Identifying Conducive Institutional Determinants of Entrepreneurial and Intrapreneurial Activity using Robust Elastic Net Regression

Research Compendium

With the supervision of dr. W. Liebregts

Frank Ritchi

1025402 (TU/e), 929981 (TiU)

f.a.ritchi@student.tue.nl

Eindhoven University of Technology / Tilburg University

Jheronimus Academy of Data Science

15th of June 2020

1. Contact

Author:

Name: F.A. Ritchi

Address: Geldersestraat 34B

6136 AT Sittard

The Netherlands

Tel: 0031610820699

Email: f.a. ritchi@student.tue.nl

Access to Data and Code:

GitHub: https://tinyurl.com/y74ubsxg

2. Data Reproducibility

Data Sources

GEM Adult Population Survey – National Level

URL: https://www.gemconsortium.org/data/sets?id=aps

Filenames:

- 1425660576GEM_2011_APS_Global_National_Level_Data_1Feb2015.sav
- 1422549452GEM_2012_APS_Global_National_Level_Data_1Feb2015.sav
- 1422390399GEM_2013_APS_Global_National_Level_Data_1Feb2015.sav
- GEM 2014 APS Global National Level Data_9Mar2016.sav
- GEM 2015 APS Global National Level Data_6Dec2016.sav

Last accessed: 10th of June 2020

GEM National Expert Survey – National Level

URL: https://www.gemconsortium.org/data/sets?id=nes

Filenames:

- NES AGGREGATED NATIONS.sav
- 2012_GEM_NES_NATIONAL LEVEL_1.sav
- GEM 2013 NES Global National Level Data rev14052014.sav
- GEM 2014 NES NATIONAL LEVEL_HARMONIZED_2.sav
- CORRECTED_GEM 2015 NES NATIONAL LEVEL.sav

Last accessed: 10th of June 2020

WEF Global Competitiveness Index

URL: http://hdr.undp.org/en/data

Filename:

- GCI_Dataset_2007-2017.xlsx

Last accessed: 10th of June 2020

UNDP Human Development Index

URL:http://reports.weforum.org/global-competitiveness-index-2016-2017/downloads/?doing_wp_cron=1592173769.3871660232543945312500

Filename:

- Human Development Index (HDI).csv

Last accessed: 10th of June 2020

3. Full Code (Dataset and Elastic Net Models)

```
######################
######################
##                  ##
##      DATASET     ##
##                  ##
######################
######################


#########################
# STEP 1: PRELIMINARIES #
#########################

# Install packages
install.packages("car")
install.packages("data.table")
install.packages("ggcorrplot")
install.packages("ggplot2")
install.packages("haven")
install.packages("Hmisc")
install.packages("naniar")
install.packages("readxl")
install.packages("reshape2")

# Load libraries
library(car)
library(data.table)
library(ggcorrplot)
library(ggplot2)
library(haven)
library(Hmisc)
library(naniar)
library(readxl)
library(reshape2)

# set seed
set.seed(42)

# Set temporary working directory
setwd("C:/Users/s161386/OneDrive - TU Eindhoven/Bachelor End Project/Model/Data/")
```

```
########################
# STEP 2: GEM APS DATA  #
########################

# Convert files from SPSS to CSV and read APS datasets
APS2011_SAV <- read_sav("1425660576GEM_2011_APS_Global_National_Level_Data_1Feb2015.sav")
write.csv(APS2011_SAV,"APS2011.csv")
APS2011 <- read.csv("APS2011.csv")

APS2012_SAV <- read_sav("1422549452GEM_2012_APS_Global_National_Level_Data_1Feb2015.sav")
write.csv(APS2012_SAV,"APS2012.csv")
APS2012 <- read.csv("APS2012.csv")

APS2013_SAV <- read_sav("1422390399GEM_2013_APS_Global_National_Level_Data_1Feb2015.sav")
write.csv(APS2013_SAV,"APS2013.csv")
APS2013 <- read.csv("APS2013.csv")

APS2014_SAV <- read_sav("GEM 2014 APS Global National Level Data_9Mar2016.sav")
write.csv(APS2014_SAV,"APS2014.csv")
APS2014 <- read.csv("APS2014.csv")

APS2015_SAV <- read_sav("GEM 2015 APS Global National Level Data_6Dec2016.sav")
write.csv(APS2015_SAV,"APS2015.csv")
APS2015 <- read.csv("APS2015.csv")

# Select and rename relevant variables and add variable 'Year'
APS2011 <- cbind(APS2011[c("COUNTRY_NAME", "TEA11", "IPACTLD_ALL")],Year =
rep("2011",nrow(APS2011)))
setnames(APS2011, old = c('COUNTRY_NAME','TEA11','IPACTLD_ALL'), new =
c('Country','TEA','EEA'))

APS2012 <- cbind(APS2012[c("country_name", "TEA12", "IPACTLD_ALL")],Year =
rep("2012",nrow(APS2012)))
setnames(APS2012, old = c('country_name','TEA12','IPACTLD_ALL'), new =
c('Country','TEA','EEA'))

APS2013 <- cbind(APS2013[c("country_name", "TEA13", "IPACTLD_ALL")],Year =
rep("2013",nrow(APS2013)))
setnames(APS2013, old = c('country_name','TEA13','IPACTLD_ALL'), new =
c('Country','TEA','EEA'))

APS2014 <- cbind(APS2014[c("country_name", "TEA14", "IPACTLD_ALL")],Year =
rep("2014",nrow(APS2014)))
setnames(APS2014, old = c('country_name','TEA14','IPACTLD_ALL'), new =
c('Country','TEA','EEA'))

APS2015 <- cbind(APS2015[c("country_name", "TEA15", "IPACTLD_ALL")],Year =
rep("2015",nrow(APS2015)))
setnames(APS2015, old = c('country_name','TEA15','IPACTLD_ALL'), new =
c('Country','TEA','EEA'))

# Merge and sort the datasets
APS <- rbind(APS2011, APS2012, APS2013, APS2014, APS2015)

# Change country names to remove duplicates
levels(APS$Country) <- c(levels(APS$Country), "South Korea", "Czechia", "North Macedonia",
"Trinidad and Tobago")
APS$Country[APS$Country == "Korea"] <- "South Korea"
APS$Country[APS$Country == "Bosnia"] <- "Bosnia and Herzegovina"
APS$Country[APS$Country == "USA"] <- "United States"
APS$Country[APS$Country == "Czech Republic"] <- "Czechia"
APS$Country[APS$Country == "Macedonia"] <- "North Macedonia"
APS$Country[APS$Country == "Trinidad & Tobago"] <- "Trinidad and Tobago"

# Check for, identify and remove missing values (present in column 'EEA' and 2 empty rows)
sum(is.na(APS))
APS[rowSums(is.na(APS)) > 0,]
APS <- na.omit(APS)

# Turn factors into characters
APS$Country <- as.character(APS$Country)
APS$Year <- as.character(APS$Year)
```

```r
#####################
# STEP 3: HDI DATA  #
#####################

# Read in files from CSV
HDI <- read.csv("Human Development Index (HDI).csv")

# Subset relevant columns
HDI <- HDI[,c(2,24,25,26,27,28)]

# Rename columns of years
for (col in 1:ncol(HDI)){
  colnames(HDI)[col] <-  sub("X", "", colnames(HDI)[col])}

# Transpose data and rename column names
HDI <- melt(HDI, id="Country")
setnames(HDI, old = c('variable', 'value'), new = c('Year', 'HDI'))

# Substitute country names to be similar to APS dataset
levels(HDI$Country) <- c(levels(HDI$Country), unique(APS$Country))
HDI$Country[HDI$Country == "Bolivia (Plurinational State of)"] <- "Bolivia"
HDI$Country[HDI$Country == "Iran (Islamic Republic of)"] <- "Iran"
HDI$Country[HDI$Country == "Palestine, State of"] <- "Palestine"
HDI$Country[HDI$Country == "Russian Federation"] <- "Russia"
HDI$Country[HDI$Country == "Korea (Republic of)"] <- "South Korea"
HDI$Country[HDI$Country == "Venezuela (Bolivarian Republic of)"] <- "Venezuela"
HDI$Country[HDI$Country == "Viet Nam"] <- "Vietnam"

# Keep only countries also present in APS dataset
HDI <- HDI[HDI$Country %in% APS$Country, ]

# Turn factors into characters or numerics
HDI$Country <- as.character(HDI$Country)
HDI$Year <- as.character(HDI$Year)
HDI$HDI <- as.numeric(HDI$HDI)

# Check for missing values. There are none.
sum(is.na(HDI))


#########################
# STEP 4: GEM NES DATA  #
#########################

# Convert files from SPSS to CSV
NES2011_SAV <- read_sav("2011 GEM NES AGGREGATED NATIONS.sav")
write.csv(NES2011_SAV,"NES2011.csv")
NES2011 <- read.csv("NES2011.csv")

NES2012_SAV <- read_sav("2012_GEM_NES_NATIONAL LEVEL_1.sav")
write.csv(NES2012_SAV,"NES2012.csv")
NES2012 <- read.csv("NES2012.csv")

NES2013_SAV <- read_sav("GEM 2013 NES Global National Level Data rev14052014.sav")
write.csv(NES2013_SAV,"NES2013.csv")
NES2013 <- read.csv("NES2013.csv")

NES2014_SAV <- read_sav("GEM 2014 NES NATIONAL LEVEL_HARMONIZED_2.sav")
write.csv(NES2014_SAV,"NES2014.csv")
NES2014 <- read.csv("NES2014.csv")

NES2015_SAV <- read_sav("CORRECTED_GEM 2015 NES NATIONAL LEVEL.sav")
write.csv(NES2015_SAV,"NES2015.csv")
NES2015 <- read.csv("NES2015.csv")

# Rename relevant variables
setnames(NES2011, old = 'NES11CRNAME', new = 'Country')
for (col in 1:ncol(NES2011)){
  colnames(NES2011)[col] <-  sub("11", "", colnames(NES2011)[col])
  colnames(NES2011)[col] <-  sub("_MEAN", "", colnames(NES2011)[col])}

setnames(NES2012, old = 'NES12RNAME', new = 'Country')
for (col in 1:ncol(NES2012)){
  colnames(NES2012)[col] <-  sub("12", "", colnames(NES2012)[col])
  colnames(NES2012)[col] <-  sub("_MEAN", "", colnames(NES2012)[col])}
```

```r
setnames(NES2013, old = 'NES13RNAME', new = 'Country')
for (col in 1:ncol(NES2013)){
  colnames(NES2013)[col] <-  sub("13", "", colnames(NES2013)[col])
  colnames(NES2013)[col] <-  sub("_MEAN", "", colnames(NES2013)[col])}

setnames(NES2014, old = 'NES14RNAME', new = 'Country')
for (col in 1:ncol(NES2014)){
  colnames(NES2014)[col] <-  sub("14", "", colnames(NES2014)[col])
  colnames(NES2014)[col] <-  sub("_MEAN", "", colnames(NES2014)[col])}

setnames(NES2015, old = 'NES15RNAME', new = 'Country')
for (col in 1:ncol(NES2015)){
  colnames(NES2015)[col] <-  sub("15", "", colnames(NES2015)[col])
  colnames(NES2015)[col] <-  sub("_MEAN9", "", colnames(NES2015)[col])}

# Select relevant variables and add variable 'Year'
NES_Variables <- c(
  "Country", "NES_A01", "NES_A02", "NES_A03", "NES_A04", "NES_A05", "NES_A06",
  "NES_B01", "NES_B02", "NES_B03", "NES_B04", "NES_B05", "NES_B06", "NES_B07",
  "NES_C01", "NES_C02", "NES_C03", "NES_C04", "NES_C05", "NES_C06",
  "NES_D01", "NES_D02", "NES_D03", "NES_D04", "NES_D05", "NES_D06",
  "NES_E01", "NES_E02", "NES_E03", "NES_E04", "NES_E05", "NES_E06",
  "NES_F01", "NES_F02", "NES_F03", "NES_F04", "NES_F05",
  "NES_G01", "NES_G02", "NES_G03", "NES_G04", "NES_G05", "NES_G06",
  "NES_H01", "NES_H02", "NES_H03", "NES_H04", "NES_H05",
  "NES_I01", "NES_I02", "NES_I03", "NES_I04", "NES_I05")

NES2011 <- cbind(NES2011[NES_Variables],Year = rep("2011",nrow(NES2011)))
NES2012 <- cbind(NES2012[NES_Variables],Year = rep("2012",nrow(NES2012)))
NES2013 <- cbind(NES2013[NES_Variables],Year = rep("2013",nrow(NES2013)))
NES2014 <- cbind(NES2014[NES_Variables],Year = rep("2014",nrow(NES2014)))
NES2015 <- cbind(NES2015[NES_Variables],Year = rep("2015",nrow(NES2015)))

# Merge and sort the datasets
NES <- rbind(NES2011, NES2012, NES2013, NES2014, NES2015)
NES <- subset(NES, select=c(1,54,2:53))

# Substitute country names to be similar to APS dataset
levels(NES$Country) <- c(levels(NES$Country), unique(APS$Country))
NES$Country[NES$Country == "ALGERIA"] <- "Algeria"
NES$Country[NES$Country == "ANGOLA"] <- "Angola"
NES$Country[NES$Country == "ARGENTINA"] <- "Argentina"
NES$Country[NES$Country == "AUSTRALIA"] <- "Australia"
NES$Country[NES$Country == "AUSTRIA"] <- "Austria"
NES$Country[NES$Country == "BANGLADESH"] <- "Bangladesh"
NES$Country[NES$Country == "BARBADOS"] <- "Barbados"
NES$Country[NES$Country == "BELGIUM"] <- "Belgium"
NES$Country[NES$Country == "BELIZE"] <- "Belize"
NES$Country[NES$Country == "BOLIVIA"] <- "Bolivia"
NES$Country[NES$Country == "BOSNIA & HZ"] <- "Bosnia and Herzegovina"
NES$Country[NES$Country == "BOTSWANA"] <- "Botswana"
NES$Country[NES$Country == "BRAZIL"] <- "Brazil"
NES$Country[NES$Country == "BULGARIA"] <- "Bulgaria"
NES$Country[NES$Country == "BURKINA FASO"] <- "Burkina Faso"
NES$Country[NES$Country == "CAMEROON"] <- "Cameroon"
NES$Country[NES$Country == "CANADA"] <- "Canada"
NES$Country[NES$Country == "CHILE"] <- "Chile"
NES$Country[NES$Country == "CHINA"] <- "China"
NES$Country[NES$Country == "COLOMBIA"] <- "Colombia"
NES$Country[NES$Country == "COSTA RICA"] <- "Costa Rica"
NES$Country[NES$Country == "CROATIA"] <- "Croatia"
NES$Country[NES$Country == "CZECH REP"] <- "Czechia"
NES$Country[NES$Country == "CZECH RP"] <- "Czechia"
NES$Country[NES$Country == "DENMARK"] <- "Denmark"
NES$Country[NES$Country == "ECUADOR"] <- "Ecuador"
NES$Country[NES$Country == "EGYPT"] <- "Egypt"
NES$Country[NES$Country == "EL SALVADOR"] <- "El Salvador"
NES$Country[NES$Country == "ESTONIA"] <- "Estonia"
NES$Country[NES$Country == "ETHIOPIA"] <- "Ethiopia"
NES$Country[NES$Country == "FINLAND"] <- "Finland"
NES$Country[NES$Country == "FRANCE"] <- "France"
NES$Country[NES$Country == "GEORGIA"] <- "Georgia"
NES$Country[NES$Country == "GERMANY"] <- "Germany"
NES$Country[NES$Country == "GREECE"] <- "Greece"
NES$Country[NES$Country == "GUATEMALA"] <- "Guatemala"
NES$Country[NES$Country == "HUNGARY"] <- "Hungary"
NES$Country[NES$Country == "INDIA"] <- "India"
```

```r
NES$Country[NES$Country == "INDONESIA"] <- "Indonesia"
NES$Country[NES$Country == "IRAN"] <- "Iran"
NES$Country[NES$Country == "IRELAND"] <- "Ireland"
NES$Country[NES$Country == "ISRAEL"] <- "Israel"
NES$Country[NES$Country == "ITALY"] <- "Italy"
NES$Country[NES$Country == "JAMAICA"] <- "Jamaica"
NES$Country[NES$Country == "JAPAN"] <- "Japan"
NES$Country[NES$Country == "KAZAKHSTAN"] <- "Kazakhstan"
NES$Country[NES$Country == "KOREA SR"] <- "South Korea"
NES$Country[NES$Country == "KOSOVO"] <- "Kosovo"
NES$Country[NES$Country == "LATVIA"] <- "Latvia"
NES$Country[NES$Country == "LEBANON"] <- "Lebanon"
NES$Country[NES$Country == "LITHUANIA"] <- "Lithuania"
NES$Country[NES$Country == "LUXEMBOURG"] <- "Luxembourg"
NES$Country[NES$Country == "MACEDONIA"] <- "North Macedonia"
NES$Country[NES$Country == "MALAYSIA"] <- "Malaysia"
NES$Country[NES$Country == "MEXICO"] <- "Mexico"
NES$Country[NES$Country == "MOROCCO"] <- "Morocco"
NES$Country[NES$Country == "NAMIBIA"] <- "Namibia"
NES$Country[NES$Country == "NETEHERLANDS"] <- "Netherlands"
NES$Country[NES$Country == "NIGERIA"] <- "Nigeria"
NES$Country[NES$Country == "NORWAY"] <- "Norway"
NES$Country[NES$Country == "PAKISTAN"] <- "Pakistan"
NES$Country[NES$Country == "PALESTINE"] <- "Palestine"
NES$Country[NES$Country == "PANAMA"] <- "Panama"
NES$Country[NES$Country == "PERU"] <- "Peru"
NES$Country[NES$Country == "PHILIPPINES"] <- "Philippines"
NES$Country[NES$Country == "POLAND"] <- "Poland"
NES$Country[NES$Country == "PORTUGAL"] <- "Portugal"
NES$Country[NES$Country == "PUERTO RICO"] <- "Puerto Rico"
NES$Country[NES$Country == "QATAR"] <- "Qatar"
NES$Country[NES$Country == "ROMANIA"] <- "Romania"
NES$Country[NES$Country == "RUSSIA"] <- "Russia"
NES$Country[NES$Country == "SENEGAL"] <- "Senegal"
NES$Country[NES$Country == "SINGAPORE"] <- "Singapore"
NES$Country[NES$Country == "SLOVAKIA"] <- "Slovakia"
NES$Country[NES$Country == "SLOVENIA"] <- "Slovenia"
NES$Country[NES$Country == "SOUTH AFRICA"] <- "South Africa"
NES$Country[NES$Country == "SOUTH KOREA"] <- "South Korea"
NES$Country[NES$Country == "SPAIN"] <- "Spain"
NES$Country[NES$Country == "SURINAME"] <- "Suriname"
NES$Country[NES$Country == "SWEDEN"] <- "Sweden"
NES$Country[NES$Country == "SWITZERLAND"] <- "Switzerland"
NES$Country[NES$Country == "TAIWAN"] <- "Taiwan"
NES$Country[NES$Country == "THAILAND"] <- "Thailand"
NES$Country[NES$Country == "TRINIDAD&T"] <- "Trinidad and Tobago"
NES$Country[NES$Country == "TUNISIA"] <- "Tunisia"
NES$Country[NES$Country == "TURKEY"] <- "Turkey"
NES$Country[NES$Country == "UAE"] <- "United Arab Emirates"
NES$Country[NES$Country == "UGANDA"] <- "Uganda"
NES$Country[NES$Country == "UK"] <- "United Kingdom"
NES$Country[NES$Country == "URUGUAY"] <- "Uruguay"
NES$Country[NES$Country == "USA"] <- "United States"
NES$Country[NES$Country == "VENEZUELA"] <- "Venezuela"
NES$Country[NES$Country == "VIETNAM"] <- "Vietnam"

# Keep only countries also present in APS dataset
NES <- NES[NES$Country %in% APS$Country, ]

# Turn factors into characters
NES$Country <- as.character(NES$Country)
NES$Year <- as.character(NES$Year)

# Check for missing values. There are none.
sum(is.na(NES))


#########################
# STEP 5: WEF GCI DATA  #
#########################

# Read dataset from Excel sheet and transform to CSV format
GCI_XLSX <- read_excel("GCI_Dataset_2007-2017.xlsx", range = "Data!A4:FL6527")
write.csv(GCI_XLSX,"GCI.csv")
GCI <- read.csv("GCI.csv")

# Adapt variable names
```

```r
setnames(GCI, old = c('Edition', 'Code.GCR'), new = c('Year', 'GCI_Variable'))

# Subset the actual GCI values and relevant years
GCI <- GCI[which(GCI$Attribute == 'Value'),]
GCI$Year <- substr(GCI$Year,0,4)
GCI <- GCI[which((GCI$Year>'2010')&(GCI$Year<'2016')),]

# Remove irrelevant and supranational data
GCI <- GCI[,-c(1,2,3,5,7,8,9,(ncol(GCI)-7):ncol(GCI))]

# Rounding errors present in the variable names in the original dataset need to be replaced.
levels(GCI$GCI_Variable) <- c(levels(GCI$GCI_Variable), c(
  "1.09", "1.10", "1.11", "1.12", "1.13", "1.14", "1.15", "1.16", "1.20",
  "2.01", "2.03", "2.05", "2.07",
  "4.02", "4.06", "4.10",
  "5.02", "5.06",
  "6.10",
  "7.10",
  "8.03", "8.04", "8.05",
  "9.03", "9.04", "9.05",
  "10.03", "10.04"))

GCI$GCI_Variable[GCI$GCI_Variable == "1.0900000000000001"] <- "1.09"
GCI$GCI_Variable[GCI$GCI_Variable == "1.1000000000000001"] <- "1.10"
GCI$GCI_Variable[GCI$GCI_Variable == "1.1100000000000001"] <- "1.11"
GCI$GCI_Variable[GCI$GCI_Variable == "1.1200000000000001"] <- "1.12"
GCI$GCI_Variable[GCI$GCI_Variable == "1.1299999999999999"] <- "1.13"
GCI$GCI_Variable[GCI$GCI_Variable == "1.1399999999999999"] <- "1.14"
GCI$GCI_Variable[GCI$GCI_Variable == "1.1499999999999999"] <- "1.15"
GCI$GCI_Variable[GCI$GCI_Variable == "1.1599999999999999"] <- "1.16"
GCI$GCI_Variable[GCI$GCI_Variable == "1.2"] <- "1.20"
GCI$GCI_Variable[GCI$GCI_Variable == "2.0099999999999998"] <- "2.01"
GCI$GCI_Variable[GCI$GCI_Variable == "2.0299999999999998"] <- "2.03"
GCI$GCI_Variable[GCI$GCI_Variable == "2.0499999999999998"] <- "2.05"
GCI$GCI_Variable[GCI$GCI_Variable == "2.0699999999999998"] <- "2.07"
GCI$GCI_Variable[GCI$GCI_Variable == "4.0199999999999996"] <- "4.02"
GCI$GCI_Variable[GCI$GCI_Variable == "4.0599999999999996"] <- "4.06"
GCI$GCI_Variable[GCI$GCI_Variable == "4.0999999999999996"] <- "4.10"
GCI$GCI_Variable[GCI$GCI_Variable == "5.0199999999999996"] <- "5.02"
GCI$GCI_Variable[GCI$GCI_Variable == "5.0599999999999996"] <- "5.06"
GCI$GCI_Variable[GCI$GCI_Variable == "6.1"] <- "6.10"
GCI$GCI_Variable[GCI$GCI_Variable == "7.1"] <- "7.10"
GCI$GCI_Variable[GCI$GCI_Variable == "8.0299999999999994"] <- "8.03"
GCI$GCI_Variable[GCI$GCI_Variable == "8.0399999999999991"] <- "8.04"
GCI$GCI_Variable[GCI$GCI_Variable == "8.0500000000000007"] <- "8.05"
GCI$GCI_Variable[GCI$GCI_Variable == "9.0299999999999994"] <- "9.03"
GCI$GCI_Variable[GCI$GCI_Variable == "9.0399999999999991"] <- "9.04"
GCI$GCI_Variable[GCI$GCI_Variable == "9.0500000000000007"] <- "9.05"
GCI$GCI_Variable[GCI$GCI_Variable == "10.029999999999999"] <- "10.03"
GCI$GCI_Variable[GCI$GCI_Variable == "10.039999999999999"] <- "10.04"

# make lists of the relevant variables and subset the dataset
Relevant_GCI_Variables <- c(
  "1.01","1.02","1.03","1.04","1.05","1.06","1.07","1.08","1.09", "1.10", "1.11",
  "1.12", "1.13", "1.14", "1.15", "1.16", "1.17", "1.18", "1.19", "1.20", "1.21",
  "2.01","2.02","2.03","2.04","2.05","2.06","2.07","2.08","2.09",
  "3.01","3.02","3.03","3.04","3.05",
  "4.01","4.02","4.03","4.04","4.05","4.06","4.07","4.08","4.09",
  "5.01","5.02","5.03","5.04","5.05","5.06","5.07","5.08",
  "6.01","6.02","6.03","6.04","6.05","6.06","6.07","6.08","6.09", "6.10", "6.11",
  "6.12", "6.13", "6.14", "6.15", "6.16",
  "7.01","7.02","7.03","7.04","7.05","7.06","7.07","7.08","7.09", "7.10",
  "8.01","8.02","8.03","8.04","8.05","8.06","8.07","8.08",
  "9.01","9.02","9.03","9.04","9.05","9.06","9.07",
  "10.01","10.02","10.03","10.04",
  "11.01","11.02","11.03","11.04","11.05","11.06","11.07","11.08","11.09",
  "12.01","12.02","12.03","12.04","12.05","12.06","12.07")

GCI <- subset(GCI, subset = (GCI$GCI_Variable %in% Relevant_GCI_Variables))

# Turn factors into characters and adapt variable names
GCI$GCI_Variable=paste0("GCI_",GCI$GCI_Variable)

# Adapt country names to match other datasets
setnames(GCI, old = c('Bosnia.and.Herzegovina', 'Burkina.Faso','Costa.Rica', 'Czech.Republic',
                      'El.Salvador', 'Iran..Islamic.Rep.', 'Korea..Rep.', 'Macedonia..FYR',
                      'Russian.Federation', 'Slovak.Republic', 'South.Africa',
```

```r
                          'Trinidad.and.Tobago', 'United.Arab.Emirates', 'United.Kingdom',
                          'United.States', 'Viet.Nam'),
         new = c('Bosnia and Herzegovina','Burkina Faso','Costa Rica', 'Czechia',
                 'El Salvador', 'Iran', 'South Korea', 'North Macedonia',
                 'Russia','Slovakia', 'South Africa',
                 'Trinidad and Tobago', 'United Arab Emirates', 'United Kingdom',
                 'United States', 'Vietnam'))

# Remove countries that are not in other datasets
GCI_Country_Year <- GCI[,c(1,2)]
GCI_Variables <- GCI[,colnames(GCI) %in% APS$Country]
GCI <- cbind(GCI_Country_Year, GCI_Variables)

# Some columns are of type logical and cannot be converted to numeric and need to be removed
GCI_Logical <- sapply(GCI, is.logical)
GCI <- (GCI[!GCI_Logical])

# Transpose data and rename column names
GCI_melt <- melt(GCI, id.vars = c("Year","GCI_Variable"))
GCI <- dcast(GCI_melt, Year+variable~GCI_Variable)
setnames(GCI, old = 'variable', new = 'Country')

# Set all values (temporarily) to character values
GCI[c(2:ncol(GCI))] <- sapply(GCI[c(2:ncol(GCI))],as.character)

# Regular expression to identify all types of non-numerical values
num <- function(x) grepl("^[-]*[0-9][0-9]*[.]*[0-9]*$", x)

# Apply regular expression to all columns and store all non-numerical values
non_numeric <- list()
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.01[num(GCI$GCI_1.01) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.02[num(GCI$GCI_1.02) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.03[num(GCI$GCI_1.03) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.04[num(GCI$GCI_1.04) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.05[num(GCI$GCI_1.05) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.06[num(GCI$GCI_1.06) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.07[num(GCI$GCI_1.07) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.08[num(GCI$GCI_1.08) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.09[num(GCI$GCI_1.09) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.10[num(GCI$GCI_1.10) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.11[num(GCI$GCI_1.11) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.12[num(GCI$GCI_1.12) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.13[num(GCI$GCI_1.13) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.14[num(GCI$GCI_1.14) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.15[num(GCI$GCI_1.15) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.16[num(GCI$GCI_1.16) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.17[num(GCI$GCI_1.17) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.18[num(GCI$GCI_1.18) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.19[num(GCI$GCI_1.19) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.20[num(GCI$GCI_1.20) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_1.21[num(GCI$GCI_1.21) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_2.01[num(GCI$GCI_2.01) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_2.02[num(GCI$GCI_2.02) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_2.03[num(GCI$GCI_2.03) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_2.04[num(GCI$GCI_2.04) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_2.05[num(GCI$GCI_2.05) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_2.06[num(GCI$GCI_2.06) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_2.07[num(GCI$GCI_2.07) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_2.08[num(GCI$GCI_2.08) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_2.09[num(GCI$GCI_2.09) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_3.01[num(GCI$GCI_3.01) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_3.02[num(GCI$GCI_3.02) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_3.03[num(GCI$GCI_3.03) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_3.04[num(GCI$GCI_3.04) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_3.05[num(GCI$GCI_3.05) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_4.01[num(GCI$GCI_4.01) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_4.02[num(GCI$GCI_4.02) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_4.03[num(GCI$GCI_4.03) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_4.04[num(GCI$GCI_4.04) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_4.05[num(GCI$GCI_4.05) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_4.06[num(GCI$GCI_4.06) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_4.07[num(GCI$GCI_4.07) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_4.08[num(GCI$GCI_4.08) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_4.09[num(GCI$GCI_4.09) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_5.01[num(GCI$GCI_5.01) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_5.02[num(GCI$GCI_5.02) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_5.03[num(GCI$GCI_5.03) == FALSE])))
```

```r
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_5.04[num(GCI$GCI_5.04) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_5.05[num(GCI$GCI_5.05) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_5.06[num(GCI$GCI_5.06) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_5.07[num(GCI$GCI_5.07) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_5.08[num(GCI$GCI_5.08) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.01[num(GCI$GCI_6.01) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.02[num(GCI$GCI_6.02) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.03[num(GCI$GCI_6.03) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.04[num(GCI$GCI_6.04) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.05[num(GCI$GCI_6.05) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.06[num(GCI$GCI_6.06) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.07[num(GCI$GCI_6.07) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.08[num(GCI$GCI_6.08) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.09[num(GCI$GCI_6.09) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.10[num(GCI$GCI_6.10) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.11[num(GCI$GCI_6.11) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.12[num(GCI$GCI_6.12) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.13[num(GCI$GCI_6.13) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.14[num(GCI$GCI_6.14) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.15[num(GCI$GCI_6.15) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_6.16[num(GCI$GCI_6.16) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_7.01[num(GCI$GCI_7.01) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_7.02[num(GCI$GCI_7.02) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_7.03[num(GCI$GCI_7.03) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_7.04[num(GCI$GCI_7.04) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_7.05[num(GCI$GCI_7.05) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_7.06[num(GCI$GCI_7.06) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_7.07[num(GCI$GCI_7.07) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_7.08[num(GCI$GCI_7.08) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_7.09[num(GCI$GCI_7.09) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_7.10[num(GCI$GCI_7.10) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_8.01[num(GCI$GCI_8.01) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_8.02[num(GCI$GCI_8.02) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_8.03[num(GCI$GCI_8.03) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_8.04[num(GCI$GCI_8.04) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_8.05[num(GCI$GCI_8.05) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_8.06[num(GCI$GCI_8.06) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_8.07[num(GCI$GCI_8.07) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_8.08[num(GCI$GCI_8.08) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_9.01[num(GCI$GCI_9.01) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_9.02[num(GCI$GCI_9.02) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_9.03[num(GCI$GCI_9.03) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_9.04[num(GCI$GCI_9.04) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_9.05[num(GCI$GCI_9.05) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_9.06[num(GCI$GCI_9.06) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_9.07[num(GCI$GCI_9.07) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_10.01[num(GCI$GCI_10.01) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_10.02[num(GCI$GCI_10.02) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_10.03[num(GCI$GCI_10.03) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_10.04[num(GCI$GCI_10.04) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_11.01[num(GCI$GCI_11.01) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_11.02[num(GCI$GCI_11.02) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_11.03[num(GCI$GCI_11.03) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_11.04[num(GCI$GCI_11.04) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_11.05[num(GCI$GCI_11.05) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_11.06[num(GCI$GCI_11.06) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_11.07[num(GCI$GCI_11.07) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_11.08[num(GCI$GCI_11.08) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_11.09[num(GCI$GCI_11.09) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_12.01[num(GCI$GCI_12.01) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_12.02[num(GCI$GCI_12.02) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_12.03[num(GCI$GCI_12.03) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_12.04[num(GCI$GCI_12.04) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_12.05[num(GCI$GCI_12.05) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_12.06[num(GCI$GCI_12.06) == FALSE])))
non_numeric <- unique(append(non_numeric, unique(GCI$GCI_12.07[num(GCI$GCI_12.07) == FALSE])))

# Some identified values are in fact numerical (e.g. 1.2E-2) and will be retained.
non_num <- function(x) !grepl("[E][-][0-9]$", x)
non_numeric_values <- unique(non_numeric)[non_num(unique(non_numeric))]

# All other values will be changed in NA.
GCI <- replace_with_na_all(GCI, condition = ~. %in% non_numeric_values)
GCI <- as.data.frame(GCI)

# There are a significant amount of NA's, most of them in just a few rows and columns.
sum(is.na(GCI))
```

```
sort(rowSums(is.na(GCI)),  decreasing = TRUE)
sort(colSums(is.na(GCI)),  decreasing = TRUE)

# Removal of rows and columns with many NA's. The few remaining NA's too will be omitted.
GCI <- GCI[which(rowMeans(is.na(GCI)) < 0.20),which(colMeans(is.na(GCI)) < 0.20)]
sum(is.na(GCI))
GCI <- na.omit(GCI)

# Turn characters into numerics
GCI_Variables <- colnames(GCI)[3:(length(colnames(GCI)))]
GCI[GCI_Variables] <- sapply(GCI[GCI_Variables],as.numeric)

# Sort column names
GCI <- GCI[,c(1:23, 43:104,24:42)]


############################
# STEP 6: MERGING DATASETS  #
############################

# Merge APS and HDI datasets
APS_HDI <- merge(APS, HDI, by.x = c("Country", "Year"), all.x = TRUE)

# Check for, identify, and remove missing values
# (3 countries present in APS data, but not in HDI Data)
sum(is.na(APS_HDI))
APS_HDI[rowSums(is.na(APS_HDI)) > 0,]
APS_HDI <- (na.omit(APS_HDI))

# Merge NES dataset with APS and HDI datasets
APS_HDI_NES <- merge(APS_HDI, NES, by.x = c("Country", "Year"), all.x = TRUE)

# Check for, identify, and remove missing values
# (6 instances in APS data, but not in NES Data)
sum(is.na(APS_HDI_NES))
APS_HDI_NES[rowSums(is.na(APS_HDI_NES)) > 0,]
APS_HDI_NES <- (na.omit(APS_HDI_NES))

# Merge GCI dataset with APS, HDI and NES datasets
Merged_Dataset <- merge(APS_HDI_NES, GCI, by.x = c("Country", "Year"), all.x = TRUE)

# Check for, identify, and remove missing values
# (51 instances in APS data, but not in GCI Data)
sum(is.na(Merged_Dataset))
Merged_Dataset[rowSums(is.na(Merged_Dataset)) > 0,]
Merged_Dataset <- (na.omit(Merged_Dataset))
Final_Dataset <- Merged_Dataset

########################
# STEP 7: CORRELATIONS #
########################

# Calculate correlations of dependent and control variables
corr_dependent <- rcorr(as.matrix(Final_Dataset[,3:5]))
p.mat <- cor_pmat(cor(Final_Dataset[,3:5]))

# Create a correlation matrix
flattenCorrMatrix <- function(cormat, pmat) {
  ut <- upper.tri(cormat)
  data.frame(
    row = rownames(cormat)[row(cormat)[ut]],
    column = rownames(cormat)[col(cormat)[ut]],
    cor  =(cormat)[ut],
    p = p.mat[ut])}

correlation_matrix_dep <- flattenCorrMatrix(corr_dependent$r, corr_dependent$P)
correlation_matrix_dep <- correlation_matrix_dep[order(correlation_matrix_dep$p),]
correlation_matrix_dep[,3:4] <- round(correlation_matrix_dep[,3:4],3)

# Create plot of average TEA and EEA rates per HDI cohort
Dep_vars <- Final_Dataset[,c(5,3,4)]
Dep_vars$HDI_Cohort <- cut(x=Dep_vars$HDI, breaks=seq(from=0, to=1, by = 0.1))
HDI_cohorts <- na.omit(cbind(tapply(Dep_vars$TEA, Dep_vars$HDI_Cohort, mean),
                             tapply(Dep_vars$EEA, Dep_vars$HDI_Cohort, mean)))
colnames(HDI_cohorts) <- c("TEA","EEA")
HDI_cohorts_melted <- melt(HDI_cohorts, varnames=c('HDI', 'Rate'))
colnames(HDI_cohorts_melted) <- c('HDI', 'Var', 'Rate')
```

```r
ggplot(HDI_cohorts_melted, aes(HDI, Rate, fill = Var)) + geom_bar(stat="identity", position =
"dodge") + theme(legend.title = element_blank())

# Calculate correlation of independent variables
corr_independent <- rcorr(as.matrix(Final_Dataset[,7:ncol(Final_Dataset)]))
p.mat <- cor_pmat(cor(Final_Dataset[,7:ncol(Final_Dataset)]))

# Create and export correlation matrix of independent variables
colors_axes <- ifelse(colnames(Final_Dataset[,7:ncol(Final_Dataset)]) %in% NES_Variables,
"#6D9EC1", "#E46726")
p.mat <- cor_pmat(cor(Final_Dataset[,7:ncol(Final_Dataset)]))
ggcorrplot(cor(Final_Dataset[,7:ncol(Final_Dataset)]), hc.order = FALSE, p.mat = p.mat,insig =
"blank", outline.col = "white", colors = c("#6D9EC1", "white", "#E46726"))  + theme_gray() +
theme(axis.text=element_text(size=2)) + theme(axis.text.x = element_text(angle = 90)) +
theme(axis.text.x = element_text(colour = colors_axes)) + theme(axis.text.y =
element_text(colour = colors_axes)) + theme(axis.title = element_blank())


########################
# STEP 8: EXPORT FILES #
########################

# Export Final Dataset as CSV-file
write.csv(Final_Dataset, "Final_Dataset.csv")

# Export plot of average TEA and EEA rates per HDI cohort
tiff("HDI_Cohorts.tiff", units="in", width=5, height=5, res=1800)
ggplot(HDI_cohorts_melted, aes(HDI, Rate, fill = Var)) + geom_bar(stat="identity", position =
"dodge") + theme(legend.title = element_blank())
dev.off()

# Export plot of correlation matrix of independent variables
tiff("Correlation_Matrix_Independent_Variables.tiff", units="in", width=5, height=5, res=1800)
ggcorrplot(cor(Merged_Dataset[,7:ncol(Merged_Dataset)]), hc.order = FALSE, p.mat = p.mat,insig
= "blank", outline.col = "white", colors = c("#6D9EC1", "white", "#E46726"))  + theme_gray() +
theme(axis.text=element_text(size=2)) + theme(axis.text.x = element_text(angle = 90, colour =
colors_axes)) + theme(axis.text.y = element_text(angle = 0, colour = colors_axes)) +
theme(axis.title = element_blank())
dev.off()



##########################
##########################
##                      ##
##  ELASTIC NET MODELS  ##
##                      ##
##########################
##########################


#########################
# STEP 1: PRELIMINARIES #
#########################

#Install packages
install.packages("enetLTS")
install.packages("glmnet")
install.packages("ggplot2")
install.packages("robustHD")
install.packages("grid")
install.packages("reshape")
install.packages("parallel")
install.packages("cvTools")
install.packages("stats")

# Load libraries
library(enetLTS)
library(glmnet)
library(ggplot2)
library(robustHD)
library(grid)
library(reshape)
library(parallel)
library(cvTools)
library(stats)
```

```
# Set seed for reproducibility
set.seed(42)

# Set temporary working directory
setwd("C:/Users/s161386/OneDrive - TU Eindhoven/Bachelor End Project/Model/Data/")

# Load dataset
Dataset <- read.csv("Final_Dataset.csv")


####################################
# STEP 2: DEFINE TRAIN & TEST DATA #
####################################

# Define dependent and indepedent variables and subset by HDI score
TEA_HDI_All  <- data.matrix(Dataset[,4])
EEA_HDI_All  <- data.matrix(Dataset[,5])
X_HDI_All    <- data.matrix(Dataset[,c(7:ncol(Dataset))])

TEA_HDI_High <- data.matrix(Dataset[Dataset$HDI > 0.800,][,4])
EEA_HDI_High <- data.matrix(Dataset[Dataset$HDI > 0.800,][,5])
X_HDI_High   <- data.matrix(Dataset[Dataset$HDI > 0.800,][,c(7:ncol(Dataset))])

TEA_HDI_Low  <- data.matrix(Dataset[Dataset$HDI < 0.801,][,4])
EEA_HDI_Low  <- data.matrix(Dataset[Dataset$HDI < 0.801,][,5])
X_HDI_Low    <- data.matrix(Dataset[Dataset$HDI < 0.801,][,c(7:ncol(Dataset))])

# Split data into training (70%) and testing (30%) the datasets, per HDI subset
Train_HDI_All  <- sample(1:nrow(X_HDI_All) , .70*nrow(X_HDI_All) , replace = FALSE)
Train_HDI_High <- sample(1:nrow(X_HDI_High), .70*nrow(X_HDI_High), replace = FALSE)
Train_HDI_Low  <- sample(1:nrow(X_HDI_Low) , .70*nrow(X_HDI_Low) , replace = FALSE)

# Training and test sets for TEA, EEA and independent variables, per HDI subset
TEA_HDI_All_Train  <- TEA_HDI_All[ Train_HDI_All]
TEA_HDI_All_Test   <- TEA_HDI_All[-Train_HDI_All]
EEA_HDI_All_Train  <- EEA_HDI_All[ Train_HDI_All]
EEA_HDI_All_Test   <- EEA_HDI_All[-Train_HDI_All]
X_HDI_All_Train    <- X_HDI_All[ Train_HDI_All,]
X_HDI_All_Test     <- X_HDI_All[-Train_HDI_All,]

TEA_HDI_High_Train <- TEA_HDI_High[ Train_HDI_High]
TEA_HDI_High_Test  <- TEA_HDI_High[-Train_HDI_High]
EEA_HDI_High_Train <- EEA_HDI_High[ Train_HDI_High]
EEA_HDI_High_Test  <- EEA_HDI_High[-Train_HDI_High]
X_HDI_High_Train   <- X_HDI_High[ Train_HDI_High,]
X_HDI_High_Test    <- X_HDI_High[-Train_HDI_High,]

TEA_HDI_Low_Train  <- TEA_HDI_Low[ Train_HDI_Low]
TEA_HDI_Low_Test   <- TEA_HDI_Low[-Train_HDI_Low]
EEA_HDI_Low_Train  <- EEA_HDI_Low[ Train_HDI_Low]
EEA_HDI_Low_Test   <- EEA_HDI_Low[-Train_HDI_Low]
X_HDI_Low_Train    <- X_HDI_Low[ Train_HDI_Low,]
X_HDI_Low_Test     <- X_HDI_Low[-Train_HDI_Low,]


#######################
# STEP 3: TRAIN MODELS #
#######################

# Fitting the models, with 10-fold cross-validation

# Model 1 & 2: All HDI - TEA & EEA
Fit_TEA_HDI_All  <- enetLTS(X_HDI_All_Train,  TEA_HDI_All_Train,  family = "gaussian", nfold =
10)
Fit_EEA_HDI_All  <- enetLTS(X_HDI_All_Train,  EEA_HDI_All_Train,  family = "gaussian", nfold =
10)

# Model 3 & 4: High HDI - TEA & EEA
Fit_TEA_HDI_High <- enetLTS(X_HDI_High_Train, TEA_HDI_High_Train, family = "gaussian", nfold =
10)
Fit_EEA_HDI_High <- enetLTS(X_HDI_High_Train, EEA_HDI_High_Train, family = "gaussian", nfold =
10)

# Model 5 & 6: Low HDI - TEA & EEA
Fit_TEA_HDI_Low  <- enetLTS(X_HDI_Low_Train,  TEA_HDI_Low_Train,  family = "gaussian", nfold =
10)
```

```
Fit_EEA_HDI_Low  <- enetLTS(X_HDI_Low_Train,  EEA_HDI_Low_Train,  family = "gaussian", nfold =
10)


################################
# STEP 4: RESULTS OF TRAINING #
################################

# Model 1: All HDI - TEA
Fit_TEA_HDI_All$num.nonzerocoef
Fit_TEA_HDI_All$a0
Fit_TEA_HDI_All$coefficients[Fit_TEA_HDI_All$coefficients > 0 | Fit_TEA_HDI_All$coefficients <
0]
Fit_TEA_HDI_All$fitted.values
Fit_TEA_HDI_All$residuals
Fit_TEA_HDI_All$rmse
length(Fit_TEA_HDI_All$wt[Fit_TEA_HDI_All$wt == 0])
plotCoef.enetLTS(Fit_TEA_HDI_All)
plotResid.enetLTS(Fit_TEA_HDI_All)
plotDiagnostic.enetLTS(Fit_TEA_HDI_All)

# Model 2: All HDI - EEA
Fit_EEA_HDI_All$num.nonzerocoef
Fit_EEA_HDI_All$a0
Fit_EEA_HDI_All$coefficients[Fit_EEA_HDI_All$coefficients > 0 | Fit_EEA_HDI_All$coefficients <
0]
Fit_EEA_HDI_All$fitted.values
Fit_EEA_HDI_All$residuals
Fit_EEA_HDI_All$rmse
length(Fit_EEA_HDI_All$wt[Fit_EEA_HDI_All$wt == 0])
plotCoef.enetLTS(Fit_EEA_HDI_All)
plotResid.enetLTS(Fit_EEA_HDI_All)
plotDiagnostic.enetLTS(Fit_EEA_HDI_All)

# Model 3: High HDI - TEA
Fit_TEA_HDI_High$num.nonzerocoef
Fit_TEA_HDI_High$a0
Fit_TEA_HDI_High$coefficients[Fit_TEA_HDI_High$coefficients > 0 |
Fit_TEA_HDI_High$coefficients < 0]
Fit_TEA_HDI_High$fitted.values
Fit_TEA_HDI_High$residuals
Fit_TEA_HDI_High$rmse
length(Fit_TEA_HDI_High$wt[Fit_TEA_HDI_High$wt == 0])
plotCoef.enetLTS(Fit_TEA_HDI_High)
plotResid.enetLTS(Fit_TEA_HDI_High)
plotDiagnostic.enetLTS(Fit_TEA_HDI_High)

# Model 4: High HDI - EEA
Fit_EEA_HDI_High$num.nonzerocoef
Fit_EEA_HDI_High$a0
Fit_EEA_HDI_High$coefficients[Fit_EEA_HDI_High$coefficients > 0 |
Fit_EEA_HDI_High$coefficients < 0]
Fit_EEA_HDI_High$fitted.values
Fit_EEA_HDI_High$residuals
Fit_EEA_HDI_High$rmse
length(Fit_EEA_HDI_High$wt[Fit_EEA_HDI_High$wt == 0])
plotCoef.enetLTS(Fit_EEA_HDI_High)
plotResid.enetLTS(Fit_EEA_HDI_High)
plotDiagnostic.enetLTS(Fit_EEA_HDI_High)

# Model 5: Low HDI - TEA
Fit_TEA_HDI_Low$num.nonzerocoef
Fit_TEA_HDI_Low$a0
Fit_TEA_HDI_Low$coefficients[Fit_TEA_HDI_Low$coefficients > 0 | Fit_TEA_HDI_Low$coefficients <
0]
Fit_TEA_HDI_Low$fitted.values
Fit_TEA_HDI_Low$residuals
Fit_TEA_HDI_Low$rmse
length(Fit_TEA_HDI_Low$wt[Fit_TEA_HDI_Low$wt == 0])
plotCoef.enetLTS(Fit_TEA_HDI_Low)
plotResid.enetLTS(Fit_TEA_HDI_Low)
plotDiagnostic.enetLTS(Fit_TEA_HDI_Low)

# Model 6: Low HDI - EEA
Fit_EEA_HDI_Low$num.nonzerocoef
Fit_EEA_HDI_Low$a0
```

```
Fit_EEA_HDI_Low$coefficients[Fit_EEA_HDI_Low$coefficients > 0 | Fit_EEA_HDI_Low$coefficients <
0]
Fit_EEA_HDI_Low$rmse
Fit_EEA_HDI_Low$fitted.values
Fit_EEA_HDI_Low$residuals
length(Fit_EEA_HDI_Low$wt[Fit_EEA_HDI_Low$wt== 0])
plotCoef.enetLTS(Fit_EEA_HDI_Low)
plotResid.enetLTS(Fit_EEA_HDI_Low)
plotDiagnostic.enetLTS(Fit_EEA_HDI_Low)


#######################
# STEP 5: TEST MODELS #
#######################

# Run fitted models on testing data

# Model 1 & 2: All HDI - TEA & EEA
Predicted_TEA_HDI_All  <- predict(Fit_TEA_HDI_All,  newX = X_HDI_All_Test,  type = "response")
Predicted_EEA_HDI_All  <- predict(Fit_EEA_HDI_All,  newX = X_HDI_All_Test,  type = "response")

# Model 3 & 4: High HDI - TEA & EEA
Predicted_TEA_HDI_High <- predict(Fit_TEA_HDI_High, newX = X_HDI_High_Test, type = "response")
Predicted_EEA_HDI_High <- predict(Fit_EEA_HDI_High, newX = X_HDI_High_Test, type = "response")

# Model 5 & 6: Low HDI - TEA & EEA
Predicted_TEA_HDI_Low  <- predict(Fit_TEA_HDI_Low,  newX = X_HDI_Low_Test,  type = "response")
Predicted_EEA_HDI_Low  <- predict(Fit_EEA_HDI_Low,  newX = X_HDI_Low_Test,  type = "response")


##############################
# STEP 6: RESULTS ON TESTING #
##############################

# Model 1 & 2: All HDI - TEA & EEA
TEA_HDI_All_Residuals_Test   <- TEA_HDI_All_Test-Predicted_TEA_HDI_All$reweighted.response[,1]
TEA_HDI_All_RMSE_Test    <- sqrt(mean((TEA_HDI_All_Test-
Predicted_TEA_HDI_All$reweighted.response[,1])^2))
TEA_HDI_All_RMSE_Test

EEA_HDI_All_Residuals_Test   <- EEA_HDI_All_Test-Predicted_EEA_HDI_All$reweighted.response[,1]
EEA_HDI_All_RMSE_Test    <- sqrt(mean((EEA_HDI_All_Test-
Predicted_EEA_HDI_All$reweighted.response[,1])^2))
EEA_HDI_All_RMSE_Test

# Model 3 & 4: High HDI - TEA & EEA
TEA_HDI_High_Residuals_Test  <- TEA_HDI_High_Test-
Predicted_TEA_HDI_High$reweighted.response[,1]
TEA_HDI_High_RMSE_Test   <- sqrt(mean((TEA_HDI_High_Test-
Predicted_TEA_HDI_High$reweighted.response[,1])^2))
TEA_HDI_High_RMSE_Test

EEA_HDI_High_Residuals_Test  <- EEA_HDI_High_Test-
Predicted_EEA_HDI_High$reweighted.response[,1]
EEA_HDI_High_RMSE_Test   <- sqrt(mean((EEA_HDI_High_Test-
Predicted_EEA_HDI_High$reweighted.response[,1])^2))
EEA_HDI_High_RMSE_Test

# Model 5 & 6: Low HDI - TEA & EEA
TEA_HDI_Low_Residuals_Test   <- TEA_HDI_Low_Test-Predicted_TEA_HDI_Low$reweighted.response[,1]
TEA_HDI_Low_RMSE_Test    <- sqrt(mean((TEA_HDI_Low_Test-
Predicted_TEA_HDI_Low$reweighted.response[,1])^2))
TEA_HDI_Low_RMSE_Test

EEA_HDI_Low_Residuals_Test   <- EEA_HDI_Low_Test-Predicted_EEA_HDI_Low$reweighted.response[,1]
EEA_HDI_Low_RMSE_Test    <- sqrt(mean((EEA_HDI_Low_Test-
Predicted_EEA_HDI_Low$reweighted.response[,1])^2))
EEA_HDI_Low_RMSE_Test
```