.frame(Set = c(1:3, "Pooled")), res)

kable(res)
```

Set	Train.MSE	Test.MSE	Train.MAE	Test.MAE	Train.R2	Test.R2
1	34.2	23.9	3.73	2.80	0.902	0.719
2	32.5	24.5	3.69	2.85	0.907	0.711
3	34.4	23.9	3.76	2.79	0.901	0.719
Pooled	34.2	24.0	3.73	2.80	0.903	0.718

Coefficients overview.

```
var <- c("year", "incomeL", "incomeLM", "incomeUM",
    "edu.total", "hlth", "mil", "lbr.part", "gcf",
    "lfdi")

res <- do.call(rbind, lapply(fit3, function(f) f$coefficients[var]))
kable(res)
```

year	incomeL	incomeLM	incomeUM	edu.total	hlth	mil	lbr.part	gcf	lfdi
-0.359	8.70	-1.466	-3.47	-0.651	0.252	-0.541	0.223	-0.268	-0.131

year	incomeL	incomeLM	incomeUM	edu.total	hlth	mil	lbr.part	gcf	lfdi
-0.368	10.41	-0.534	-2.59	-0.450	0.452	-0.281	0.333	-0.274	-0.132
-0.353	8.56	-1.508	-3.45	-0.887	0.352	-0.242	0.217	-0.272	-0.130

We can further treat over-fitting with regularization.

C. Regularization Helper functions to standardise data.

```
sd0 <- function(vct) {
    if (!is.numeric(vct)) {
        return(NA)
    }

    return(sd(vct, na.rm = T))
}


std0 <- function(vct, scl) {
    if (!is.numeric(vct)) {
        return(vct)
    }

    return(vct/scl)
}

standardise <- function(train, test) {
    # save number of train and combine both data
    # set
    trainCases <- 1:nrow(train)
    data <- rbind(train, test)
    # calc scaler
    scaler <- unlist(lapply(data, sd0))
    # get the numeric columns
    numCols <- which(unlist(lapply(data, is.numeric)))
    # divide numeric columns by scalers, and
    # combine with factor columns
    num <- as.data.frame(mapply(std0, vct = data[,
        numCols], scl = scaler[numCols], SIMPLIFY = T))
    fct <- data[, -numCols]
    final <- cbind(fct, num)
    # split to train and test
    trainSet <- final[trainCases, ]
    testSet <- final[-trainCases, ]

    res <- list(final = final, train = trainSet, test = testSet,
        scaler = scaler)
    return(res)
}
```

Helper function to build the optimal model.

```
optimalModel <- function(formula, train, test) {
    set.seed(1984)
    # standise and save number of train cases (to
    # split later)
    std.data <- standardise(train, test)
    trainCases <- 1:nrow(train)
    # matrix-ise
    xs <- model.matrix(formula, std.data$final)[, -1]
    ys <- std.data$final$pov
    # split data
    train.x <- xs[trainCases, ]
    train.y <- ys[trainCases]
    test.x <- xs[-trainCases, ]
    test.y <- ys[-trainCases]
    # list of tested alpha, resolution = 0.1
    alphas <- seq(0, 1, 0.1)
    # build a bunch of models
    models <- lapply(alphas, function(a) cv.glmnet(train.x,
        train.y, type.measure = "mse", alpha = a))
    cv.error <- unlist(lapply(models, function(model) model$cvm[model$lambda ==
        model$lambda.min]))
    # best model
    best.model.idx <- which.min(cv.error)
    # optimal alpha and lambda
    alpha.opt <- alphas[best.model.idx]
    lambda.opt <- models[[best.model.idx]]$lambda.min
    best.model <- glmnet(train.x, train.y, alpha = alpha.opt,
        lambda = lambda.opt)
    # predict for test data
    train.fit <- predict(best.model, train.x)
    test.fit <- predict(best.model, test.x)
    # evaluation - train
    k <- best.model$df
    train.r2 <- best.model$dev.ratio
    train.adjR2 <- adjR2(train.fit, train.y, k)
    train.resid <- train.fit - train.y
    train.mse <- mean(train.resid^2)
    train.rmse <- sqrt(train.mse)
    train.mae <- mean(abs(train.resid))
    train.mape <- mean(abs(train.resid/train.y))
    # evaluation - test
    test.r2 <- r2(test.fit, test.y)
    test.adjR2 <- adjR2(test.fit, test.y, k)
    test.resid <- test.fit - test.y
    test.mse <- mean(test.resid^2)
    test.rmse <- sqrt(test.mse)
    test.mae <- mean(abs(test.resid))
    test.mape <- mean(abs(test.resid/test.y))
    # final results
    results <- list(train = list(data = train, r2 = train.r2,
        adj.r2 = train.adjR2, fitted = train.fit, residuals = train.resid,
        mse = train.mse, rmse = train.rmse, mae = train.mae,
        mape = train.mape), test = list(data = test,
```

```
        r2 = test.r2, adj.r2 = test.adjR2, fitted = test.fit,
        residuals = test.resid, mse = test.mse, rmse = test.rmse,
        mae = test.mae, mape = test.mape), scaler = std.data$scaler,
        alpha = alpha.opt, lambda = lambda.opt, model = best.model)

    return(results)
}
```

Build the optimal models using each imputed data set.

```
# helper function to build from a specific
# imputed data
buildWith <- function(set, imputed, test, formula,
    fun) {
    train <- complete(imputed, set)
    model <- fun(formula, train, test)
    return(model)
}

set.seed(1984)

models <- lapply(1:countries1.imputed$m, buildWith,
    imputed = countries1.imputed, test = countries.test.4,
    formula = as.formula(formula.str), fun = optimalModel)
```

Display parameters and evaluation.

```
result.list <- lapply(models, function(mod) {
    data.frame(alpha = mod$alpha, lambda = mod$lambda,
        `Train MSE` = mod$train$mse, `Test MSE` = mod$test$mse,
        `Train MAE` = mod$train$mae, `Test MAE` = mod$test$mae,
        `Train R2` = mod$train$r2, `Test R2` = mod$test$r2)
})

result.df <- do.call(rbind.data.frame, result.list)
result.df <- cbind(data.frame(Set = 1:countries1.imputed$m),
    result.df)
kable(result.df)
```

Set	alpha	lambda	Train.MSE	Test.MSE	Train.MAE	Test.MAE	Train.R2	Test.R2
1	0.0	0.079	0.116	0.073	0.211	0.147	0.897	0.735
2	0.0	0.080	0.111	0.075	0.209	0.149	0.902	0.729
3	0.7	0.000	0.112	0.078	0.214	0.159	0.901	0.718

There are some improvement on the test data performance based on adjusted R-Squared. MSE and MAE remain low on both data sets.

Coefficients overview.

```r
var <- c("year", "incomeL", "incomeLM", "incomeUM",
    "edu.total", "hlth", "mil", "lbr.part", "gcf",
    "lfdi")

res <- do.call(rbind, lapply(models, function(f) coefficients(f$model)[var,
    ]))
res <- cbind(data.frame(Set = 1:3), res)
kable(res)
```

Set	year	incomeL	incomeLM	incomeUM	edu.total	hlth	mil	lbr.part	gcf	lfdi
1	-0.142	0.763	0.076	-0.118	-0.071	-0.013	-0.030	0.123	-0.091	-0.040
2	-0.146	0.812	0.080	-0.091	-0.058	0.001	-0.017	0.155	-0.092	-0.043
3	-0.179	0.524	-0.061	-0.186	-0.083	0.041	-0.012	0.110	-0.109	-0.028

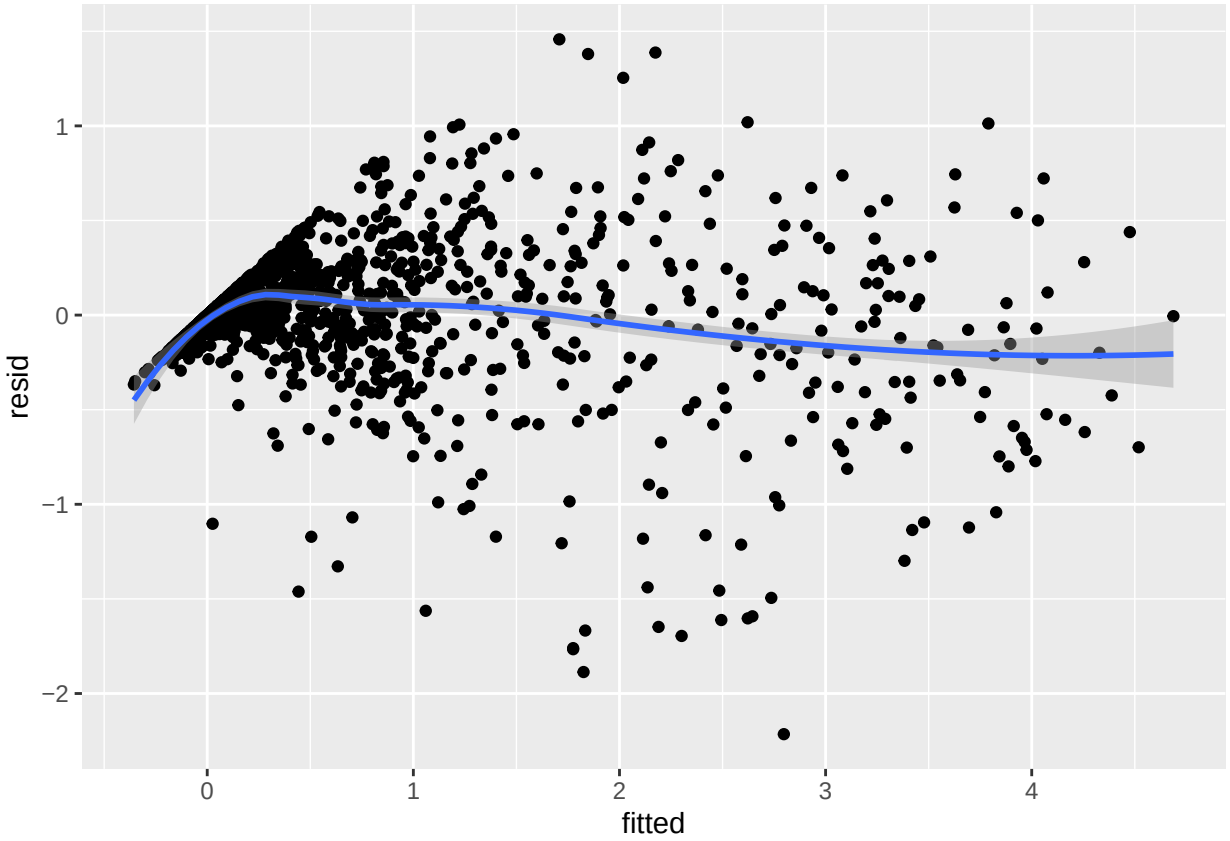
All non-country.code variables are non-zero. We can observe that the regularisation models just remove the effect of `country.code`, for countries that have similar "baseline".

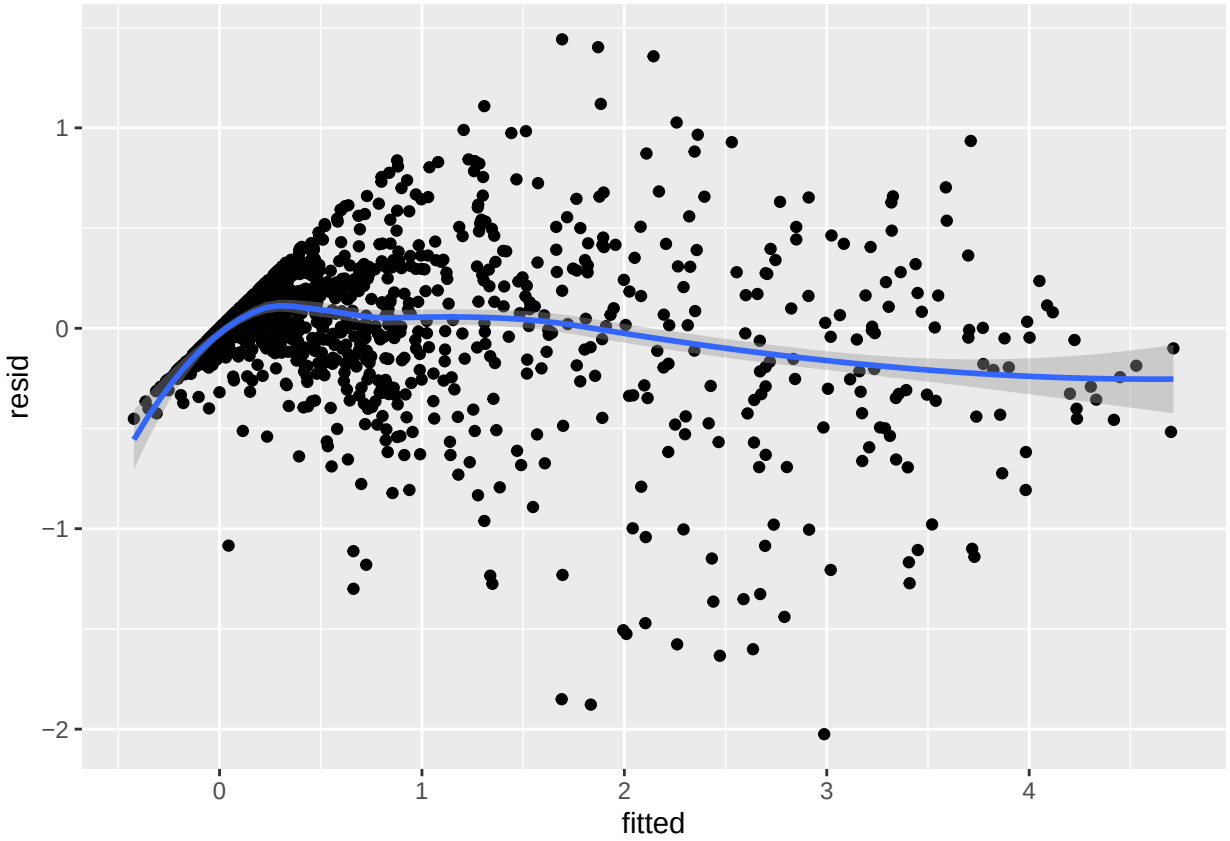
3.4.2. Assessment

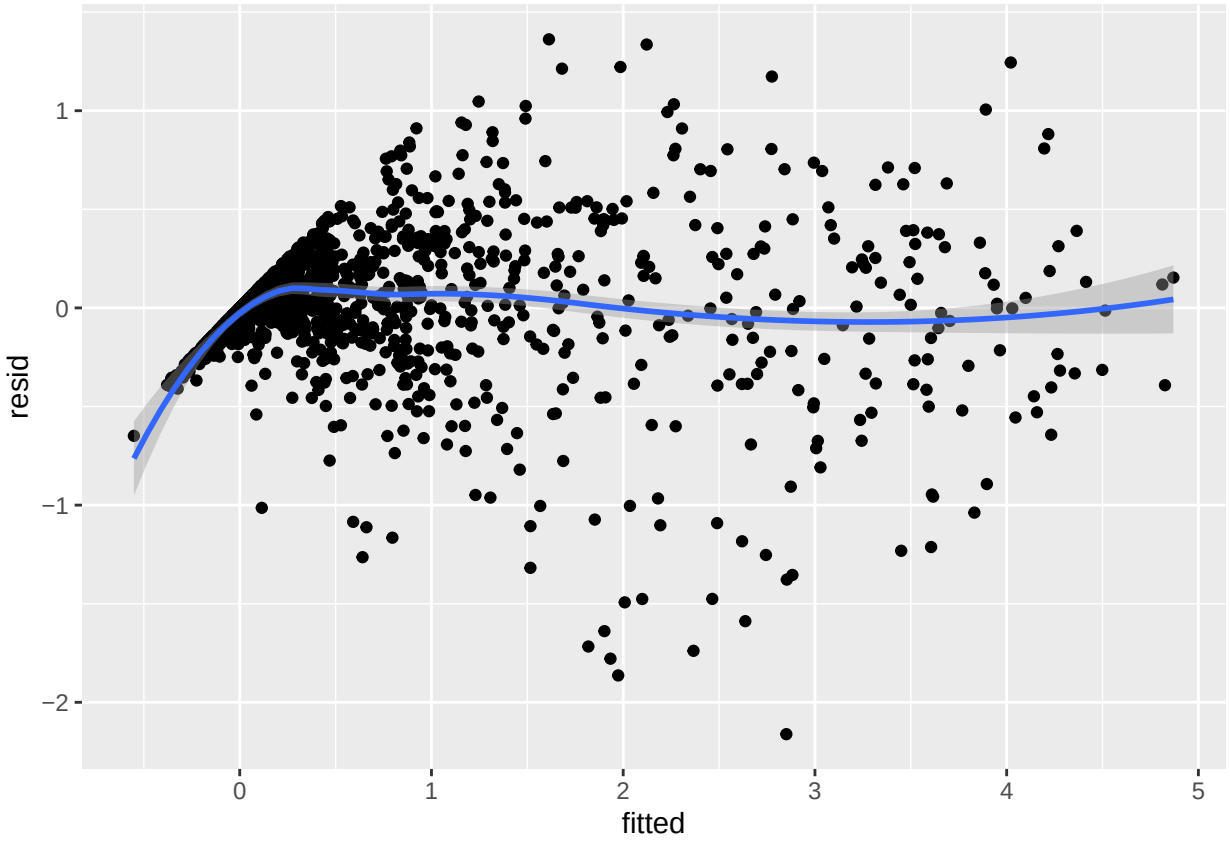
Now we are evaluating some performance metrics, checking assumptions, and remedy some potential problems.

A. Homoscedasticity Plot the residuals ~ pov

```r
for (mod in models) {
    resid <- mod$train$residuals
    fitted <- mod$train$fitted
    # plot(resid ~ pov, main = paste('Model', i))
    print(ggplot(mapping = aes(y = resid, x = fitted)) +
        geom_point() + geom_smooth(formula = y ~ x,
        method = loess))
}
```


resid
fitted
1
0
−1
−2
0
1
2
3
4


resid
fitted
1
0
−1
−2
0
1
2
3
4



There seems to be some larger variance with higher values of `pov`. We can conduct a statistical test to confirm the present of heteroscedasticity.

The Goldfeld-Quandt test is performed by eliminating a certain number of observations from the dataset's center, then comparing the spread of residuals between the two datasets on either side of the central observations.

The Goldfeld-Quandt test examines two submodels' variances divided by a defined breakpoint and rejects if the variances disagree.

Under H0, the Goldfeld-Quandt test's test statistic follows an F distribution with degrees of freedom as specified in the parameter.

- **Null (H0):** Heteroscedasticity is not present.
- **Alternative (H1):** Heteroscedasticity is present.

```
lapply(models, function(m) {
    resid <- m$train$residuals
    # apply Goldfeld-Quandt test
    gqtest(formula = resid ~ 1, order.by = m$train$fitted)
})
```

```
## [[1]]
##
##  Goldfeld-Quandt test
##
## data:  resid ~ 1
```

```
## GQ = 17, df1 = 753, df2 = 753, p-value <2e-16
## alternative hypothesis: variance increases from segment 1 to 2
##
##
## [[2]]
##
##  Goldfeld-Quandt test
##
## data:  resid ~ 1
## GQ = 14, df1 = 753, df2 = 753, p-value <2e-16
## alternative hypothesis: variance increases from segment 1 to 2
##
##
## [[3]]
##
##  Goldfeld-Quandt test
##
## data:  resid ~ 1
## GQ = 13, df1 = 753, df2 = 753, p-value <2e-16
## alternative hypothesis: variance increases from segment 1 to 2
```

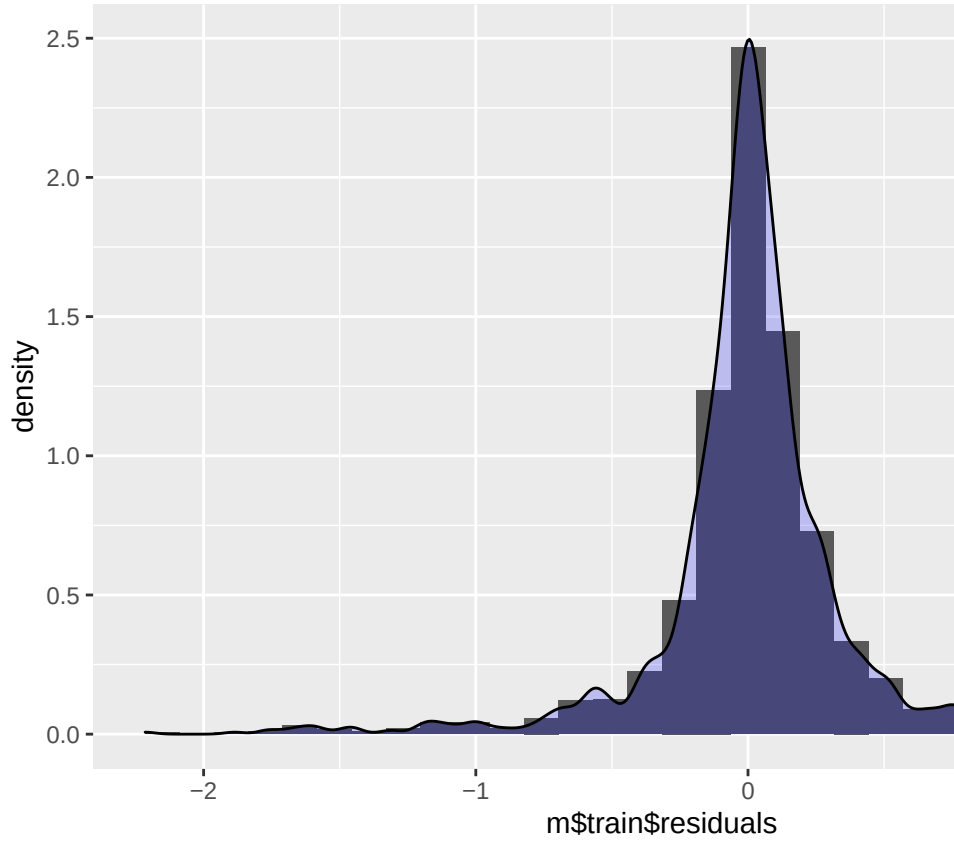
There is heteroscedasticity in our models. Generalized Least Squares With Unknown For of Variance is a possible remedy of this problem, but at the cost of model interpretation. The high variance in higher end of `pov` can be explained by the relative volatile economical and political climate in highly poor countries (one standard deviation from mean), leading to unstable effect of predictors.

B. Independence Some possible source of dependency are poverty measured on the same country, or in the same year. These are accounted in our models by controlling those variables.

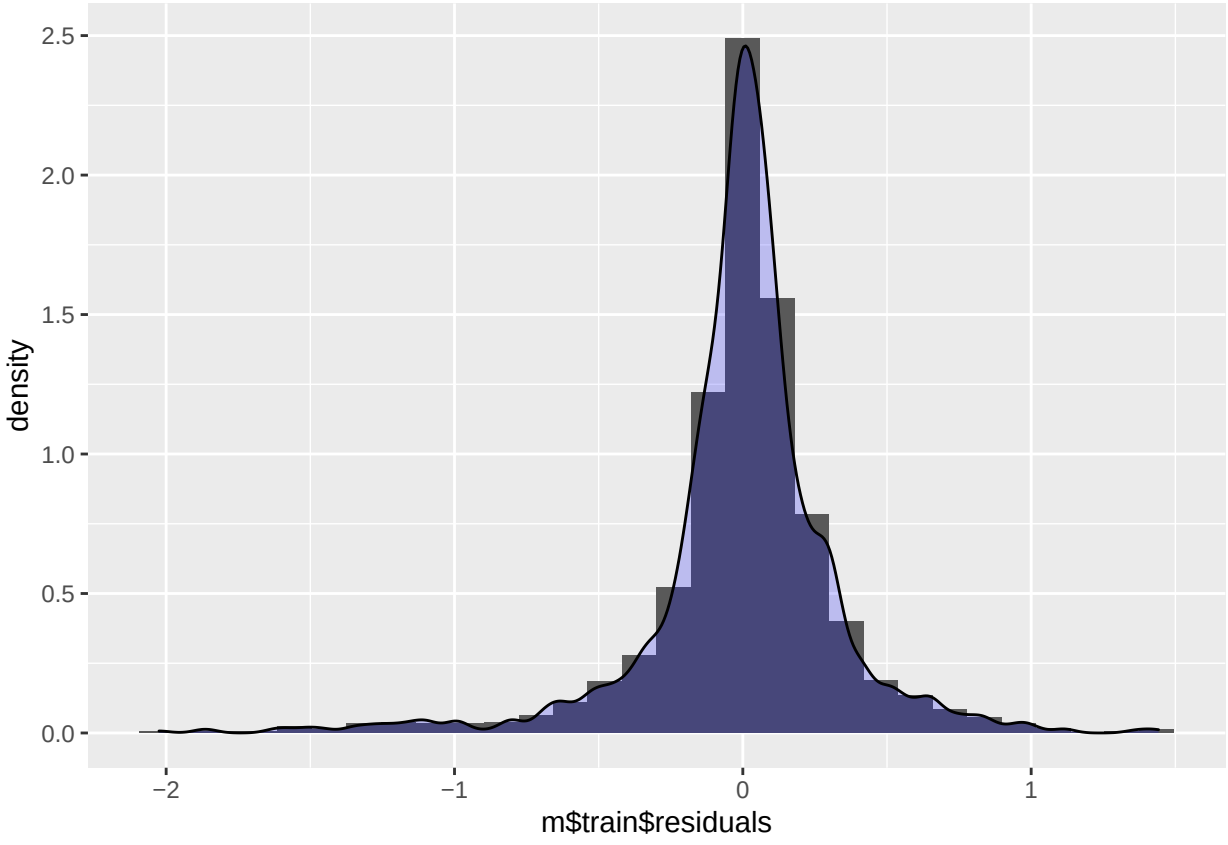
Some countries have been engaging in international wars, being under the influences of foreign forces such as Iraq, and Afghanistan. They possess similar (and different) political climates. Some other groups of economical alliance, or enjoying similar natural resources: OPEC, EU, ASEAN, etc. might have similar characteristics that make them dependent.

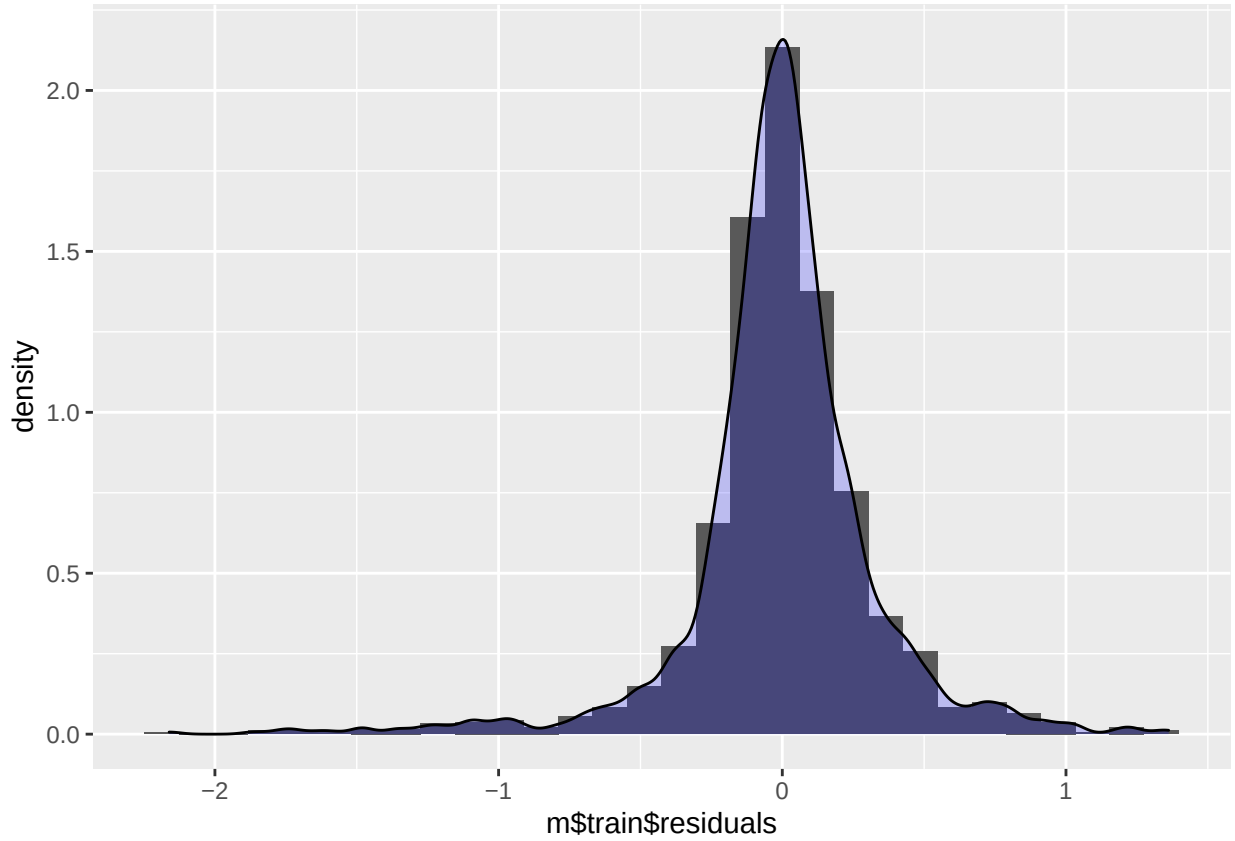
A solution to resolve this is to conduct the study on countries with similar geographical characteristics. In our study, however, we are interested in the general effects of certain variable across the globe. Although there are some dependence, the large number of entities with various features, we hypothesize, will cancel each other out, resulting in a net effect that gives an overview of the influences of variables.

```
for (m in models) {
    p <- ggplot(mapping = aes(x = m$train$residuals)) +
        geom_histogram(aes(y = ..density..), bins = 30) +
        geom_density(alpha = 0.2, fill = "blue")

    print(p)
}
```



C. Normally Distributed Residuals

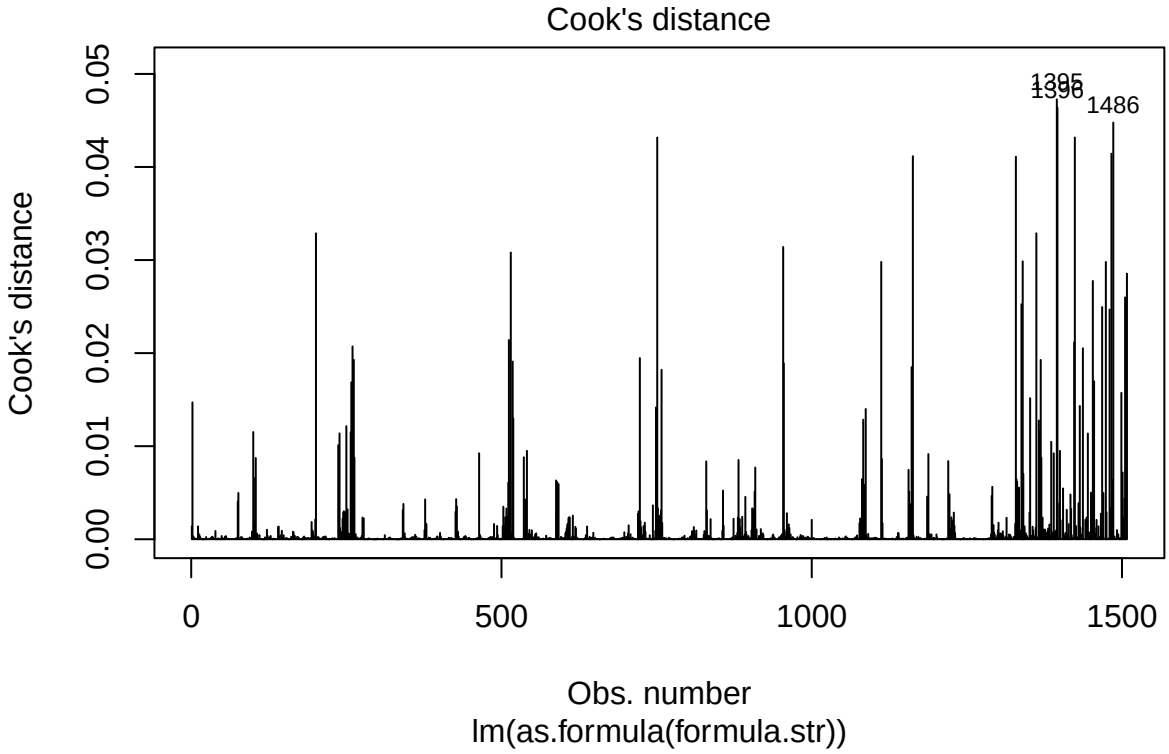


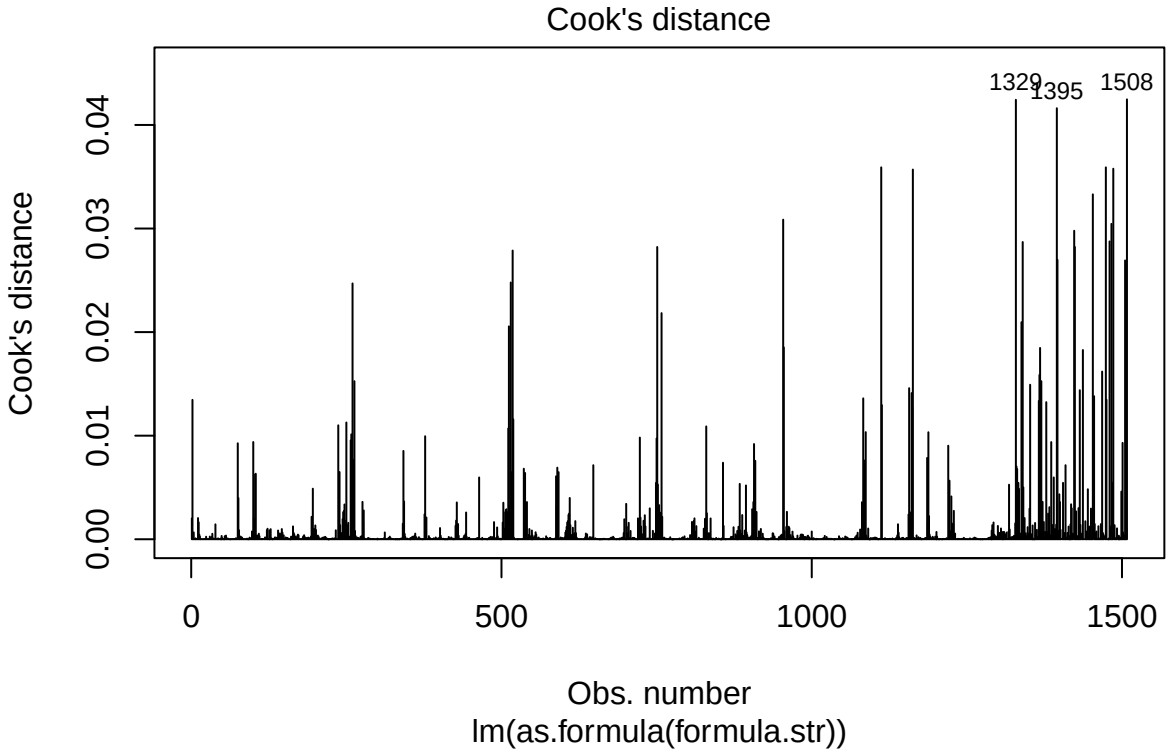


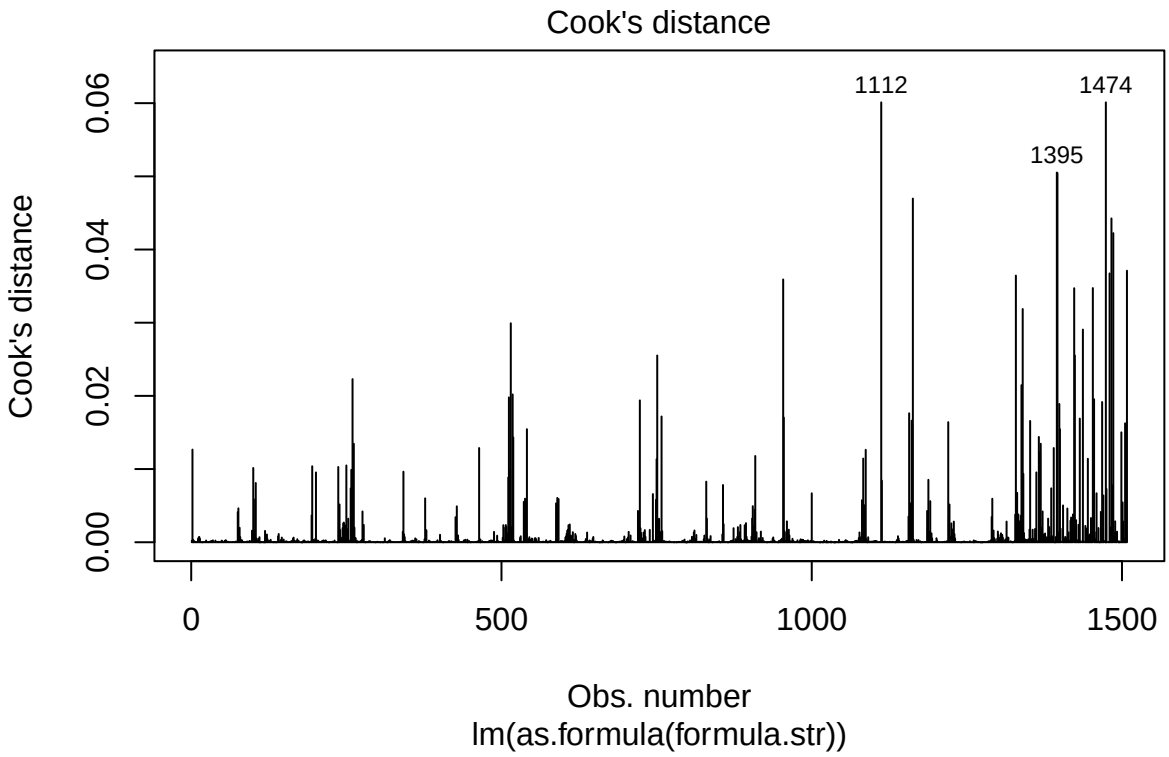
Although quite negatively skewed, the residuals are visibly normal.

D. Outliers As elastic net models produced in the last section are not very fit for detecting outliers using R-support function. As they just removed dummy variables in `country.code`, we can using OLS to detect some potential outliers. They are the same models we produced after the step-AIC selection (`fit3` and `fit4`)

```
for (mod in fit3) {
    plot(mod, which = 4)
}
```


Cook's distance
1395
1396
1486
Cook's distance
0.00
0.01
0.02
0.03
0.04
0.05
0
500
1000
1500
Obs. number
lm(as.formula(formula.str))


Cook's distance
1329
1395
1508
Cook's distance
0.00
0.01
0.02
0.03
0.04
0
500
1000
1500
Obs. number
lm(as.formula(formula.str))



Some obvious outliers are point 1112, 1329, 1395, 1396, 1424, 147, 1486, 1508.

```
out <- c(1112, 1329, 1395, 1396, 1424, 147, 1486, 1508)
var <- c("country.code", "country.name", "year", "pov",
    "income", "edu.total", "hlth", "mil", "lbr.part",
    "gcf", "lfdi")
# See the original data instead of imputed data
countries.train.4[out, var]
```

```
## # A tibble: 8 x 11
## # Groups:   country.code [8]
##   country.~1 count~2  year   pov income edu.t~3  hlth    mil lbr.p~4   gcf  lfdi
##   <fct>      <chr>   <dbl> <dbl> <fct>    <dbl> <dbl>  <dbl>   <dbl> <dbl> <dbl>
## 1 SSD        South ~  2016  67.3 L         1.54 NA     4.60     NA   NA     NA
## 2 YEM        Yemen,~  2014  19.8 LM       NA     4.84  3.97     36.2 NA     NA
## 3 GIN        Guinea   1991  87.9 L         2.02 NA     2.38     NA   18.2   17.5
## 4 GMB        Gambia~  1998  74.1 L        NA    NA     0.385    NA    4.88  17.0
## 5 LCA        St. Lu~  1995  38.1 UM       NA    NA    NA        67.8 NA     17.3
## 6 BOL        Bolivia  2001  19   LM       NA     4.83  2.26     67.8 14.3   20.4
## 7 TJK        Tajiki~  1999  62   L         2.08 NA     1.39     NA   17.3   15.7
## 8 ZWE        Zimbab~  2011  21.6 L        NA     8.08  1.41     82.4 17.4   19.7
## # ... with abbreviated variable names 1: country.code, 2: country.name,
## #   3: edu.total, 4: lbr.part
```

These countries all appear very infrequently in our data set (only twice for South Sudan and St.Lucia). The gap between their appearance are quite large, too. Such reasons could have made the model struggle to

predict their poverty states.

Yemen's data were collected in 1998, 2005, and 2014, a huge gap in time, by when it has doubled its poverty rate.

Guinea is an interesting point. Because the outlier data (87.9% in 1991) is quite different to the next data point (40.8% in 1994), a huge jump in just 3 years. This might be a consequence of the political changes (new constitution, establishment of the Supreme Court, etc.) in this period. They are the changes not captured by the variables.

Gambia has the issue with Guinea, when it saw a huge reduction of poverty of 25% from 1998-2003.

Lucia only has two data points with large gap in time (1995-2016). It can be difficult to know much about someone if you only met them twice, with 21 years in between.

These data seems valid and there is no convincing reasons to remove them from the model (in the case of Guinea and Gambia, maybe there was some false reports because the jump were quite suspicious, but we need more evidence to conclude that they are false data).

3.4.3. Interpretation

Final model coefficients (exclude `country.code`):

```
var <- c("year", "incomeL", "incomeLM", "incomeUM",
    "edu.total", "hlth", "mil", "lbr.part", "gcf",
    "lfdi")

res <- do.call(rbind, lapply(models, function(f) coefficients(f$model)[var,
    ]))
res <- cbind(data.frame(Set = 1:3), res)
kable(res)
```

Set	year	incomeL	incomeLM	incomeUM	edu.total	hlth	mil	lbr.part	gcf	lfdi
1	-0.142	0.763	0.076	-0.118	-0.071	-0.013	-0.030	0.123	-0.091	-0.040
2	-0.146	0.812	0.080	-0.091	-0.058	0.001	-0.017	0.155	-0.092	-0.043
3	-0.179	0.524	-0.061	-0.186	-0.083	0.041	-0.012	0.110	-0.109	-0.028

We should convert them to un-standardised value for easier interepretation.

```
var <- c("year", "incomeL", "incomeLM", "incomeUM",
    "edu.total", "hlth", "mil", "lbr.part", "gcf",
    "lfdi")

res <- do.call(rbind, lapply(models, function(f) {
    # get scaler saved in the model
    scaler <- f$scaler
    # categorical has scaler = 1
    scaler[c("incomeL", "incomeLM", "incomeUM")] <- 1
    # beta = std.beta * sigma_y / sigma_x
    coefficients(f$model)[var, ] * scaler["pov"]/scaler[var]
}))
kable(res)
```

year	incomeL	incomeLM	incomeUM	edu.total	hlth	mil	lbr.part	gcf	lfdi
-0.268	13.38	1.32	-2.06	-0.775	-0.087	-0.394	0.240	-0.223	-0.197
-0.277	14.24	1.41	-1.60	-0.632	0.007	-0.217	0.306	-0.230	-0.214
-0.339	9.19	-1.07	-3.27	-0.919	0.289	-0.160	0.215	-0.269	-0.139

4. Conclusion

Approach 1, 2 and 4 seem to have the best performace.

First Approach:

```
knitr::kable(summary_3models[1:3, -5], digits = 3)
```

	MSE	MAE	RMSE	Rsquare
Baseline MLR_Train	3.336	1.137	1.826	0.965
Ridge Model_Train	0.221	0.356	0.470	0.862
LASSO Model_Train	0.216	0.352	0.465	0.865

```
knitr::kable(summary_3models[4:6, -5], digits = 3)
```

	MSE	MAE	RMSE	Rsquare
Baseline MLR_Test	3.598	1.293	1.897	0.881
Ridge Model_Test	0.171	0.318	0.414	0.863
LASSO Model_Test	0.159	0.311	0.398	0.873

Second Approach:

```
knitr::kable(summary_lasso, digits = 3)
```

	MSE	MAE	RMSE	MAPE	R2
LASSO_train	4.94	1.48	2.22	Inf	0.948
LASSO_test	4.67	1.50	2.16	Inf	0.846

```
knitr::kable(summary_Ridge, digits = 3)
```

	MSE	MAE	RMSE	MAPE	R2
Ridge_train	5.37	1.47	2.32	Inf	0.943
Ridge_test	4.42	1.46	2.10	Inf	0.854

Fourth Approach:

```
kable(result.df[c("alpha", "lambda", "Train.R2", "Test.R2")])
```

alpha	lambda	Train.R2	Test.R2
0.0	0.079	0.897	0.735
0.0	0.080	0.902	0.729
0.7	0.000	0.901	0.718

We can deduce our final interpretation from these models.

```
var <- c("incomeL", "incomeLM", "incomeUM", "edu.total",
    "hlth", "mil")
result1 <- coefficients(model3)[var]
# result1[names(scaler2)] <-
# result1[names(scaler2)] /
# scaler2[names(scaler2)] result1 <- result1 *
# scaler2['pov'] result1 <- result1[var]

result2 <- coefficients(glm_Ridge)[, "s0"]
result2[names(scaler2)] <- result2[names(scaler2)]/scaler2[names(scaler2)]
result2 <- result2 * scaler2["pov"]
result2 <- result2[var]

result4 <- res[2, var]

# combine
finalResults <- data.frame(Approach = c(1, 2, 4))
finalResults <- finalResults %>%
    cbind(rbind(result1, result2, result4))

kable(finalResults)
```

	Approach	incomeL	incomeLM	incomeUM	edu.total	hlth	mil
result1	1	0.075	-1.60	-1.45	-0.372	-0.248	0.520
result2	2	65.178	6.84	-8.22	-2.980	-0.591	-0.828
result4	4	14.241	1.41	-1.60	-0.632	0.007	-0.217

Some important interpretation:

- Low income countries generally have higher poverty index than High income countries.
- As suggested by 1st approach, Lower-Middle income countries have higher poverty index than High income countries. While 2nd and 4th approach suggest otherwise.
- Upper-Middle income countries in general have lower poverty index than High income countries.

As suggested by these models. Upper-Middle income countries have lowest poverty rate. This can be attributed to that these countries tend to have more even wealth distribution than high income countries, and have enough money to alleviate poor people.

- Spending on education is negatively correlated with poverty. On average, 1% (of GDP) more spending in education results in 3 point decrease in poverty index. This suggests that public spending on education can reduce poverty.

- Spending on Healthcare is negatively correlated with poverty. On average, 1% (of GDP) more spending in education results in 0.5 point decrease in poverty index.
- Spending on military has ambiguous effect on poverty. Different model results in different interpretation. This can be accounted to the difference in the data used in each approach. Approach 1 only use complete cases, which caused the model to predict positive relationship. However, as more data is utilised in approach 2&4 use more data, the effect tend to be neutral/negative.

5. References

Akbar, Muhammad, Mukaram Khan, Haidar Farooqe, and Kaleemullah. 2019. "Public Spending, Education and Poverty: A Cross Country Analysis" 4 (April): 12–20.

Alruhaymi, Abdullah Z, and Charles J Kim. 2021. "Why Can Multiple Imputations and How (MICE) Algorithm Work?" *Open J. Stat.* 11 (05): 759–77.

Arel-Bundock, Vincent, and Krzysztof J. Pelc. 2018. "When Can Multiple Imputation Improve Regression Estimates?" *Political Analysis* 26 (2): 240–45. https://doi.org/10.1017/pan.2017.43.

Gopalan, Bhuvaneswari. 2020. "Mice Algorithm to Impute Missing Values in a Dataset." *Numpy Ninja*. Numpy Ninja. https://www.numpyninja.com/post/mice-algorithm-to-impute-missing-values-in-a-dataset.