



HUMBOLDT UNIVERSITY

STATISTICAL PROGRAMMING LANGUAGES (WS 17/18)

The driving factors behind food prices in developing countries

Authors:

Benjamin Jaidi
Stefan Ramakrishnan
Raiber Alkurdi

Professor:

Alla Petukhina

March 31, 2018

Contents

1	Introduction	2
1.1	Related Work	2
2	Methodology	3
2.1	Workflow	3
2.2	Dataset Descriptions	4
2.3	Function Descriptions	5
3	Implementation	9
3.1	Data Preparation	9
3.2	Data overview	11
3.3	Variable Selection	12
3.3.1	VIF based method	12
3.3.2	Lasso based method	14
3.3.3	Random Forest based method	16
3.4	Result Exploration	17
4	Content related analysis	18
4.1	Datasets	18
4.2	Results	23
4.2.1	Correlation / VIF based method	23
4.2.2	Lasso based method	32
4.2.3	Random Forest based method	35
4.2.4	Summary	38
5	Conclusion	39

1 Introduction

The issue of food security is important in our times of more extreme weather occurrences, political instability and economic uncertainty. According to the Food and Agriculture Organization of the United Nations (FAO), the demand for food is growing while production to cover this demand is being surpassed. For example the global production of grain will have reached 2.1 billion tons by 2030 whereas the demand for grain will have increased up to 2.7 Billion tons. (FAO, 2016). Although population growth will most likely be slower in the future, even now 2 billion people in developing countries spent up to 70% of their disposable income on food (Erokhin, 2017). This problem is further exacerbated for low incoming families. Ören recognizes income level as the most decisive variable for food security (Ören, 2013).

As pointed out by Erokhin an important factor contributing to food security are food prices, as well as economic, climatic conditions and political stability. We leaned on the (Erokhin, 2017) paper in the development of the contributing factor framework. The OECD has further identified a set of factors influencing crop prices. This paper copes with the influence on crop prices on domestic markets in selected developing countries. The goal is to find out to which extent the OECDs findings are also applicable to India, Rwanda and the Philippines. This country selection aims to cover variety of economic factors as both Rwanda and the Philippines are considered developing countries whereas India is considered a lower-middle income country. Geographical and population differences between these countries allow us to form a more comprehensive conclusion if the OECD approach is reproducible for a variety of base conditions. Apart from the raw data selection process the Factor selection is approach in a variety of ways, with importance of factors varying widely in the different frameworks. We leaned on the related literature for the creation of contributing factor framework and adjusted it on granular level to fit our needs. Erokhin groups the factors through two parameters. One the one hand he focuses on physical availability of food (domestic production and import), the second being economic access (purchasing power, food inflation, distribution etc.) This was take into consideration but we choose the overarching parameters division of supply and demand factors as our structure format. Similar to (Smith, 1990) he groups factors into supply (weather, production, policy incentives, stocks and imports) and demand factors (population growth, income growth and distribution, and export revenue). Our finished framework has the two overarching categories of supply factors - demand factors. The subcategories for the supply side are climatic factors consisting of rain and temperature data, production factors consisting of the oil price and production amount and macro economic factors consisting of the Agricultural GDP, inflation and imports. On the demand side the subcategories are demographic factors consisting of GNI, per capita caloric intake and population growth and macro economic factors consisting of export of goods. This selection of factors will be discussed in more detail in the Methodology part of our paper.

1.1 Related Work

This works mainly leaned on the “Establishing Food Security and Alternatives for International Trade in Emerging Economies” by Erokhin for the creation of our methodology as highlighted priorly in the introduction section. The OECD work regarding the rise of food prices and the common consequences was used as a benchmark. In their work they take global look at food prices development and how specific factors impacted these prices movements. The argument being that tight market conditions for essential agricultural commodities need to be understood so that national governments cant create meaningful policy answers (OECD, 2008). The OECD paper aims to aid governments in the policy making process which necessitates an understanding on which factors impact food prices. While the OECD goal is to aid the policy formation of national governments our work builds upon the general framework of the OECD’s understanding of food price impacting factors. The OECD created a global framework, whereas our work takes these global notions to test if they are applicable on a country specific level. Our work builds upon this more global framework by testing the OECD global approach on a country specific level in order to access if these general notions hold true when used within a smaller case scenarios.

2 Methodology

This section gives an overview of every step in the workflow from raw data to interpretable results as well as involved utility functions and the datasets.

2.1 Workflow

The process to achieve the desired results was structured into four main stages as depicted in **Figure 1**. Each step includes working R scripts that were used to generate the desired output files for the following steps, as well as Quantlets. These are supposed to work as runnable examples of every step conducted in the process. They are functionally independent of the working scripts and of their resources since every resource required by each quantlet is stored in the quantlet's own folder. Working scripts are executable as well except for scripts located in folder Processing_scripts, due to missing resource files that could not be uploaded because of github's file size constraints.

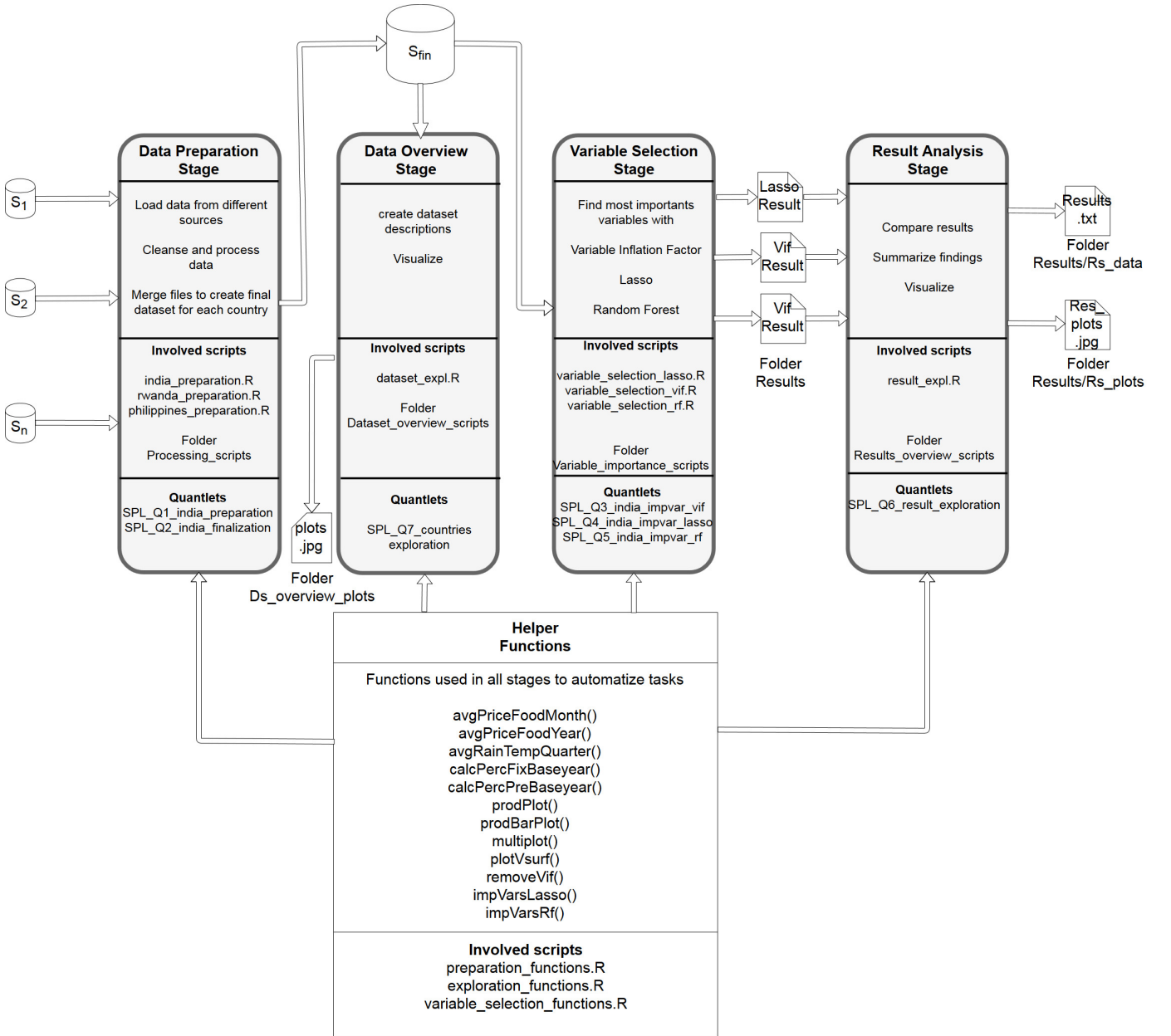


Figure 1: Workflow Structure.

In order to build a clean and valid dataset suitable for further analysis, first of all various different pieces of information needed to be taken from a range of different sources and blended together. This is done in the data preparation stage. There is a preparation script for each country we analyzed in the folder Processing_scripts in our github repository (https://github.com/jaidikam/sps_ws1718). The scripts make use of utility functions avgPriceFoodMonth(), avgPriceFoodYear() and avgRainTempQuarter() defined in the script preparation_functions in folder Helper_functions. Quantlets, SPL_Q1_india_preparation and SPL_Q2_india_finalization serve as a runnable more compact example of all the steps executed in this stage for the country india, independent from our working scripts. The subsequent data overview stage includes all the steps taken in the explorative analyzation of the datasets created in the previous stage. The script dataset_expl.R in the folder Dataset_overview_scripts holds all the code required to generate graphs such as the development of prices over the years and production rates. The results are stored in .jpeg file format in folder Dataset_overview_scripts/Plots. Utility functions used are calcPercFixBaseyear(), calcPercPreBaseyear(), prodPlot() and multiplot() in the script exploration_functions in folder Helper_functions. The Quantlet SPL_Q7_countries_exploration shows an example of how the explorative graphs were created. The next step is to find out the set of important variables for each dataset. This is achieved in the variable selection stage. For each technique, there is a script in the folder Variable_importance_scripts. Scripts make use of the functions removeVif(), impVarsLasso(), impVarsRf() defined in the script variable_selection_functions.R in the folder Helper_functions. The resulting Files including the important variables and additional information are stored in folder Results. The Quantlets SPL_Q3_india_impvar_vif, SPL_Q4_india_impvar_lasso, SPL_Q5_india_impvar_rf show a functionally independent example of how each technique was used in our process. The final result analysis stage contains all the steps related to display and analyzation of the result files produced in the previous step. Code is included in the working script result_expl.R in folder Results_overview_scripts. For demonstrating purposes there is an independent Quantlet SPL_Q6_result_exploration.

2.2 Dataset Descriptions

Dataset description: The following tables will showcase the variables used in our datasets. We have three country specific datasets (India, Rwanda and the Philippines) containing information on supply and demand related factors influencing food prices. For the sake of clarity we have split the variables in tables below showing which data is shared by the individual sets and which data is country specific. Information on the variable names, type, range, factor group as well as a description is included.

The initial table show cases information of the data shared by all three countries: India / Philippines / Rwanda.

VariableName	DataType	Range(India)	Range(Philippines)	Range(Rwanda)	FactorGroup	Description
Year	int	2001-2015	1998-2015	1991-2015	x	Starting and end year of data collection
prod_name	char	x	x	x	x	name of the selected product
prod_price	num	8.392 - 37.02	24.70-452.10	37.9-1095.9	supply/production	?
tas_q1	num	19.86-21.64	22.826-52.1206	18.54-21.13	supply/climatic	average temperature in celsius for quartile 1
tas_q2	num	28.59-30.25	11.5366-21.1364	19.03-20.90	supply/climatic	average temperature in celsius for quartile 2
tas_q3	num	26.67-27.36	15.3575-24.5690	19.00-21.26	supply/climatic	average temperature in celsius for quartile 3
tas_q4	num	21.07-22.34	15.9073-48.1226	19.17-21.42	supply/climatic	average temperature in celsius for quartile 4
pr_q1	num	7.039-23.189	13.4319-61.9415	72.86-198.95	supply/climatic	average rain fall in mm for quartile 1
pr_q2	num	54.93-109.00	16.2071-62.5248	45.45-126.96	supply/climatic	average rain fall in mm for quartile 2
pr_q3	num	161.3-246.1	19.1587-37.5307	19.39-135.49	supply/climatic	average rain fall in mm for quartile 3
pr_q4	num	21.64-49.27	77.865-49.4250	85.2-186.9	supply/climatic	average rain fall in mm for quartile 4
avg_p_barrel	num	23.12-109.45	12.28-109.45	12.28-109.45	supply/production	annual oil price (U.S. dollars per barrel)
population	num	1.071e+09-1.309e+09	74694-101716	5928-11630	demand/demographic	annual population level
prod_amount	int	41555-362333	4106698-28376518	2300-3547200	supply/production	produced amount in (1k tons)
gni_pc	num	778.4-1737.8	1784-3163	203.8-696.8	demand/demographic	per capita income(GNI per capita in constant 2010 US\$)
cp_inflation	num	3.685-11.992	1.434-9.235	-2.406-56.000	supply/macroeconomic	inflation consumer prices (annual %)
agri_gdp	num	2.092e+11-3.281e+11	1.636e+10-2.670e+10	4.256e+08-2.098e+09	supply/macroeconomic	GDP in agriculture value added (constant 2010 US\$)
daily_caloric_supply	num	2256-2459	2292-2595	1723-2270	demand/demographic	per capita calorie intake (kcal)
imp_cer	int	25911-992977	27159-117853	7625-142335	supply/macroeconomic	annual imports in thousands of dollars

Table 1: Data Shared By All Countries

The following table will showcase information of the data shared by India and Rwanda

VariationName	DataType	Range(India)	Range(Rwanda)	FactorGroup	Description
imp_veg	num	1140825-6317271	1954-23112	supply/macroeconomic	annual vegetable imports in thousands of dollars
exp_cer	num	3.210e+07-2.248e+09	0.51-54746.39	demand/macroeconomic	annual cereal exports in thousands of dollars
exp_veg	num	851222-3559119	43.56-8891.15	demand/macroeconomic	exported vegetables in US\$

Table 2: Data Shared By India and Rwanda

The next table will showcase information of the data shared by the Philippines and Rwanda.

VariableName	DataType	Range(Rwanda)	Range(Philippines)	FactorGroup	Description
GDP	num	7.536e+08-8.261e+09	7.221e+10-2.928e+11	supply/macroeconomic	
exchange_rate	num	125.2-721.0	39.09-56.04	supply/macroeconomic	exchange rate in relation to 2010 US\$
population_unit	num	1000	1000	demand/demographic	Unit for population size (1000 Persons)
flag	chr	x	x	x	Country name
flag.description	chr	x	x	x	Description of source for country name

Table 3: Data Shared by Philippines and Rwanda

The following table will showcase information of the data specific to India

VariableName	DataType	Range(India)	FactorGroup	Description
country	Factor	x	x	Country Name
imp_sug	num	26034-1419642	supply/macroeconomic	annual imports in thousands of dollars
exp_sug	num	91273-2247911	demand/macroeconomic	annual exports in thousands of dollars

Table 4: Data Specific to India

The final table will showcase information specific to the Philippines

VariableName	DataType	Range(Philippines)	FactorGroup	Description
exp_agri	num	36.30-142.32	demand/macroeconomic	

Table 5: Data Specific to Philippines

2.3 Function Descriptions

- avgPriceFoodMonth
 - description:
 - * calculates the average price per month for each product for a given dataframe.
 - * Adds column avg_price_prod_month.
 - input parameters:
 - * ds: food price data as a dataframe
 - * cm_name: the name of the column holding the product name as a String
 - * mp_price : the name of the column holding the product price as a String
 - * mp_year: the name of the column holding the year as a String
 - * mp_month: the name of the column holding the month as a String
 - output parameters
 - * ds: the input dataframe including an additional column avg_price_prod_month
 - typical issues:
 - * Wrong column name specified: Error
 - * Column name not passed as String: Error
- avgPriceFoodYear
 - description:

- * calculates the average price per year for each product for a given dataframe
 - * Adds column avg_price_prod_year.
- input parameters:
 - * Ds: food price data as a dataframe
 - * cm_name: the name of the column holding the product name as a String
 - * mp_year: the name of the column holding the year as a String
 - * avg_price_prod_month: the name of the column holding the average food price per month as a String
- output parameters
 - * ds: the input dataframe including an additional column avg_price_prod_year
- typical issues:
 - * Wrong column name specified: Error
 - * Column name not passed as String: Error
- avgRainTempQuarter
 - description:
 - * Calculates the average temperature and average amount of rain per quarter year for a given dataframe
 - input parameters:
 - * ds: rain and temperature data for every month in each year as a dataframe
 - * month: the name of the column holding the month as a String
 - * mp_year: the name of the column holding the year as a String
 - * pr: the name of the column holding the amount of rain per month as a string
 - * tas: the name of the column holding the average temperature per month as a string
 - output parameters
 - * ds: the input dataframe including an additional
 - * columns: tas_q1, tas_q2, tas_q3, tas_q4, pr_q1, pr_q2, pr_q3, pr_q4
 - typical issues:
 - * Wrong column name specified: Error
 - * Column name not passed as String: Error
- plotVsurf
 - description:
 - * Plots VSURF objects for thresholding and interpretation step
 - input parameters:
 - * iVsurfOb: A VSURF object
 - * iStep: the step for which results are to be plotted as a string
 - * iCountry: the country for which results are to be plotted as a string
 - typical issues:
 - * Other object than VSRUF object passed to function
 - * Other string than "thres" or "interp" passed as value for iStep
- removeVif
 - description:
 - * Removes multicorrelated numeric variables from a given dataframe based on variance inflation factor

- input parameters:
 - * `explan_vars`: the numeric variables as a dataframe
 - * `cutoffval`: the maximum allowed vif for remaining variables as a number
- output parameters
 - * `tempresults`: the remaining variable names and their corresponding vif as a dataframe
- typical issues:
 - * non numeric variables in input dataframe
- `impVarsLasso`
 - description:
 - * Identifies important variables for a given dataframe based on the lasso method
 - input parameters:
 - * `ds`: the variables as a dataframe
 - * `targ`: the name of the target variable column in `ds` as a String
 - output parameters
 - * `resultset`: the lasso model, fitted model and the label for display in a graph as a vectorlist
 - typical issues:
 - * Column name not passed as String: Error
- `impVarsRf`
 - description:
 - * Identifies important variables for a given dataframe based on MSE error minimization in OOB samples of random forest
 - input parameters:
 - * `ds`: the variables as a dataframe
 - * `targ`: the name of the target variable column in `ds` as a String
 - output parameters
 - * `resultset`: names of important variables and mean OOB rate as a vectorlist
 - typical issues:
 - * Column name not passed as String: Error
- `calcPercFixBaseyear`
 - description:
 - * calculate the percentage of the change in a column's value based on a fixed year
 - input parameters:
 - * `ds`: the variables as a dataframe
 - * `areacol`: name of the column holding the areas as a String
 - * `areaname`: name of selected area
 - * `yearcol`: name of the column holding the years as a String
 - * `baseyear`: selected base year
 - * `valuecol`: name of the target column holding the values as numbers for the calculation
 - * `perccol`: name of the new generated column holding the results of the calculations
 - output parameters
 - * `ds`: the input dataframe including an newly generated column

- typical issues:
 - * the newly generated column is not identified at the first assignment: Error
- calcPercPreBaseyear
 - description:
 - * calculate the percentage of changes in a column's values in a predefined year, where the base is for every change is the value from the previous year.
 - input parameters:
 - * ds: the variables as a dataframe
 - * areacol: name of the column holding the areas as a String
 - * areaname: name of selected area
 - * yearcol: name of the column holding the years as a String
 - * valuecol: name of the target column holding the values as numbers for the calculation
 - output parameters
 - * ds: the input dataframe including an newly generated column
 - typical issues:
 - * At the loop the previous has a value but the later year not: Divide by Zero Error
- prodPlot
 - description:
 - * plot production data with specific area and items
 - input parameters:
 - * ds: the variables as a dataframe
 - * area: name of the selected area as a String
 - * items: name of the selected Items as a sequence of Strings
 - output parameters
 - * p: a plot showing the percentage change in production quantities, based on a specific area and items
- prodBarPlot
 - description:
 - * plot production data with specific area and items
 - input parameters:
 - * ds: the variables as a dataframe
 - * dnmae: name of the selected area as a String
 - output parameters
 - * p: a plot showing the percentage change in product's price, based on a specific area

3 Implementation

This section gives detailed information about the actual implementation of every step featured in **Section 2** including r code and theoretical background of every variable technique that was applied.

3.1 Data Preparation

In order to get comparable results, it was necessary to work with datasets containing comparable features. After specifying which variables were supposed to be included in the final datasets, those information needed to be put together from a range of heterogenous sources. The source dataset containing food prices, for an instance, included the monthly price of each crop of interest for each marketplace in a certain country. That made it necessary to calculate the average price per crop per month for the whole country which is done by the following function:

```
1 #Define the function for food price per month across all markets
2 avgPriceFoodMonth = function(ds,cm_name,mp_price,mp_year,mp_month){
3   dsfoods = unique(ds[[cm_name]])
4   ds$avg_price_prod_month = 0
5   for (k in min(ds[[mp_year]]):max(ds[[mp_year]])){
6     print(paste('year is ',k))
7     for (j in 1:NROW(dsfoods)){
8       a <- ds[ds[[mp_year]] == k & ds[[cm_name]] == dsfoods[j],]
9       print(paste('food is ',dsfoods[j]))
10      for (i in 1:12){
11        b <- a[a[[mp_month]] == i,]
12        if (NROW(ds[ds[[mp_year]] == k & ds[[cm_name]] == dsfoods[j] & ds[[mp_month]]
13          == i,$avg_price_prod_month) >0) {
14          ds[ds[[mp_year]] == k & ds[[cm_name]] == dsfoods[j] & ds[[mp_month]] == i,]
15            $avg_price_prod_month = sum(b[[mp_price]])/ NROW(b)
16          print(paste('month is ',i))
17        }
18      }
19    }
20  }
21  return(ds)
22 }
23 india = avgPriceFoodMonth(india,"cm_name","price","year","month")
```

Listing 1: [SPL_Q1_india_preparation](#)

Function avgPriceFoodMonth takes the dataframe containing price information and allows the user to explicitly pass the names of the columns holding relevant information as strings. First, possible duplicates in the input dataset are removed and the name of every crop is stored in a variable dsfoods. Then a column avg_price_prod_month to hold the result is added to the dataset with value 0 . The first most outer loop iterates through the year. The second loop goes through every crop. The most inner loop goes through every month for every possible year/crop pair. If rows exist for a certain month/year/crop combination, the sum of prices for that combination are divided by the number of occurrences resulting in the desired average. The if – condition is necessary to avoid the possibility to divide by 0. Finally, the dataframe plus the newly created column is returned. Later on in the process we decided to look at crop prices per year, so we created another similar function to calculate the average price per food per year:

```
1 avgPriceFoodYear = function(ds,cm_name,mp_year,avg_price_prod_month){
2   dsfoods = unique(ds[[cm_name]])
3   ds$avg_price_prod_year = 0
4   for (x in min(ds[[mp_year]]):max(ds[[mp_year]])){
5     print(paste('year is ',x))
6     for(y in 1:NROW(dsfoods)){
7       print(paste('food is ',dsfoods[y]))
8       if(NROW(ds[ds[[mp_year]] == x & ds[[cm_name]] == dsfoods[y],]$avg_price_prod_
9         year) > 0){
```

```

9      ds[ds[[mp_year]] == x & ds[[cm_name]] == dsfoods[y],]$avg_price_prod_year =
      sum(ds[ds[[mp_year]] == x & ds[[cm_name]] == dsfoods[y],][[avg_price_prod
      _month]]) / NROW(ds[ds[[mp_year]] == x & ds[[cm_name]] == dsfoods[y],)[[
      avg_price_prod_month]])
10    }
11  }
12 }
13 return(ds)
14 }
15 india = avgPriceFoodYear(india,"cm_name","year","avg_price_prod_month")

```

Listing 2: SPL_Q2_india_preparation

The function takes a dataframe including crop prices on a monthly level and allows the user to specify the columns holding crop name, year and price per month. We pass the dataset that was created by the previous function and the column for the average price per year is added. The outer loop iterates through the years, the inner loop goes through the list of unique crop names. For every existing year/crop combination the average is price calculated.

Another dataset that required processing was the climate set, containing the amount of rain and the temperature for every month in a year, each in a column of its own. Instead of one row per month, we needed the average amount of rain / temperature for each quarter of the year to be a row identified by the year. A function was written to achieve that:

```

1 avgRainTempQuarter = function(ds,month,mp_year,pr,tas){
2   if (is.na(ds[[pr]]) || is.na(ds[[tas]])) {
3     message(paste("No missing values allowed!"))
4   } else {
5     ds$tas_q1 = 0
6     ds$tas_q2 = 0
7     ds$tas_q3 = 0
8     ds$tas_q4 = 0
9     ds$pr_q1 = 0
10    ds$pr_q2 = 0
11    ds$pr_q3 = 0
12    ds$pr_q4 = 0
13
14    for(z in min(ds[[mp_year]]):max(ds[[mp_year]])){
15
16      ds[ds[[mp_year]] == z ,]$pr_q1 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in
      % c("1","2","3"),)[[pr]])/3
17      ds[ds[[mp_year]] == z ,]$pr_q2 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in
      % c("4","5","6"),)[[pr]])/3
18      ds[ds[[mp_year]] == z ,]$pr_q3 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in
      % c("7","8","9"),)[[pr]])/3
19      ds[ds[[mp_year]] == z ,]$pr_q4 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in
      % c("10","11","12"),)[[pr]])/3
20      ds[ds[[mp_year]] == z ,]$tas_q1 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in
      % c("1","2","3"),)[[tas]])/3
21      ds[ds[[mp_year]] == z ,]$tas_q2 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in
      % c("4","5","6"),)[[tas]])/3
22      ds[ds[[mp_year]] == z ,]$tas_q3 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in
      % c("7","8","9"),)[[tas]])/3
23      ds[ds[[mp_year]] == z ,]$tas_q4 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in
      % c("10","11","12"),)[[tas]])/3
24    }
25    return(ds)
26  }
27 }
28 raintemp = avgRainTempQuarter(raintemp,"month","year","pr","tas")
29 #load the india dataset and the remaining variables
30 india_wip = readRDS("...\\Qfolder1\\Q1_india_wip.rds")
31 rest = readRDS("...\\Qfolder2\\Q2_india_rest.rds")
32 #merge india with wheater data
33 india_wip = merge(india_wip,unique(raintemp[c("tas_q1","tas_q2","tas_q3","tas_q4","pr

```

```

34 |     _q1", "pr_q2", "pr_q3", "pr_q4", "year"]]), by=c("year"))
35 | #join the datasets
india_fin = merge(india_wip, rest, by=c("prod_price")) #prod_price is unique,
               therefore it can be used as a key for the merge

```

Listing 3: [SPL_Q2_india_finalization](#)

The function takes a the climate dataset and allows the user to pass strings to identify the columns holding the month, year, rain and temperature respectively. Given that the dataset has no missing values in the rain and the temperature column, additional columns are added for every quarter of the year and the filled with the sum of each quarter divided by the number of month a quarter year has. The dataset with the added columns is then returned. Afterwards, the datasets are merged to create the final dataset used in the next step.

3.2 Data overview

In this section we will explain the process and the code which we used to produce four of our graphs, which provide us with some important statistical comparison. All the produced graphs are mmen- tioned in **Section 4.1** The resulted graphs were:

- Product Price Change
- Population Change
- Price Index
- Consumer Price Development

We have collected our databases from different resources, and for the values to be comparable between countries an world wide, we had to develop some functions to calculate the percentage of the change

```

1 | #calculating the percentage of the change in a column's value on a fixed base year
2 | calcPercFixBaseyear = function(ds, areacol, areaname, yearcol, baseyear, valuecol,
   |   perccol){
3 |   base = ds[ds[[yearcol]] == baseyear & ds[[areacol]] %in% areaname, ][[valuecol]]
4 |   if(is.null(ds[[perccol]])){
5 |     ds[[perccol]] = 0
6 |   }
7 |   #
8 |   for(i in baseyear: max(ds[[yearcol]])){
9 |     later = ds[ds[[yearcol]]==i & ds[[areacol]] %in% areaname, ][[valuecol]]
10 |     sub = later - base
11 |     ds[ds[[yearcol]] == i & ds[[areacol]] %in% areaname, ][[perccol]] = (sub / later)
   |       * 100
12 |   }
13 |   return(ds)
14 | }

```

Listing 4: [SPL_Q7_countries_exploration.R](#)

Function calcPercFixBaseyear calculates the precentage change in a value basd on a specifed base year during a timeframe, where the percentage will always represent the difference between the cvalue column and the value stored in the base year, beside the base year the function takes also tar- geted dataframe, area column, area name, year column, value column and the name of the produced new column. We firstly store the given year in the base variable then we check if the goal new col- umn have been already identified or not. After that we go through a loop in value column depending on the targeted year column and store the resulted percentage change in the new column and at the the end returning the new dataframe comntain the newly calculated column for the specified area during the the specifed time duration.

```

1 # calculate the percentage of changes in a colname value in a predefined year, where
  the base is for every change is the value from the previous year.
2 calcPercPreBaseyear = function(ds, areacol, areaname, yearcol, valuecol){
3   for(i in unique(ds[[yearcol]])){
4     base = ds[ds[[yearcol]] == i & ds[[areacol]] %in% areaname, ][[valuecol]]
5     later = ds[ds[[yearcol]] == i+1 & ds[[areacol]] %in% areaname, ][[valuecol]]
6     #if(length(later) == 0L) break
7     if(i == 2015L) break
8     sub = later - base
9     if(length(sub) == 0L)next
10    ds[ds[[yearcol]] == i+1 & ds[[areacol]] %in% areaname , paste(valuecol, "Percent"
11      , sep = "_")]= (sub / later) * 100
12  }
13  ds[ds[[yearcol]] == min(ds[[yearcol]]) & ds[[areacol]] %in% areaname, paste(
14    valuecol, "Percent", sep = "_")]= 0
15  return(ds)
16 }

```

Listing 5: SPL_Q7_countries_exploration.R

Function calcPercFixBaseyear also calculate the precentage change, but this time the base year is changing and is not fixed, it takes the value of the pervious year at each calculation for a value in a specific year

```

1 # To plot production data with specific area and itmes
2 prodPlot = function(ds, area, items){
3   p = ggplot(data=ds[ds$Area == area & ds$Item %in% items,], aes(x=Year, y=Percentage
4     , colour=Item)) +
5     geom_line() +
6     geom_point()+
7     ylim(-30, 75)+
8     ggtitle(label=area)+
9     ylab(label="Percentage Production Change") +
10    xlab("Year")
11  return(p)
12 }

```

Listing 6: SPL_Q7_countries_exploration.R

We use prodPlot function to produce the six graphs in **Figure 5. Consumer Price development**, where it show us the change of the products prices in the studied countries in compare the world wide change. The function takes the dataframe along side the area name and targeted items, then it produce the plot for the defined parameters.

3.3 Variable Selection

To find out which variables have the biggest impact on food prices, we apply three techniques which were implemented as follows:

3.3.1 VIF based method

Before fitting a linear model, first of all it is necessary to check for correlation among explanatory variables, since high correlation means that variables are not independent from one another As a way to measure the degree of dependence of variable pairs, Pearson's correlation coefficient was used. Intuitively, explanatory variables should only be correlated to the target variable, not to each other. Otherwise the model will have high standard deviation and therefore high variance and possibly low predictive power. Also significant variables may appear non-significant. However, pair-wise analysis for correlation may not reveal multicollinearity, i.e. a situation where a predictor can be explained through the other predictors. Multicollinearity is excessive correlation among several explanatory variables. A commonly used way to detect that is a metric called variance inflation factor. It helps

to quantify the amount of variance that each predictor adds to the model. To calculate the VIF for a variable b_k , Given a linear model :

$$y_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_k x_{ik} + \dots + \beta_{p-1} x_{i,p-1} + \epsilon_i$$

By regressing b_k with only one of the other predictors at a time we calculate the variances for b_k . We keep the smallest variance, obtained by:

$$Var(b_k)_{min} = \frac{\delta^2}{\sum_{i=1}^n (x_{ik} - \bar{x}_k)^2}$$

In order to find out how much b_k increases the overall variance of the model we get the ratio of $Var(b_k)_{min}$ and $Var(b_k)$, the variance of b_k when all remaining variables are regressed on b_k at the same time:

$$\frac{Var(b_k)}{Var(b_k)_{min}} = \frac{\left(\frac{\delta^2}{\sum_{i=1}^n (x_{ik} - \bar{x}_k)^2} \times \frac{1}{1 - R_k^2} \right)}{\left(\frac{\delta^2}{\sum_{i=1}^n (x_{ik} - \bar{x}_k)^2} \right)} = \frac{1}{1 - R_k^2}$$

R_k^2 is the R^2 value when the k^{th} is regressed on all remaining explanatory variables
It follows :

$$VIF_k = \frac{1}{1 - R_k^2}$$

The correlation- and VIF based method was implemented as follows: After choosing the explanatory variables and standardizing them we acquire the correlation coefficient for each pair:

```

1 #initial variable selection and normalization
2 colselection_in = c("prod_price", "pr_q1", "pr_q2", "pr_q3", "pr_q4", "tas_q1", "tas_q2", "
   tas_q3", "tas_q4",
3                       "prod_amount", "daily_caloric_supply", "exp_sug", "exp_veg", "exp_cer
   ", "imp_sug", "imp_veg", "imp_cer",
4                       "agri_gdp", "gni_pc", "cp_inflation", "avg_p_barrel", "population")
5 target_in = c("prod_price")
6 normalized_in = as.data.frame(scale(india[colselection_in]))
7 feats_in = normalized_in[, !(colnames(normalized_in) %in% target_in)]
8 #Variable selection and modeling
9 #Obtaining pair-wise correlations
10 insign_in = cor(feats_in, method = "pearson", use = "complete.obs")

```

Listing 7: `SPL_Q3_india_impvar_vif`

After we calculated the correlations for each pair, every variable being part of a pair with a correlation coefficient above 0.70 is removed and a linear model is built.

```

1 # Discovering highly correlated explanatory variables
2 hicorvars_in = findCorrelation(cor(feats_in), cutoff = 0.70)
3 expvarsnohc_in = paste(colnames(feats_in[, -hicorvars_in]), collapse = "+")
4 formulanohc_in = paste(target_in, "~", expvarsnohc_in, collapse = "+")
5 mod_varnohc_in = lm(formulanohc_in, data = normalized_in)
6 #For comparison we also apply the VIF-based method to tackle multicollinearity:
7 #function for VIF based stepwise removal of multicorrelated variables
8 removeVif = function(explan_vars, cutoffval=10){

```

```

9   tempresults = as.data.frame(matrix(ncol = 2, nrow = 0))
10  colnames(tempresults) = c("variable", "vif")
11  #initially calculate VIF for each explanatory variable
12  for (i in 1:NROW(colnames(explan_vars)) ){
13    temptarget = colnames(explan_vars)[i]
14    tempexpvars = paste(colnames(explan_vars[,!(colnames(explan_vars) %in% temptarget
15    )]),collapse = "+")
16    tempformula = paste(temptarget,"~", tempexpvars, collapse = " ")
17    tempresults[i,1] = temptarget
18    tempresults[i,2] = VIF(lm( tempformula,data = explan_vars))
19  }
20  print(tempresults[order(tempresults$vif),])
21  #remove variable with highest VIF, calculate new VIF for remaining variables until
22  all VIF are below cutoff value
23  while(max(tempresults$vif) >= cutoffval){
24    tempresults = tempresults[!tempresults$vif == max(tempresults$vif),]
25    tempremvars = tempresults$variable
26    for(j in 1: NROW(tempremvars)){
27      temptarget = tempremvars[j]
28      tempexpvars = paste(tempremvars[!tempremvars %in% temptarget],collapse = "+")
29      tempformula = paste(temptarget,"~", tempexpvars, collapse = " ")
30      tempresults[j,1] = temptarget
31      tempresults[j,2] = VIF(lm( tempformula,data = explan_vars))
32    }
33    print("Remaining variables:")
34    print(tempresults[order(tempresults$vif),])
35    cat("\n")
36  }
37  return(tempresults)
38 }
39 # for highly correlated variables
40 varslovifhc_in = removeVif(feats_in[,hicorvars_in],8)
41 # for lower correlated variables
42 varslovifnohc_in = removeVif(feats_in[,~hicorvars_in],8)
43 #Model without multicollinearity
44 expvars_lovif_in = paste(paste(varslovifhc_in$variable,collapse = "+"),"+",paste(
45   varslovifnohc_in$variable,collapse = "+"),collapse = "+")
46 formula_lovif_in = paste(target_in,"~",expvars_lovif_in,collapse = "+")
47 mod_lovif_in = lm(formula_lovif_in,data = normalized_in)

```

Listing 8: SPL_Q3_india_impvar_vif

The function takes a dataframe containing explanatory variables and a cutoff value that specifies the highest admissible VIF until the function stops removing variables. The r package “fmsb” is required. The function’s algorithm in pseudocode:

insert pseudocode from presentation

Finally, the function returns a dataframe containing the remaining variables and their VIF values

The function is applied to all the highly pair-wise correlated variables identified in line xx as well as to the remaining variables in a separate call. The linear model is then constructed from the remaining predictors and saved as a file.

3.3.2 Lasso based method

Lasso (Least Absolute Shrinkage and Selection Operator) is a regression based method for regularization and feature selection .A constraint is put on the absolute sum of model parameters and a penalty is applied to the regression coefficients. The strength of the penalty is defined by the parameter lambda. The bigger lambda gets, the sooner model parameters exceed the threshold value and get dropped from the model. Intuitively, a smaller lambda causes more parameters to stay in the model. Lambda = 0 means there is no penalty at all and the regression becomes an ordinary least square regression Formulation by(Bühlmann & Van de Geer 2010); // Statistics for High-Dimensional Data: Methods, Theory and Applications. Springer 2011 in Literaturverzeichnis und referenzieren

Find a solution to the optimization problem minimize

$$\left(\frac{\|Y - X\beta\|_2^2}{n} \right) \text{ subject to } \sum_{j=1}^k \|\beta\|_1 < t$$

t is the upper bound for the sum of the coefficients. Which is equivalent to the parameter estimation

$$\arg \min_{\beta} \left(\frac{\|Y - X\beta\|_2^2}{n} + \lambda \|\beta\|_1 \right)$$

Where

$$\frac{\|Y - X\beta\|_2^2}{n} = \sum_{i=0}^n (Y_i - (X\beta)_i)^2, \|\beta\|_1 = \sum_{j=1}^k |\beta_j| \text{ and } \lambda \geq 0$$

If λ gets 0, t becomes infinite and vice versa. In other words, we find the set of coefficients from the model having the least mean squared error for a sequence of descending lambdas.

For application of the lasso method, a function was written: Our implementation of the lasso technique requires the R packages "glmnet" and "plyr".

```

1  #function for LASSO method
2  impVarsLasso = function(ds,targ){
3    #1. initial variable selection and normalization
4    val = ds[[targ]]
5    x = model.matrix(ds[[targ]]~.-1 , ds[!colnames(ds) %in% targ])
6    #2. Applying the Lasso technique
7    lasso = glmnet(x = x, y = val, standardize = TRUE, alpha = 1)
8    fit = cv.glmnet(x = x, y = val, standardize = TRUE, type.measure = "mse", alpha=1,
9                  nfolds=3)
10   #3. Results
11   #with lambda.min
12   lambda_min = which(fit$lambda == fit$lambda.min)
13   #selecting coefficients of variables at lambda where mse is minimal
14   tempmincoefs = as.data.frame(fit$glmnet.fit$beta[, which(fit$lambda ==
15   fit$lambda.min))])
16   mincoefs = data.frame(matrix(ncol = 2, nrow = (NROW(tempmincoefs))
17   ))
18   mincoefs$variables = as.vector(as.character(labels(tempmincoefs)[[1]]))
19   mincoefs$coefs_minlambda = as.vector(tempmincoefs[[1]])
20   mincoefs$X1 = NULL
21   mincoefs$X2 = NULL
22   #get names in the decreasing order they appear in when lambda is minimal
23   names = names(coef(lasso)[,ncol(coef(lasso))[order(coef(lasso)[,ncol(
24   coef(lasso))],decreasing=TRUE)])
25   names = names[!names %in% c("(Intercept)")]
26   names = as.data.frame(names)
27   colnames(names) = "variables"
28   #add coefficient to names
29   disp_colors = join(names,mincoefs, by = "variables" )
30   disp_colors = disp_colors[!disp_colors$variables %in% c("(Intercept)"),]
31   #set colors for variables when displayed in a graph
32   disp_colors$colors = 0
33   if(NROW(disp_colors[disp_colors$coefs_minlambda >0,])>0){
34     disp_colors[disp_colors$coefs_minlambda >0,]$colors = c("green")
35   }
36   if(NROW(disp_colors[disp_colors$coefs_minlambda <0,])>0){
37     disp_colors[disp_colors$coefs_minlambda <0,]$colors = c("red")
38   }
39   #create a list to store the result

```



```

36 | resultset = vector("list",3)
37 | resultset[[1]] = lasso
38 | resultset[[2]] = fit
39 | resultset[[3]] = disp_colors
40 | return(resultset)
41 | }
42 | #Get most important variables with Lasso function
43 | india_lasso_result = impVarsLasso(india,"prod_price")

```

Listing 9: SPL_Q4_india_impvar_lasso

The function takes a dataframe as input and allows the user to specify the column holding the target variable by passing the name as a string. In line 21 a glmnet model is built, applying the aforementioned lasso method. Then a fitted model is built in the same way. The mean square error of each model for every lambda is calculated using three-fold cross validation for better generalization of the results, i.e. to avoid overfitting. In the following step the coefficients and variable names from the model with the lowest MSE are stored in a dataframe called mincoefs. In order to offer additional information about the correct color when the results are plotted, the variables names are ordered in the decreasing order they appear in when lambda is minimal, i.e. when their curves would cut the right margin of the plot (line 38). If coefficients in mincoefs are greater 0, they will be displayed in green in the graph, lower 0 in red, all other will be grey. Finally, the model, fitted model and the display information are stored in a list which is the returned

3.3.3 Random Forest based method

The third variable selection technique we applied is based on random forests. Random forests are a non-parametrical statistical method used for regression and classification problems. RF are an ensemble of decision trees built from samples of the whole data. The importance of a variable X^j is calculated as follows: Each tree t runs a prediction on the Out of bag sample OOB_t (the portion of data not included in the sample to create t). The mean square error MSE of this prediction is denoted $errOOB_t$. Now the values for X^j in OOB_t are permuted randomly to get a perturbed sample OOB_t^j . Then the error $errOOB_t^j$ of t on the perturbed sample is calculated. Now the variable importance for X^j is computed as follows:

$$VI(X^j) = \frac{1}{ntree} \sum_t \left(err\widetilde{OOB}_t^j - errOOB_t \right)$$

The function used to obtain the most important variables using RF requires the r package “VSURF”. The implementation utilizes a two-step approach to find the most important variables:

- **Threshold step** After computing the VI, each variable is ranked by VI in descending order. Variables with small importance are eliminated. The threshold is derived from the standard deviations of VI. Important variables have a higher variability so only variables with a averaged VI exceeding the threshold are kept in the model
- **Variable Selection Interpretation step:** A nested collection of RF models is built for the first k to m variables. The variables leading to the minimal OOB error are selected, resulting in m' variables being retained. **Prediction step:** The variables from the interpretation step are ordered and sequentially added to an RF model. Variables which lead to an error decrease above a certain threshold are kept.

Note: Since prediction step focuses on predictive power of the model, some variables that in fact have an influence on the target are removed since others are stronger predictors. Our focus lies on generally identifying variables having a significant influence on the target, therefore the prediction step is ignored in our approach.

```

1 #function for finding most important variables based on random forest
2 impVarsRf = function(ds,targ){
3   result_rf = VSURF(ds[[targ]] ~ ., data = ds[!colnames(ds) %in% targ], ntree = 2000,
4                     nfor.thres = 50, nmin = 1, nfor.interp = 25, nsd = 1,
5                     nfor.pred = 25, nmj = 1, parallel = FALSE, ncores = detectCores()
6                     - 1,
7                     clusterType = "PSOCK")
8   #create a list to store the result
9   resultset = vector("list",2)
10  resultset[[1]] = result_rf
11  resultset[[2]] = colnames(ds[!colnames(ds) %in% targ])
12  return(resultset)
13 }
14 #apply the function
15 india_v_imp_rf = impVarsRf(india,"prod_price")

```

Listing 10: [SPL_Q5_india_impvar_rf](#)

3.4 Result Exploration

Having obtained the most important variables, the results produced in the previous steps are loaded and processed to turn them into text files and images so that our findings are readable and accessible without further programming. The following excerpt gives an impression of how we achieved this

```

1 #loading results for random forest based variable selection
2 india_rf_result = readRDS(".\\Qfolder5\\Q5_india_rf.rds")
3 #function for plotting VSURF Objects
4 plotVsurf = function(iVsurfOb,iStep,iCountry){
5   header_prefix = "not specified"
6   if(iStep == "thres"){
7     header_prefix = "Thresholding step"
8   }
9   if(iStep == "interp"){
10    header_prefix = "Interpretation step"
11  }
12  plot(iVsurfOb,step = iStep, var.names = FALSE,
13       nvar.interp = length(iVsurfOb$varselect.thres), main = paste(header_prefix,
14                               iCountry))
15 }
16 #threshold step
17 #save variables and plot to file
18 sink(".\\Qfolder6\\Q6_rf_thres_in.txt")
19 print(india_rf_result[[2]][india_rf_result[[1]]$varselect.thres])
20 sink()
21 jpeg("../Qfolder6//Q6_india_rf_thres.jpg", width = 1000, height = 700, units = "px",
22       pointsize = 20,
23       quality = 100)
24 plotVsurf(india_rf_result[[1]],"thres","India")
25 dev.off()
26 #interpretation step
27 sink(".\\Qfolder6\\Q6_rf_interp_in.txt")
28 print(india_rf_result[[2]][india_rf_result[[1]]$varselect.interp])
29 sink()
30 jpeg("../Qfolder6//Q6_india_rf_interp.jpg", width = 1000, height = 700, units = "px",
31       pointsize = 20,
32       quality = 100)
33 plotVsurf(india_rf_result[[1]],"interp","India")
34 dev.off()

```

Listing 11: [SPL_Q6_result_exploration](#)

In this example the result objects from the random forest based variable selection are loaded. For the creation of the plot a function was written. The function takes a VSURF object and allows the

user to specify the corresponding country name and the step for which the results shall be plotted (“thres“ or “interp”). The corresponding variable names are also extracted from the result object and stored in a .txt-file. The results for the other techniques are processed in a similar fashion.

4 Content related analysis

This section shows the output created in Section 3. The focus is on content-related exploration and interpretation of our findings and on comparison with the discoveries from related papers featured in **Section 1**

4.1 Datasets

The following section will visually highlight some of our findings with the help of graphs. For the sake of comparability, we used percentage changes rather than absolute numbers.

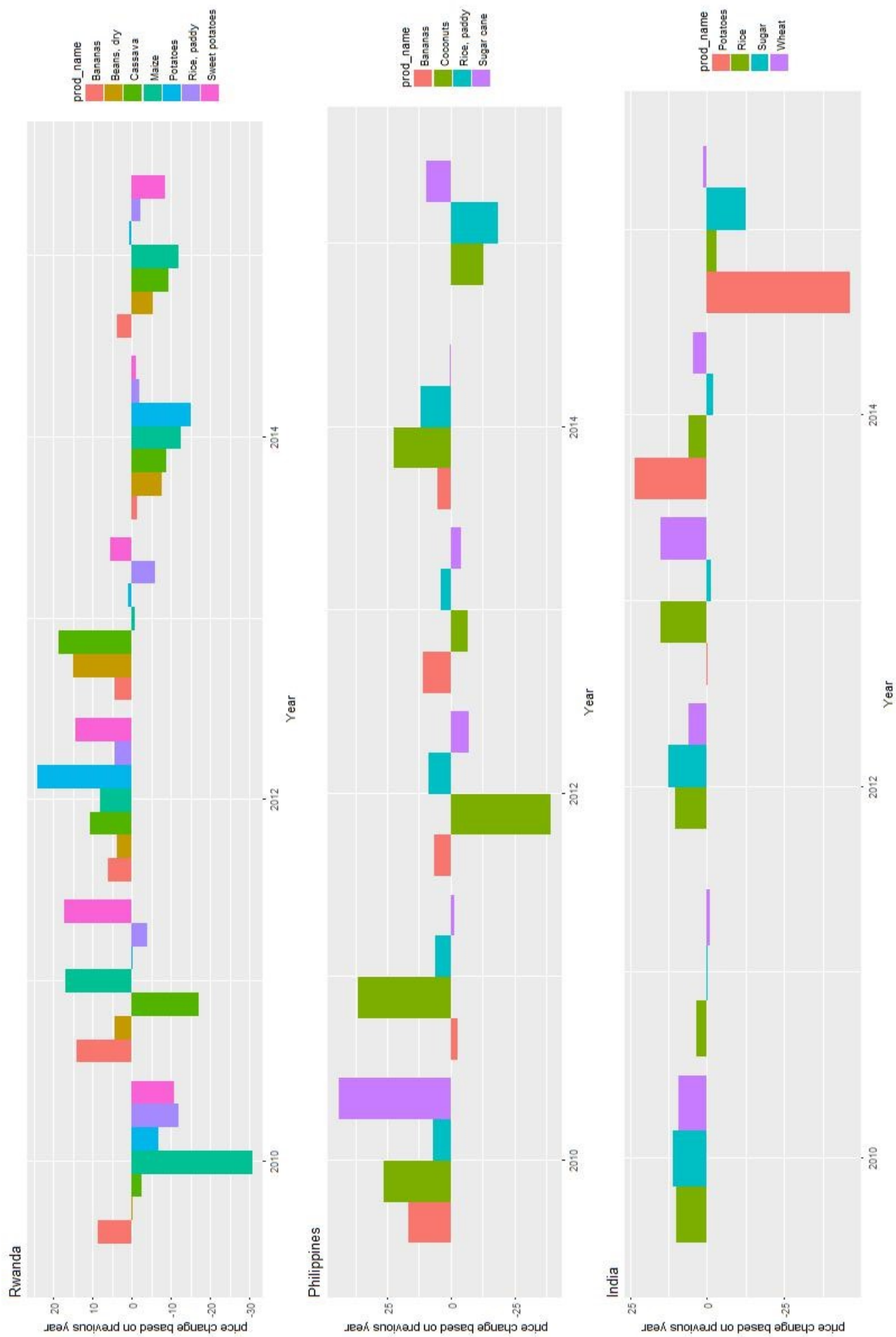


Figure 2: Products Price Change Based on The Change from Previous Year

The first graph **Figure 2** showcases the annual percentage price change of our crop selection. Our selection is based on Harmonized System Codes (HS Code 2017). We focused on the category 07-015 – "Vegetables And Certain Roots And Tubers; Edible".

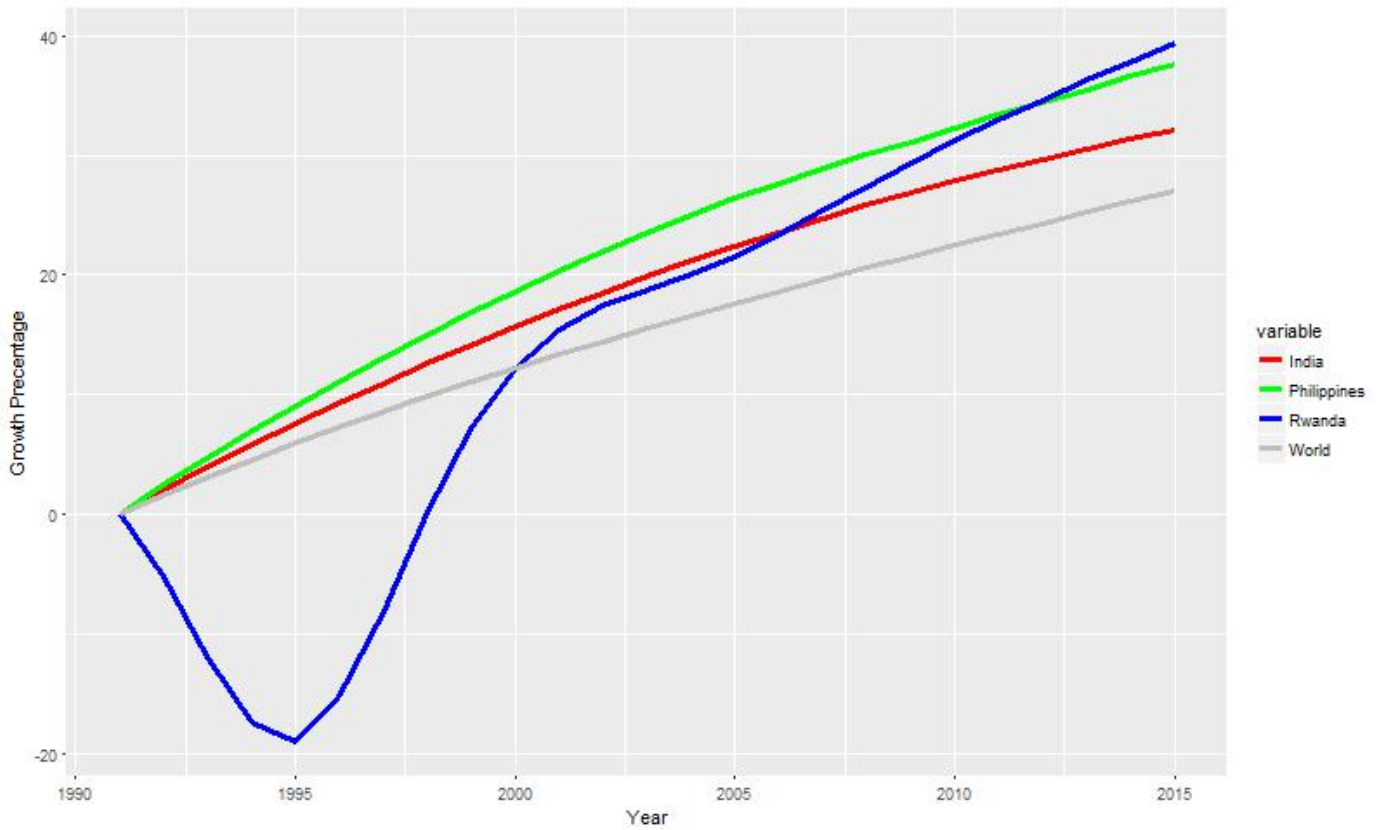


Figure 3: Population Change

The second graph **Figure 3** shows the population percentage population growth against the baseline world population percentage growth from 1990 to 2015. It is interesting to note that our country selection generally had a higher growth percentage than the global percentage trend with the notable outlier of Rwanda in the early 1990s. We assume that this sharp decline was related to the catastrophic civil war that ravaged Rwanda in 1994.

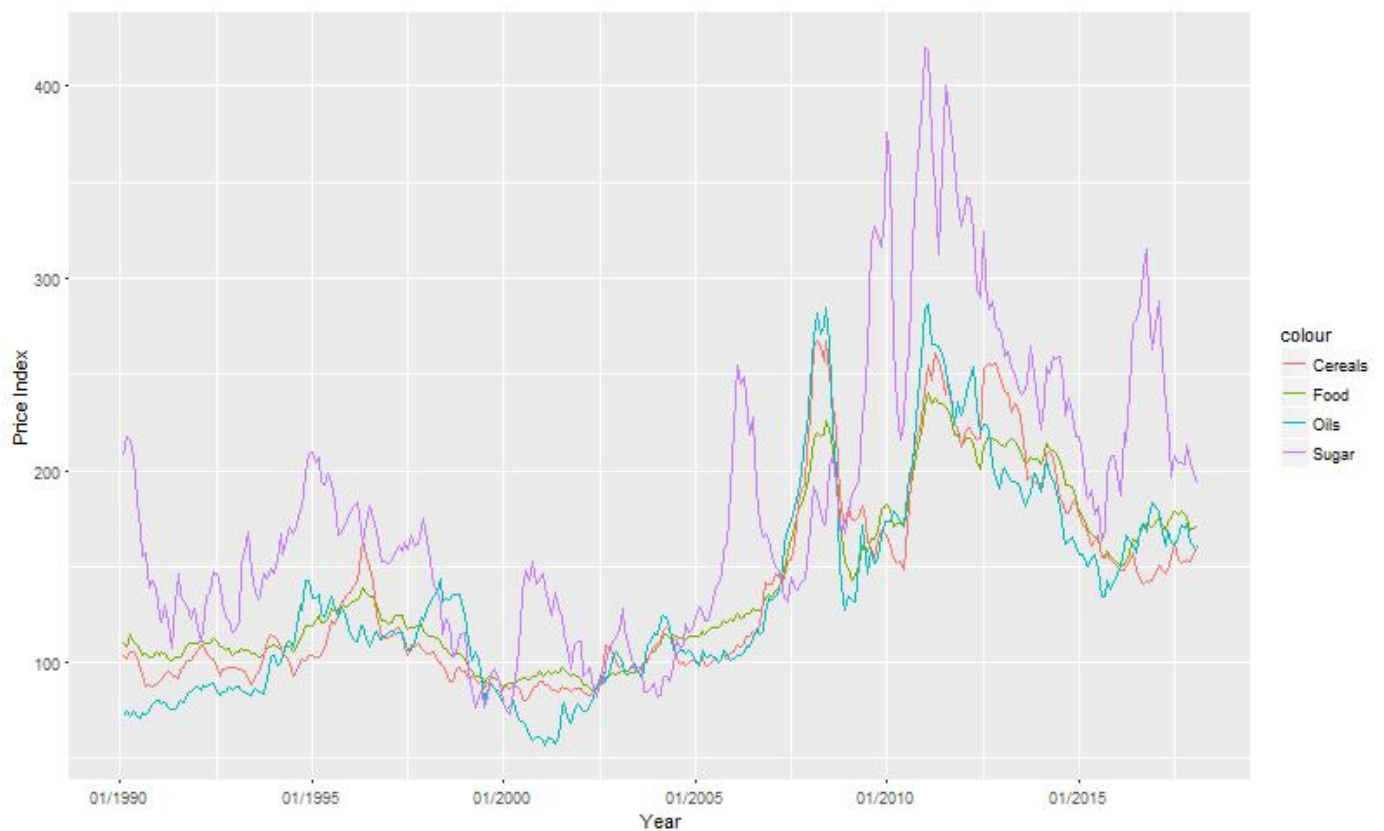


Figure 4: Global Average Food Category Price Change

The next graph **Figure 4** shows the price index percentage change of cereals, vegetable oils and sugar against the global food price index percentage change from 1990 to 2015. A general upward trend can be observed with short term price spikes. Around 2010 a general price index drop can be observed. Sugar price percentage change is higher than the more homogenous trend development of cereals and vegetable oils which seem to be in line with the global food price index trend. These observations are in line with observations presented by the OECD.

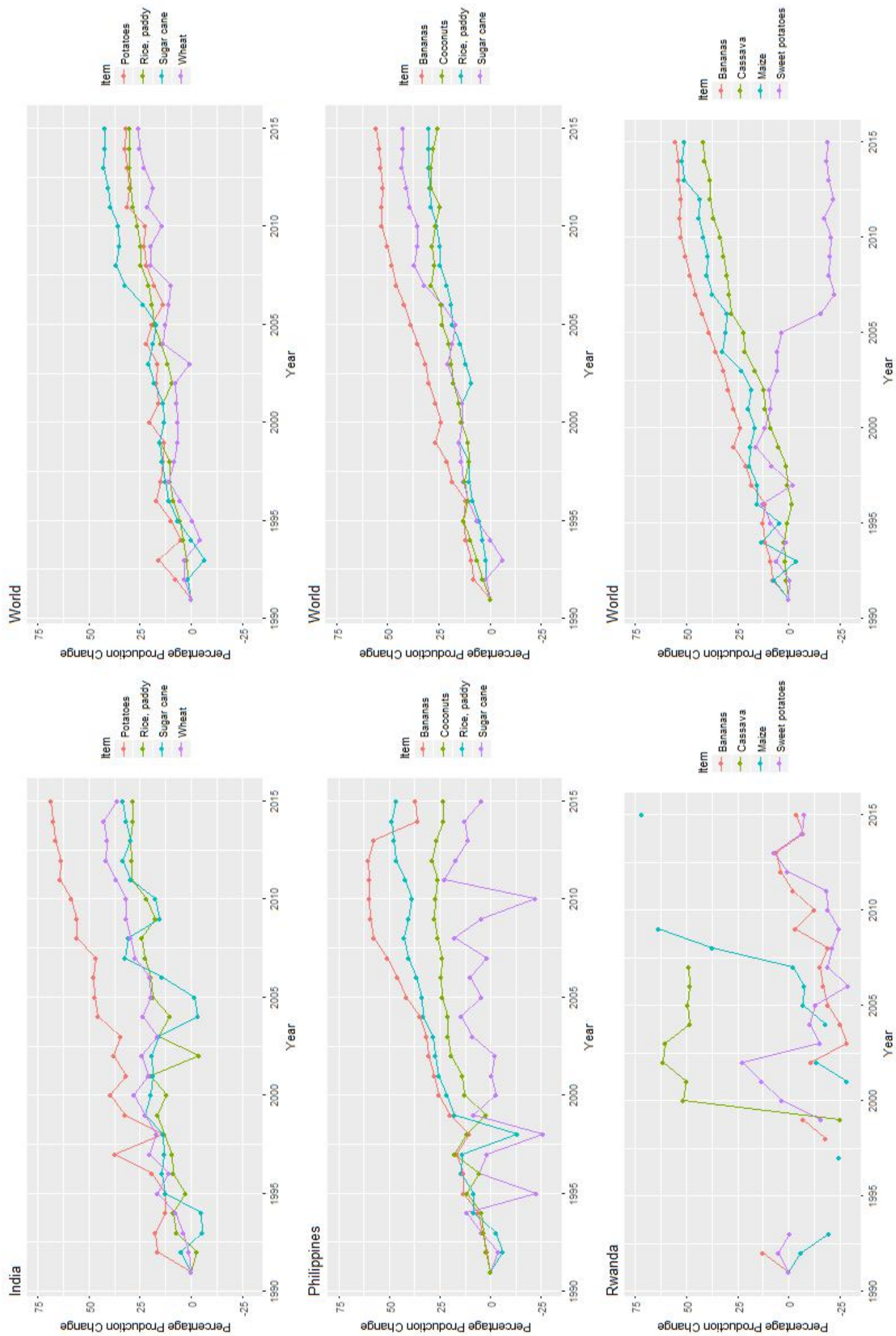


Figure 5: Consumer Price Development

The final set of graphs **Figure 5** shows the consumer price development of our selected goods on a by country level against global consumer price development for the same goods from 1990 - 2015. Starting with India one can note that the general increasing global trend of consumer prices can be observed on the country level as well. Notable exceptions are the substantially higher consumer price development of potatoes as well as sharp price drops of rice and sugar in the early to mid 2000s. The Philippines are similar to India in the sense that the selected product consumer prices seem to follow the general global upward trend. The consumer price increase of Bananas, Coconuts and Rice are above the global trend. The notable exception is sugar which has quite volatile trend with multiple sharp consumer price drops. Rwanda is interesting as its consumer price trend is erratic in comparison to the global generally increasing trend. The global consumer food price for Bananas, Cassava and Maize increase at a constant rate whereas sweet potatoes seem to decrease and stagnate globally around 2005. The Rwandan consumer price for the same goods is in a sharp decline starting in the early 1990s, again our assumption is that this decline is related to the Rwandan civil war. A strong increase in consumer prices can be noted starting in the mid 1990s, spearheaded by Cassava and Maize consumer price development.

4.2 Results

4.2.1 Correlation / VIF based method

4.2.1.1. Correlation overview The first step of this approach was to get an overview of pairwise correlations for each country's dataset, measured by pearson's correlation coefficient.

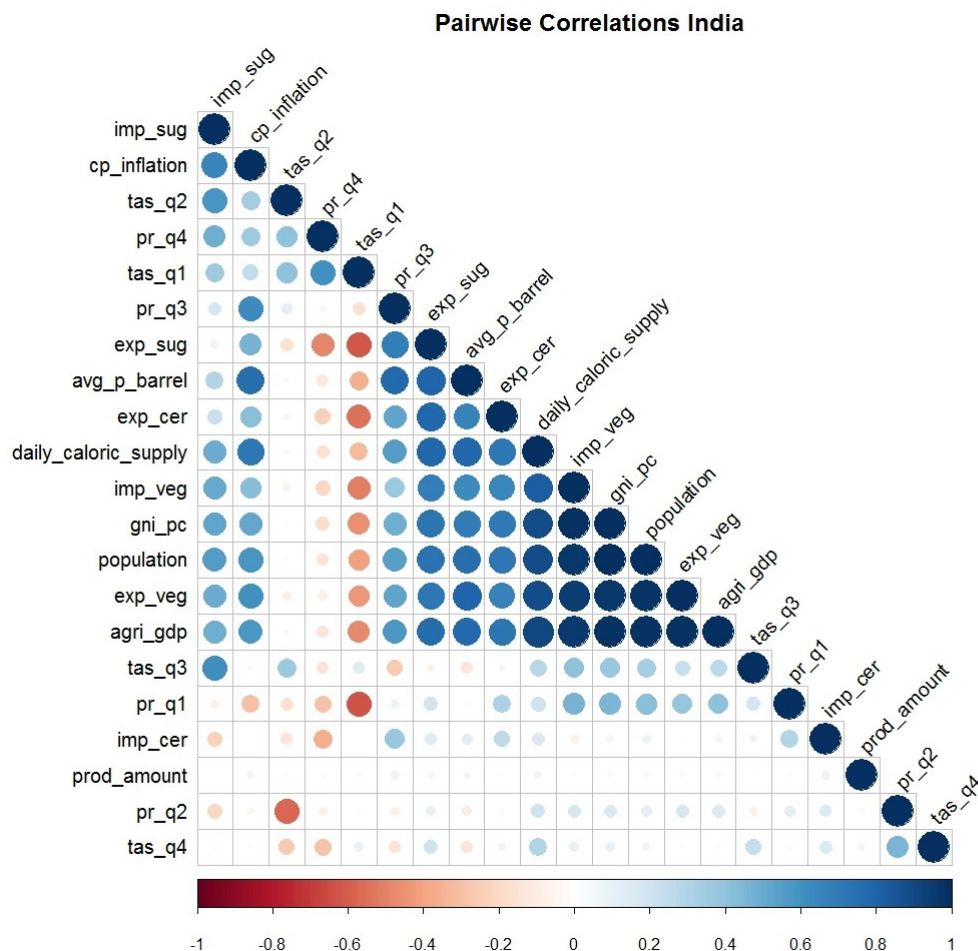


Figure 6: India Correlations

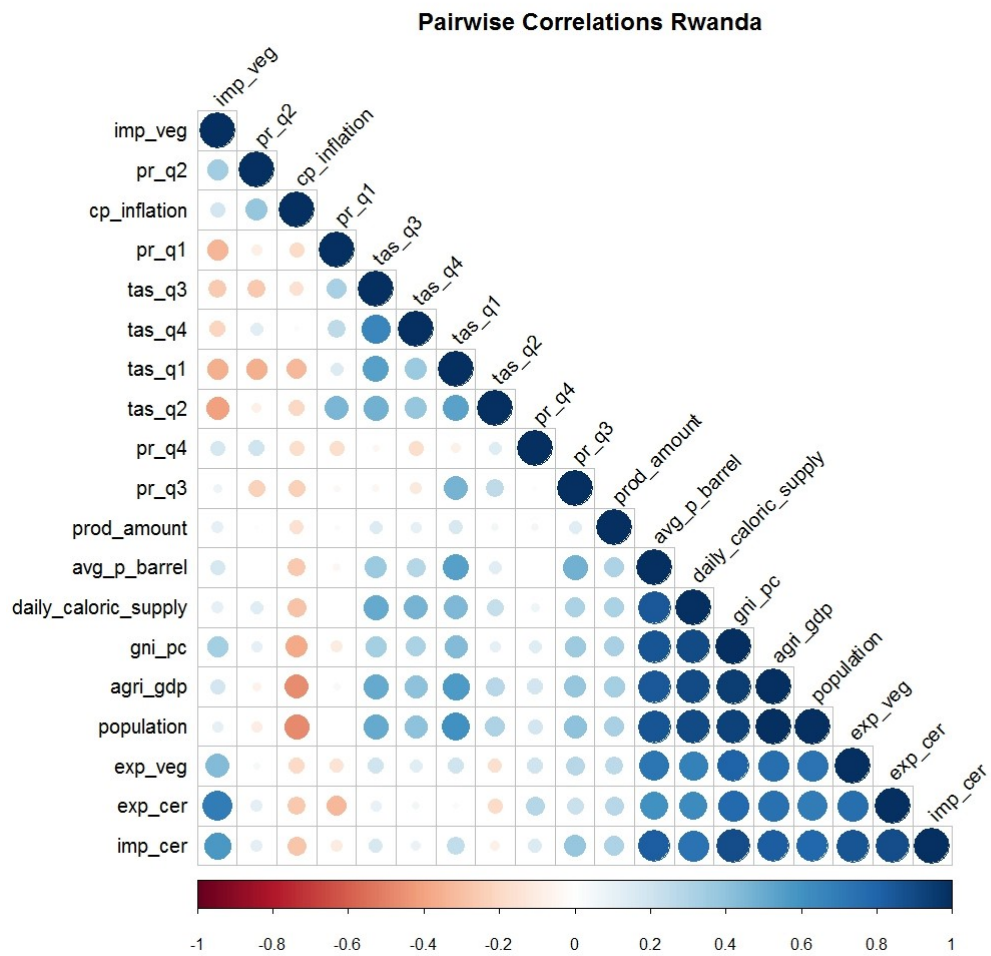


Figure 7: Rwanda Correlations

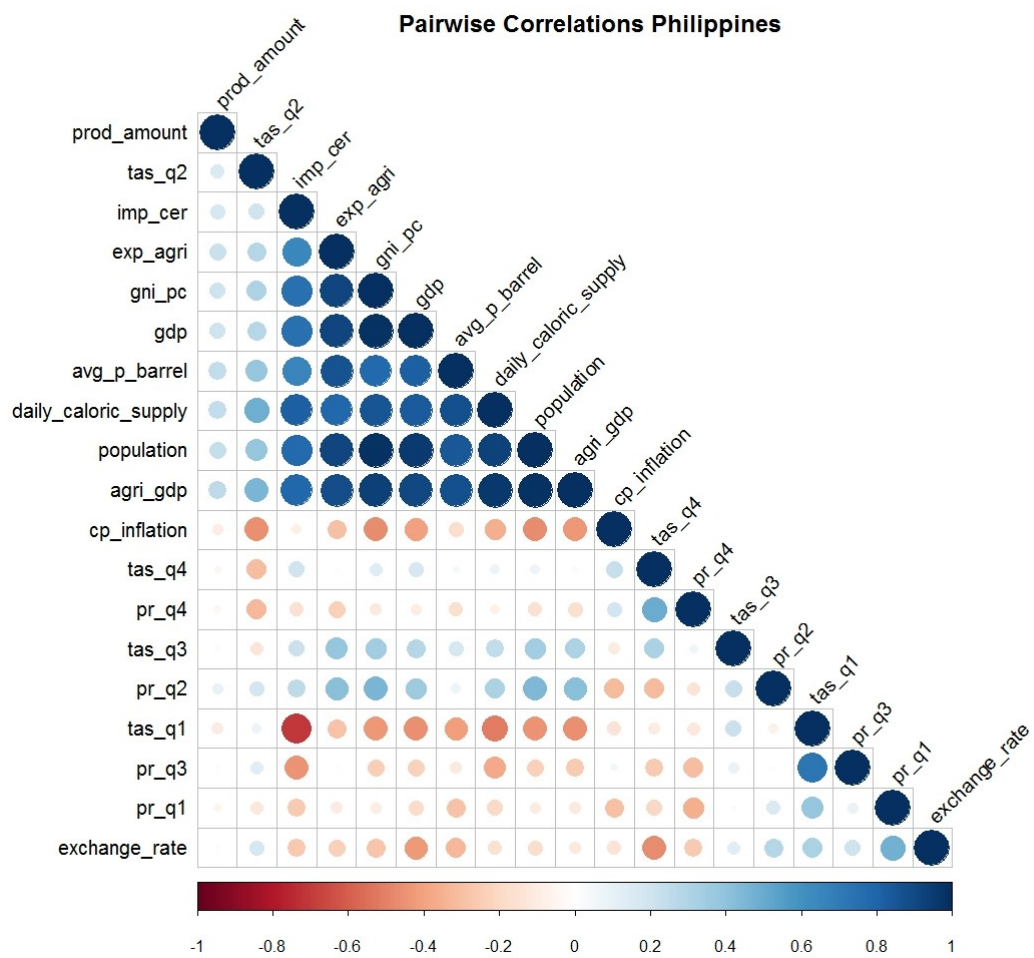


Figure 8: Philippines Correlations

Variables which are part of a correlation pair with coefficient ≥ 0.70 :

<u>India</u>	<u>Rwanda</u>	<u>Philippines</u>
gni_pc	gni_pc	gni_pc
agri_gdp	agri_gdp	agri_gdp
population	population	population
daily_caloric_supply	daily_caloric_supply	daily_caloric_supply
exp_veg	imp_cer	imp_cer
exp_sug	exp_cer	exp_agri
avg_p_barrel	avg_p_barrel	gdp
		tas_q1

Table 6: Correlations Variables

4.2.1.2. Linear Model without highly correlated variables Having removed highly correlated variables, a linear model is built from the remaining predictors:

India

Residuals:

Min	1Q	Median	3Q	Max
-0.68000	-0.23519	0.03321	0.22750	0.65974

Table 7: India Residuals

Coefficients:

Variable	Estimate	Std. Error	t value	Pr(> t)	Significant
(Intercept)	-4.83E-12	4.95E+01	0.000	1.000	
pr_q1	-5.53E+02	3.19E+02	-1.731	0.09243	x
pr_q2	-1.28E+02	1.00E+02	-1.272	0.21217	
pr_q3	-2.23E+02	2.37E+02	-0.944	0.35176	
pr_q4	2.42E+02	2.28E+02	1.063	0.29506	
tas_q1	-5.04E+02	3.24E+02	-1.554	0.12950	
tas_q2	2.77E+02	2.06E+02	1.347	0.18686	
tas_q3	-5.31E+02	5.90E+02	-0.900	0.37424	
tas_q4	1.66E+02	1.22E+02	1.360	0.18283	
prod_amount	6.34E+02	5.05E+01	12.561	2.52e-14	x
exp_cer	-3.41E+02	2.34E+02	-1.455	0.15482	
imp_sug	4.49E+02	6.66E+02	0.673	0.50548	
imp_veg	1.21E+03	3.64E+02	3.316	0.00218	x
imp_cer	5.62E+02	4.02E+02	1.396	0.17184	
cp_inflation	-3.04E+02	3.71E+02	-0.820	0.41774	

Table 8: India Coefficients

Residual standard error: 0.3466 on 34 degrees of freedom

Multiple R-squared: 0.9149, Adjusted R-squared: 0.8799

F-statistic: 26.12 on 14 and 34 DF, p-value: 3.915e-14

Rwanda

Residuals:

Min	1Q	Median	3Q	Max
-1.4427	-0.6418	-0.2439	0.3527	2.8037

Table 9: Rwanda Residuals

Coefficients:

Variable	Estimate	Std. Error	t value	Pr(> t)	Significant
(Intercept)	-9.63E-13	6.98E+01	0	1.00000	
pr_q1	1.18E+02	8.92E+01	1.327	0.18638	
pr_q2	9.90E+01	1.01E+02	0.98	0.3287	
pr_q3	3.38E+01	1.01E+02	0.335	0.73769	
pr_q4	1.01E+02	8.75E+01	1.151	0.25153	
tas_q1	2.77E+02	1.20E+02	2.318	0.0217	x
tas_q2	-1.28E+02	1.19E+02	-1.077	0.28327	
tas_q3	-1.79E+01	1.29E+02	-0.139	0.88996	
tas_q4	1.25E+02	1.11E+02	1.125	0.26234	
prod_amount	-2.19E+02	7.39E+01	-2.959	0.00355	x
exp_veg	1.26E+02	1.02E+02	1.234	0.21882	
imp_veg	2.64E+02	1.00E+02	2.643	0.00903	x
cp_inflation	-6.56E+01	8.64E+01	-0.759	0.44901	

Table 10: Rwanda Coefficients

Residual standard error: 0.9235 on 162 degrees of freedom

Multiple R-squared: 0.2059, Adjusted R-squared: 0.1471

F-statistic: 3.501 on 12 and 162 DF, p-value: 0.0001285

Philippines:

Residuals:

Min	1Q	Median	3Q	Max
-0.9879	-0.4897	-0.2891	0.3079	2.4465

Table 11: Philippines Residuals

Coefficients:

Variable	Estimate	Std. Error	t value	Pr(> t)	Significant
(Intercept)	-1.10E-13	1.02E+02	0	1.00000	
tas_q2	-5.72E+01	1.65E+02	-0.346	0.73059	
tas_q3	-1.15E+01	1.49E+02	-0.077	0.93899	
tas_q4	9.71E+01	1.67E+02	0.583	0.56205	
pr_q1	-1.01E+01	1.59E+02	-0.064	0.94933	
pr_q2	1.95E+02	1.22E+02	1.607	0.11324	
pr_q3	8.93E+00	1.18E+02	0.076	0.93986	
pr_q4	-4.00E+01	1.43E+02	-0.279	0.78124	
avg_p_barrel	4.28E+02	1.44E+02	2.978	0.00419	x
prod_amount	-3.63E+02	1.07E+02	-3.389	0.00124	x
exchange_rate	-1.71E+02	1.69E+02	-1.011	0.31585	
cp_inflation	-1.09E+02	1.45E+02	-0.750	0.45605	

Table 12: Philippines Coefficients

Residual standard error: 0.8693 on 60 degrees of freedom

Multiple R-squared: 0.3614, Adjusted R-squared: 0.2444

F-statistic: 3.087 on 11 and 60 DF, p-value: 0.002419

4.2.1.3. Linear Model with variables left after VIF-removal Since removing highly correlated variables leads to some potentially important variables being dropped, we applied VIF-removal to both the set of highly correlated variables and the remaining set. Creating another linear model with the resulting set of variables we obtained the following results

India:

Residuals:

Min	1Q	Median	3Q	Max
-0.66068	-0.22061	0.02824	0.20910	0.67543

Table 13: India Residuals

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	Significant
(Intercept)	4.16E-12	5.00E+01	0	1.0000	
population	1.41E+03	6.42E+02	2.188	0.0359	x
daily_caloric_supply	-4.90E+02	1.01E+03	-0.486	0.6301	
exp_sug	2.41E+03	2.54E+03	0.948	0.3501	
avg_p_barrel	-3.99E+02	8.40E+02	-0.475	0.6379	
pr_q1	-2.76E+02	1.69E+02	-1.634	0.1118	
pr_q2	-5.54E+02	5.53E+02	-1.002	0.3236	
pr_q3	-1.18E+03	1.05E+03	-1.130	0.2667	
pr_q4	1.31E+03	1.12E+03	1.172	0.2494	
tas_q2	-2.57E+02	5.92E+02	-0.435	0.6667	
tas_q3	2.03E+01	2.90E+02	0.070	0.9448	
tas_q4	-1.64E+02	2.40E+02	-0.682	0.5001	
prod_amount	6.34E+02	5.10E+01	12.418	5.51e-14	x
exp_cer	-9.57E+02	8.72E+02	-1.097	0.2806	
imp_cer	8.96E+02	8.99E+02	0.996	0.3264	
cp_inflation	-2.06E+02	6.71E+02	-0.307	0.7604	

Table 14: India Coefficients

Residual standard error: 0.35 on 33 degrees of freedom

Multiple R-squared: 0.9158, Adjusted R-squared: 0.8775

F-statistic: 23.92 on 15 and 33 DF, p-value: 1.758e-13

Rwanda

Residuals:

Min	1Q	Median	3Q	Max
-1.5524	-0.6039	-0.2119	0.3888	2.9797

Table 15: Rwanda Residuals

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	Significant
(Intercept)	-6.61E-13	6.91E+01	0	1.00000	
population	5.28E+02	4.79E+02	1.102	0.27214	
daily_caloric_supply	-4.03E+02	2.85E+02	-1.413	0.15966	
avg_p_barrel	3.42E+02	1.81E+02	1.892	0.06027	x
exp_cer	-1.72E+01	2.91E+02	-0.059	0.9531	
pr_q1	1.35E+02	1.00E+02	1.348	0.17956	
pr_q2	1.21E+02	1.40E+02	0.867	0.38741	
pr_q3	1.44E+01	1.15E+02	0.125	0.90036	
pr_q4	8.60E+01	1.02E+02	0.84	0.40223	
tas_q1	2.73E+01	1.88E+02	0.146	0.88442	
tas_q2	-1.56E+02	1.29E+02	-1210	0.22816	
tas_q3	2.02E+01	1.59E+02	0.127	0.89905	
tas_q4	9.00E+01	1.18E+02	0.761	0.44762	
prod_amount	-2.31E+02	7.38E+01	-3131	0.00208	x
exp_veg	-1.30E+02	1.57E+02	-0.831	0.4075	
imp_veg	2.09E+02	1.77E+02	1.180	0.2397	
cp_inflation	2.12E+01	1.08E+02	0.196	0.8446	

Table 16: Rwanda Coefficients

Residual standard error: 0.9138 on 158 degrees of freedom

Multiple R-squared: 0.2417, Adjusted R-squared: 0.1649

F-statistic: 3.148 on 16 and 158 DF, p-value: 0.0001125

Philippines

Residuals:

Min	1Q	Median	3Q	Max
-1.0930	-0.4379	-0.2995	0.2246	2.3262

Table 17: Philippines Residuals

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	Significant
(Intercept)	-1.14E-13	1.05E+02	0	1.00000	
daily_caloric_supply	4.35E+02	1.26E+03	0.346	0.7308	
exp_agri	3.70E+02	6.26E+02	0.591	0.55687	
imp_cer	-7.12E+01	5.27E+02	-0.135	0.89303	
tas_q1	-1.98E+02	3.95E+02	-0.502	0.61793	
tas_q2	-3.10E+01	2.90E+02	-0.107	0.91522	
tas_q3	-1.15E+01	2.58E+02	-0.044	0.96469	
tas_q4	4.17E+01	1.86E+02	0.225	0.82314	
pr_q1	2.46E+01	2.58E+02	0.095	0.92448	
pr_q2	-4.54E+01	3.47E+02	-0.131	0.89644	
pr_q3	1.92E+02	3.11E+02	0.619	0.53835	
pr_q4	-1.47E+01	2.45E+02	-0.060	0.95243	
avg_p_barrel	-2.64E+02	1.19E+03	-0.222	0.8252	
prod_amount	-3.64E+02	1.10E+02	-3.316	0.00161	x
exchange_rate	-2.01E+02	2.25E+02	-0.894	0.3754	
cp_inflation	-6.22E+01	1.96E+02	-0.317	0.7523	

Table 18: Philippines Coefficients

Residual standard error: 0.8884 on 56 degrees of freedom

Multiple R-squared: 0.3775, Adjusted R-squared: 0.2108

F-statistic: 2.264 on 15 and 56 DF, p-value: 0.01414

4.2.2 Lasso based method

Note: the left dotted line indicates the lambda leading to the model with minimal mse, which we chose as a base for variable selection. The right dotted line shows points at the lambda leading to a model for which the error is within one standard error of the minimum

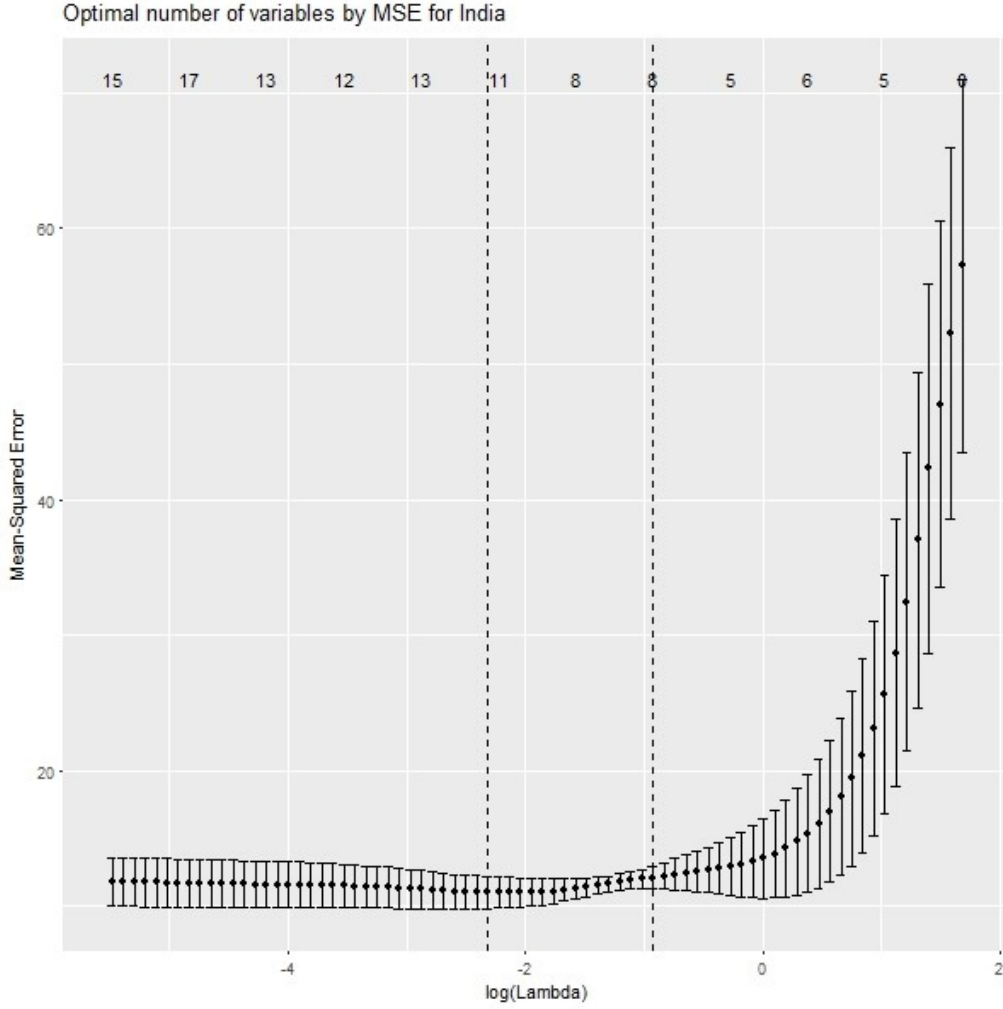


Figure 9: India Optimal Number of Variables

Important variables for India:

variables	coefs_minlambda	colors
tas_q2	8.203321e-01	green
prod_amount	4.289088e-05	green
exp_veg	2.596261e-06	green
agri_gdp	5.487558e-11	green
exp_cer	7.872368e-11	green
imp_sug	7.540223e-07	green
imp_veg	2.962602e-07	green
imp_cer	-2.776436e-06	red
pr_q2	-2.285270e-02	red
pr_q1	-1.124898e-02	red
tas_q3	8.462011e-01	green

Table 19: Important variables for India

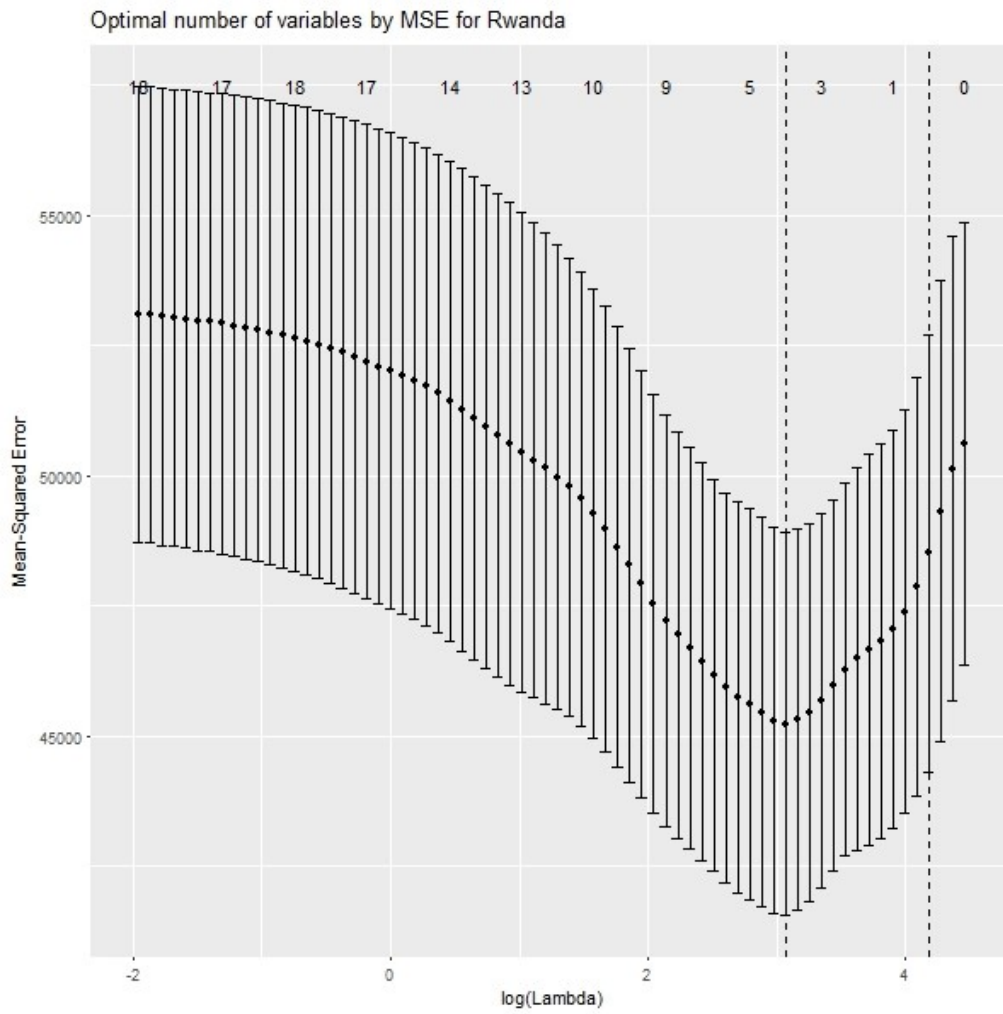


Figure 10: Rwanda Optimal Number of Variables

Important variables for Rwanda:

<u>variables</u>	<u>coefs_minlambda</u>	<u>colors</u>
tas_q4	1.543063e+00	green
avg_p_barrel	1.855989e-01	green
imp_cer	1.654824e-03	green
prod_amount	-2.160013e-05	red

Table 20: Important variables for Rwanda

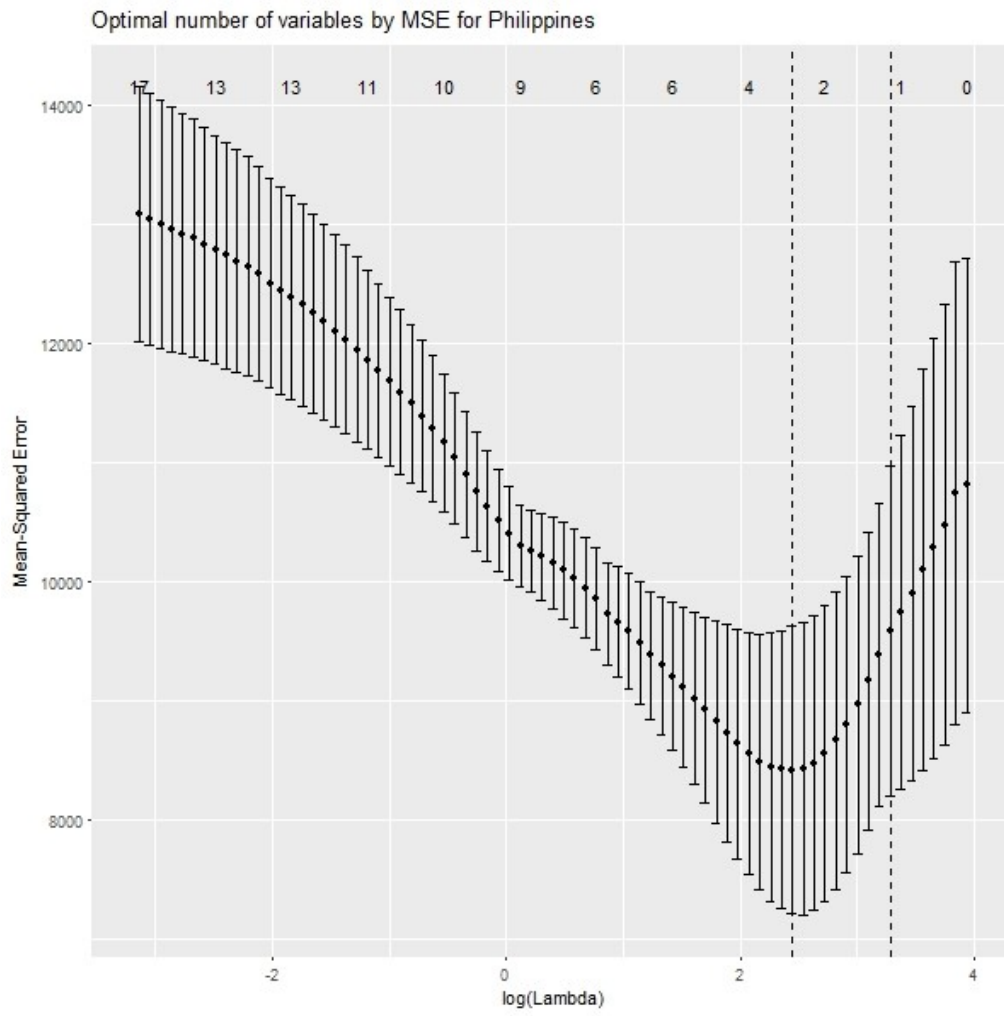


Figure 11: Philippines Optimal Number of Variables

Important variables for Philippines:

<u>variables</u>	<u>coefs_minlambda</u>
gdp	5.429905e-10
prod_amount	-3.595559e-06
avg_p_barrel	8.018443e-02

Table 21: Important variables for Philippines

4.2.3 Random Forest based method

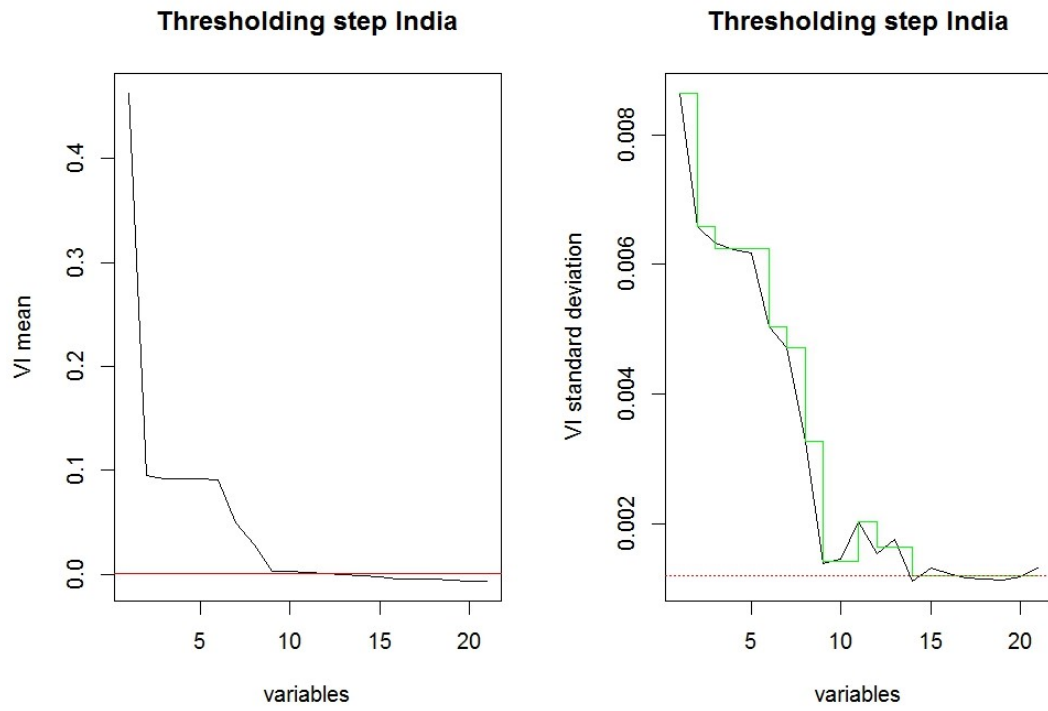


Figure 12: India Thresholding Step

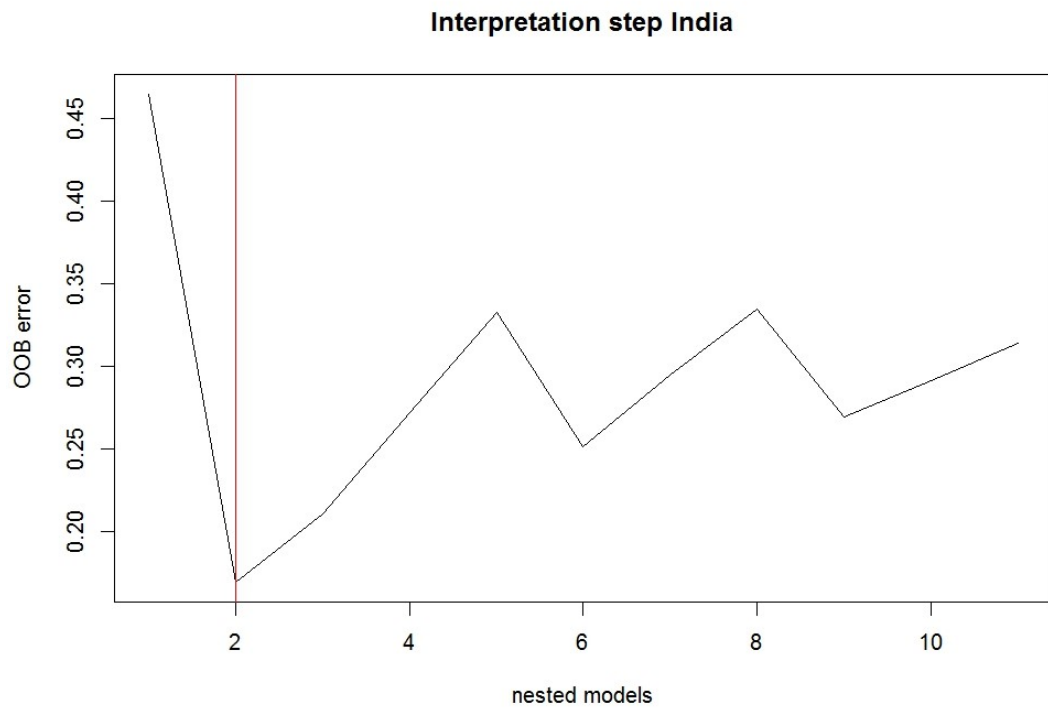


Figure 13: India interpretation Step

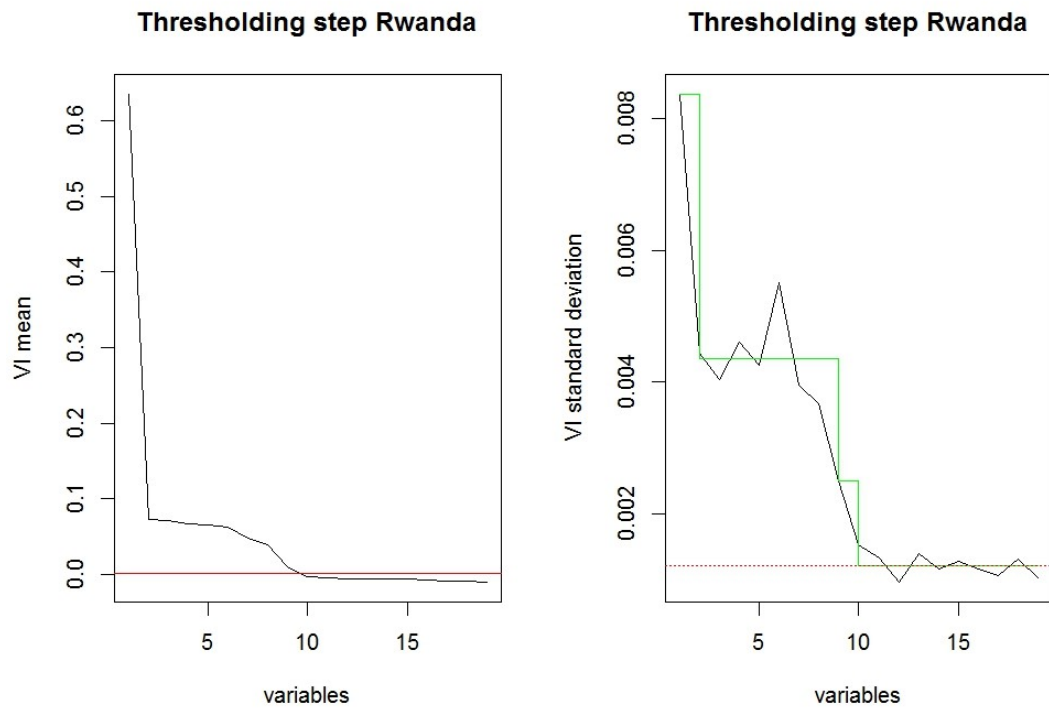


Figure 14: Rwanda Thresholding Step

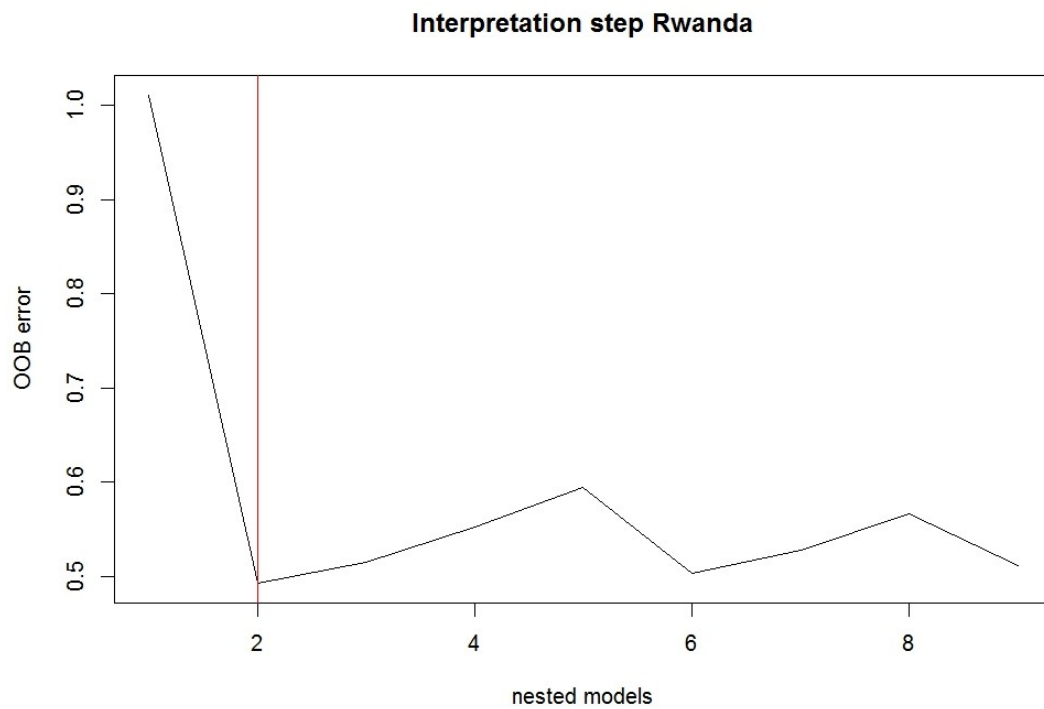


Figure 15: Rwanda interpretation Step

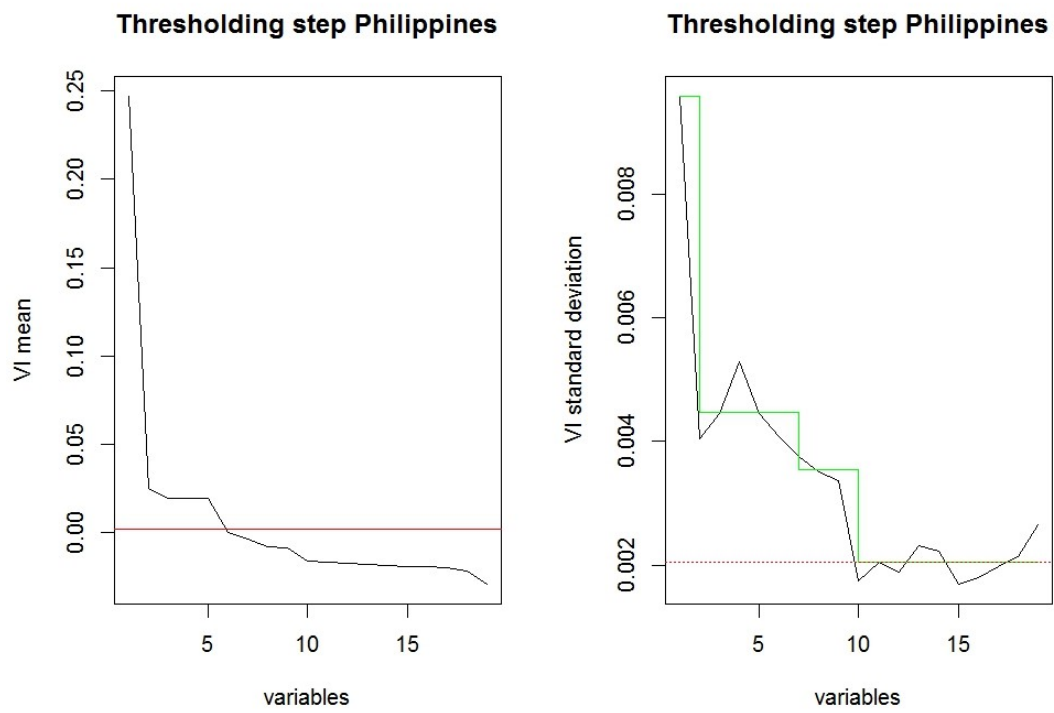


Figure 16: Philippines Thresholding Step

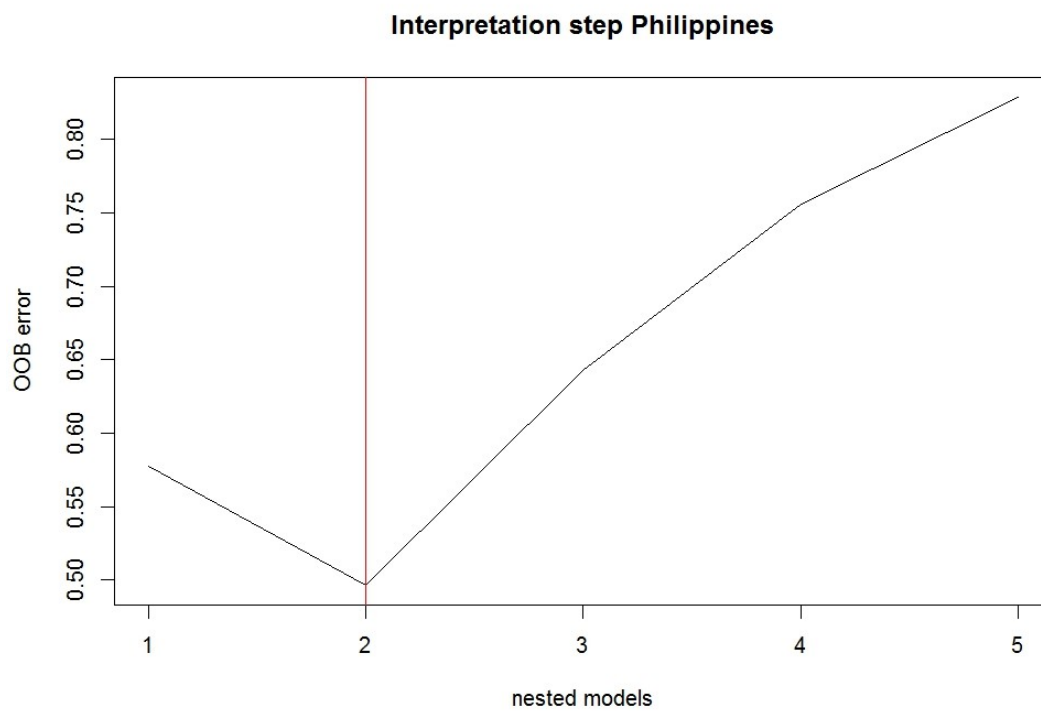


Figure 17: Philippines interpretation Step

India		Rwanda		Philippines	
threshold step	interpretation step	threshold step	interpretation step	threshold step	interpretation step
prod_amount	prod_amount	prod_amount	prod_amount	prod_amount	prod_amount
agri_gdp	agri_gdp	agri_gdp		agri_gdp	agri_gdp
gni_pc		gni_pc		gni_pc	
population		population		population	
imp_veg					
exp_veg		exp_veg			
daily_caloric_supply		daily_caloric_supply			
imp_sug					
cp_inflation					
imp_cer		imp_cer	imp_cer		
exp_sug					
		exp_cer			
		avg_p_barrel			
				gdp	

Table 22: Random Forest Selection

4.2.4 Summary

India:

The linear model with non highly correlated variables reveals that the amount of rain in the first quarter, the produced amount and the amount of imported vegetables significantly affect the price. The model made after VIF reduction has only two significant variables, population and produced amount. It appears that population, the only significant demand related factor, outweighs climatic and macroeconomic supply related factors in terms of predictive power in the linear models. However produced amount seems to be the most important factor since it is always included.

The lasso method gives a more differentiated impression. Again, produced amount seems to be important. Imports of sugar, vegetables and cereals as well as demand related factors exports of vegetables and cereals also have an impact on price. Climatic factors play a bigger role here since rain in the first half of the year and temperatures from April to September also appear to be a driving factor, probably because of their impact on the agricultural output. The fact that agricultural GDP is also on the list supports this.

The findings produced by the threshold step of the RF method generally go in the same direction as produced amount and agri_gdp seem to be important again as well as imports of sugar and vegetables and export of vegetables. In addition to that population size, gross national income per capita, daily caloric supply and inflation have an influence on price too. After the interpretation step only produced amount and agri_gdp are left.

The importance of the produced amount is evident as it is included in every result. On the demand side, population size and food exports appear to be the most influential factors.

Rwanda:

According to the findings produced by the linear model with no highly correlated variables, only supply related factors are to be considered. Produced amount appears to be very important, temperature in the first quarter and the amount of imported vegetables are also influential. The VIF based model reveals that apart from produced amount, food prices in Rwanda are also dependent on the petrol oil price. These findings are support by the results of the lasso model, as in addition to produced amount and oil price, rain in the fourth quarter of the year and the import of cereals are also an affecting food prices.

The RF model indicates, that in addition to agricultural output, demand related variables such as population size, GNI per capita, daily caloric supply and the amount of exported vegetables contribute to food price development. After the interpretation step only produced amount is left.

Again, the single most important factor appears to be produced amount. Combined with demographic factors and the dependency of Rwanda's agriculture on petroleum most of the food prices in this country can be explained.

Philippines:

The linear model with low correlated variables for the Philippines gives only two significant factors : oil price per barrel and produced amount. The VIF based model only has produced amount. While not adding any further information these findings are in line with the results for the other two countries. This is also supported by the findings obtained from the lasso model, as only gdp, produced amount and oil price per barrel appear to be important. The interpretation step of the RF model adds population and gni_pc to the list of relevant variables, produced_amount and agri_gdp are considered important as well. After the interpretation step only produced amount and agri_gdp remain.

In the Philippines, food prices appear to be mostly driven by the produced amount and the population, while other factors only play a minor role.

5 Conclusion

Erokhin highlighted the fact that one important contributing factor to food security are food prices in his work. The OECD identified an understanding on food price development as a paramount basis for sound food security policy decision making on the national level. With these observations in mind we started this work with the goal to see if we could replicate the findings of Erokhin and the OECD on a country specific level. The initial step was the country selection. We decided on India, Rwanda and the Philippines as this selection of countries provided us with a varied economic, political and climatic basis of research, which in turn allowed us to form a comprehensive conclusion if the global OECD findings were applicable on a country specific level. We will first contrast the global OECD findings on consumer food price development with the country specific selection of consumer goods food price development. All goods selected were chosen based on the Harmonized System Codes (HS Code 2017). We focused on the category 07-015.

The OECD analyzed annual consumer food price development of wheat, coarse grains rice and oil seeds from 1971 to 2007. The conclusion is that consumer food prices are volatile and price spikes are a common occurrence. The OECD has identified unfavorable climatic conditions in 2005 and 2007 in major crop producing regions as one factor contributing to an increase in consumer food prices (OECD, 2008). Further strong demand growth for food and animal feed led to increase in price development. Other factors contributing to consumer price trend development as noted by the OECD are the following: Macro-economic conditions such as GDP growth in developing countries and oil price development. The OECD predicted based on the above factors that global consumer food prices would increase in comparison to historic trends but also that consumer prices would decrease from the from 2007-2008 spike price level.

Our findings support this prediction as one can note a general upward trend in the global as well as national consumer food prices for our selection of goods as shown in section 4.1. Figure 2 shows that consumer prices of our selection of cereals, vegetables oils and sugar are increasing in accordance with global food consumer prices. One can observe that while this price development is above the historic trend and generally increasing, the price spike in 2007-2008 has normalized as predicted by the OECD. The different models used in the variable selection process gave some further insights in regard to which factors were most influential on consumer price development. Produced amount was the most influential supply side factor across all three countries. These findings are in accordance with OECD findings regarding demand development. The OECD further notes animal feed as driving demand factor, which is one limitation of our work as we have no data on life feed demand influences. Population growth was the most influential demand side factor for our country selection. Country specific difference in the influential factors could also be observed. Crop prices in the Philippines are mainly unaffected by any of the factors we considered apart from produced amount and population. India and Rwanda hat some interesting country specific influential factors. Climatic factors rain and temperature were important when looking at India, most likely as they influenced agricultural output. Further on a macro-economic level food exports were especially influential on price development in India. This in accordance with Erokhin's observation on food security being negatively influenced by an increase in exports. Food price development in Rwanda seems to hinge

exponentially on oil price development. The OECD states oil price development as a price influencing factor but price development in Rwanda seems to be more influenced by oil price development than the global trend. Our findings show that the global perspective and prediction of the OECD in regards to the consumer food price index hold true on a by country level.

OECD stance on an increasing trend of consumer food prices in regards to food security is that while the overall impact on developing countries is modest, urban poor populations will be strongly negatively impacted. This discrepancy of modest national impact to strong negative impact for the poor urban population is based on the share of disposable income available and its use to purchase food at higher consumer food prices. According to Erokhin another negative aspect of an increase in consumer food prices is that exporting goods becomes more profitable to producers from countries with a weak local currency (Erokhin, 2017). This can further exacerbate food security for the poorest populations of a country.

It appears that for any of the countries that we considered, population growth and the amount of domestically produced crops have the highest impact on local food prices. Whether this also holds true for other countries, remains to be seen. In any case both of these factors deserve further inspection, particularly focusing on the reasons and mechanisms behind population growth and production amounts.

References

- [1]. Food and Agriculture Organization of the United Nations. Food and Agriculture Data. Available online: <http://faostat.fao.org/beta/en/#home> (accessed on 15th February 2017).
- [2]. Erokhin, V. Establishing Food Security and Alternatives to International Trade in Emerging Economies; IGI Global: Hershey, PA, USA, 2017.
- [3] . Esturk, Oren, M.N. Impact of household socio-economic factors on food security: Case of Adana. Pak. J. Nutr. 2013, 13, 1–6.
- [4]. Smith, M.E. World Food Security. The Effect of U.S. Farm Policy; United States Department of Agriculture: Washington, WA, USA, 1990.
- [5]. Bühlmann, Peter, and Sara van de Geer. Statistics for High-Dimensional Data: Methods, Theory and Applications. Springer, 2013.
- [6]. OECD, (2008). Rising Food Prices: Causes and Consequences. Organisation for Economic Co-operation and Development (OECD). <http://www.oecd.org/trade/agricultural-trade/40847088.pdf> Retrieved 14th January, 2018.

Appendix

```
1  #this script shall load all final dataset files created in the preparation stage and
   # include the code for displaying
2  # explorative graphs and tables.
3
4
5  # plotting the population for the all countries and the world
6  source("..\Helper_functions\exploration_functions.r")
7
8  if(!require("reshape2")) install.packages("reshape2");library("reshape2")
9  if(!require("ggplot2")) install.packages("ggplot2");library("ggplot2")
10 if(!require("data.table")) install.packages("data.table");library("data.table")
11 if(!require("zoo")) install.packages("zoo");library("zoo")
12
13
14 #reading the data
15 world_population = read.csv("../Common_datasets\world_population.csv",
   stringsAsFactors = FALSE, sep = ",", header = TRUE)
16
17 # stacks a set of columns into a single column of data to be able to process it
18 world_population = melt(world_population, id=c("Year"), value.name = "population")
19
20 # calculating the percentage of the change in the population
21 for(i in unique(world_population$variable)){
22   # i is the name of the land
23   #print(i)
24   world_population = calcPercFixBaseyear(world_population,"variable",i,"Year",1991,"
   population", "percentage")
25 }
26
27 # creating and saving the plot
28 jpeg("../Ds_overview_plots//population_plot.jpg", width = 800, height = 480, units =
   "px", pointsize = 12,
29   quality = 75)
30 ggplot(world_population) + geom_line(aes(x=Year, y=percentage, colour=variable), size
   =1.2) +
31   scale_colour_manual(values=c("red","green","blue", "gray")) +
32   ylab(label="Growth Precentage") +
33   xlab("Year")
34 dev.off()
35
36 #
   #####
37
38
39 # plotting the production amount of the selected products for the specified countries
   compared to the world
40
41 # preparing the dataset
42 world_production = read.csv("../Common_datasets\world_production.csv",
   stringsAsFactors = FALSE, sep = ",", header = TRUE)
43 world_production$X = NULL
44 colnames(world_production) = c("Area", "Item", "1991", "1992", "1993", "1994", "1995",
   "1996", "1997", "1998", "1999", "2000", "2001", "2002", "2003", "2004", "2005",
   "2006", "2007", "2008", "2009", "2010", "2011", "2012", "2013", "2014", "2015")
45
46 world_production = melt(world_production, id=c("Area","Item"), value.name = "
   Production_Amount")
47 # select the intersting items
48 world_production = world_production[world_production$Item %in% c("Sugar cane", "Rice,
   paddy", "Wheat", "Potatoes",
49   "Bananas", "Coconuts",
```

```

50                                     "Cassava", "Beans, dry", "Maize"
                                     , "Sweet potatoes"),]
51 # remove some uninterstting itmes specified by a land
52 world_production = world_production[!(world_production$Area == "India" & world_
    production$Item %in% c("Cassava", "Bananas", "Beans, dry", "Maize", "Sweet
    potatoes", "Coconuts")),]
53
54 world_production = world_production[!(world_production$Area == "Philippines" & world_
    production$Item %in% c("Cassava", "Wheat", "Beans, dry", "Maize", "Sweet potatoes
    ", "Potatoes")),]
55
56 world_production = world_production[!(world_production$Area == "Rwanda" & world_
    production$Item %in% c("Wheat", "Sugar cane", "Coconuts")),]
57 colnames(world_production)[3] = "Year"
58 world_production$Year = as.numeric(levels(world_production$Year))[world_production$
    Year]
59 # normalize the production amount
60 #world_production$Production_Amount = scale(world_production$Production_Amount)
61 h = data.frame()
62 for(i in unique(world_production$Item)){
63     d = world_production[world_production$Item == i,]
64     for(j in unique(d$Area)){
65         # i is the name of the land
66         #print(i)
67         d = calcPercFixBaseyear(d,"Area",j,"Year",1991,"Production_Amount", "Percentage")
68     }
69     h = rbind(h,d)
70 }
71 world_production = h
72
73
74 # calling the plot function and get the plot
75 p1 = prodPlot(world_production, "India", c("Sugar cane", "Rice, paddy", "Wheat", "
    Potatoes"))
76 p2 = prodPlot(world_production, "World", c("Sugar cane", "Rice, paddy", "Wheat", "
    Potatoes"))
77 p3 = prodPlot(world_production, "Philippines", c("Sugar cane", "Bananas", "Coconuts",
    "Rice, paddy"))
78 p4 = prodPlot(world_production, "World", c("Sugar cane", "Bananas", "Coconuts", "Rice
    , paddy"))
79 # p5 = prodPlot(world_production, "Rwanda", c("Cassava", "Bananas", "Beans, dry", "
    Maize", "Sweet potatoes", "Potatoes", "Rice, paddy"))
80 # p6 = prodPlot(world_production, "World", c("Cassava", "Bananas", "Beans, dry", "
    Maize", "Sweet potatoes", "Potatoes", "Rice, paddy"))
81 p5 = prodPlot(world_production, "Rwanda", c("Cassava", "Bananas", "Maize", "Sweet
    potatoes"))
82 p6 = prodPlot(world_production, "World", c("Cassava", "Bananas", "Maize", "Sweet
    potatoes"))
83
84 # plot in one screnn and save the image
85 jpeg("../Ds_overview_plots//production.jpg", width = 1200, height = 800, units = "px"
    , pointsize = 12,
    quality = 75)
86
87 multiplot(p1, p3, p5,p2, p4,p6, cols=2)
88 dev.off()
89
90 #
    #####
91
92 # source FAO
93 price_index = read.csv("../Common_datasets\\Food_price_indices_data.csv",
    stringsAsFactors = FALSE, sep = ",")
94 price_index[,8:16] = NULL
95 price_index$Date = as.yearmon(price_index$Date, format = "%m/%Y")
96 price_index = price_index[!price_index$Date %in% c(1990,2016,2017,2018),]

```

```

97
98 # creating and saving the plot
99 jpeg("../Ds_overview_plots//price_index.jpg", width = 800, height = 480, units = "px"
    , pointsize = 12,
100     quality = 75)
101 ggplot(price_index, aes(x = Date)) +
102   geom_line(aes(y = Food.Price.Index, colour="Food")) +
103   geom_line(aes(y = Cereals.Price.Index, colour="Cereals")) +
104   geom_line(aes(y = Oils.Price.Index, colour="Oils")) +
105   geom_line(aes(y = Sugar.Price.Index, colour="Sugar")) +
106   scale_x_yearmon(format="%m/%Y", n=5)+
107   ylab(label="Price Index") +
108   xlab("Year")
109 dev.off()
110
111 #
112 #####
113
114 #plotting a bar chart for each item form 2010 - 2015
115 rdata = readRDS("../Processed_ds\\rwanda_fin.rds")
116 pdata = readRDS("../Processed_ds\\philippines_fin.rds")
117 idata = readRDS("../Processed_ds\\india_fin.rds")
118 for(i in unique(rdata$prod_name)){
119   rdata = calcPercPreBaseyear(rdata, "prod_name", i, "year", "prod_price")
120 }
121 for(i in unique(pdata$prod_name)){
122   pdata = calcPercPreBaseyear(pdata, "prod_name", i, "year", "prod_price")
123 }
124 for(i in unique(idata$prod_name)){
125   idata = calcPercPreBaseyear(idata, "prod_name", i, "year", "prod_price")
126 }
127 idata[idata$year == 2012 & idata$prod_name == "Potatoes", "prod_price_Percent"] = 0
128 b1 = prodBarPlot(rdata, "Rwanda")
129 b2 = prodBarPlot(pdata, "Philippines")
130 b3 = prodBarPlot(idata, "India")
131
132 jpeg("../Ds_overview_plots//barplot_price_change.jpg", width = 1200, height = 800,
    units = "px", pointsize = 12,
133     quality = 75)
134 multiplot(b1,b2,b3, cols=1)
135 dev.off()

```

```

1 #calculating the percentage of the change in a column's value on a fixed base year
2 calcPercFixBaseyear = function(ds, areacol, areaname, yearcol, baseyear,valuecol,
    perccol){
3   base = ds[ds[[yearcol]] == baseyear & ds[[areacol]] %in% areaname, ][[valuecol]]
4   if(is.null(ds[[perccol]])){
5     ds[[perccol]] = 0
6   }
7   #
8   for(i in baseyear: max(ds[[yearcol]])){
9     later = ds[ds[[yearcol]]==i & ds[[areacol]] %in% areaname, ][[valuecol]]
10    sub = later - base
11    ds[ds[[yearcol]] == i & ds[[areacol]] %in% areaname, ][[perccol]] = (sub / later)
12    * 100
13  }
14  return(ds)
15 }
16
17 # calculate the prcentage of changes in a colname value in a predefined year, where
    the base is for every change is the value from the previous year.
18 calcPercPreBaseyear = function(ds, areacol, areaname, yearcol, valuecol){
19   for(i in unique(ds[[yearcol]])){
20     base = ds[ds[[yearcol]] == i & ds[[areacol]] %in% areaname, ][[valuecol]]
21     later = ds[ds[[yearcol]] == i+1 & ds[[areacol]] %in% areaname, ][[valuecol]]

```

```

22     #if(length(later) == 0L) break
23     if(i == 2015L) break
24     sub = later - base
25     if(length(sub) == 0L)next
26     ds[ds[[yearcol]] == i+1 & ds[[areacol]] %in% areaname , paste(valuecol, "Percent"
    , sep = "_")]= (sub / later) * 100
27 }
28 ds[ds[[yearcol]] == min(ds[[yearcol]]) & ds[[areacol]] %in% areaname, paste(
    valuecol, "Percent", sep = "_")]= 0
29 return(ds)
30 }
31
32 # To plot production data with specific area and itmes
33 prodPlot = function(ds, area, items){
34     p = ggplot(data=ds[ds$Area == area & ds$Item %in% items,], aes(x=Year, y=Percentage
    , colour=Item)) +
35     geom_line() +
36     geom_point()+
37     ylim(-30, 75)+
38     ggtitle(label=area)+
39     ylab(label="Normalized Production Amount") +
40     xlab("Year")
41     return(p)
42 }
43
44 # bar plot for product price change
45
46 prodBarPlot = function(d, dname){
47     ggplot(d[d$year %in% c(2010:2015) ,c("year", "prod_name", "prod_price_Percent")],
    aes(x = year, y = prod_price_Percent)) +
48     geom_bar(aes(fill = prod_name), position = "dodge", stat="identity") +
49     ggtitle(label=dname)+
50     ylab(label="price change based on previous year") +
51     xlab("Year")
52 }
53
54 # Multiple plot function
55 #
56 # ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)
57 # - cols: Number of columns in layout
58 # - layout: A matrix specifying the layout. If present, 'cols' is ignored.
59 #
60 # If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),
61 # then plot 1 will go in the upper left, 2 will go in the upper right, and
62 # 3 will go all the way across the bottom.
63 #
64 multiplot <- function(..., plotlist=NULL, file, cols=1, layout=NULL) {
65     require(grid)
66
67     # Make a list from the ... arguments and plotlist
68     plots <- c(list(...), plotlist)
69
70     numPlots = length(plots)
71
72     # If layout is NULL, then use 'cols' to determine layout
73     if (is.null(layout)) {
74         # Make the panel
75         # ncol: Number of columns of plots
76         # nrow: Number of rows needed, calculated from # of cols
77         layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
78             ncol = cols, nrow = ceiling(numPlots/cols))
79     }
80
81     if (numPlots==1) {
82         print(plots[[1]])
83     }

```

```

84 } else {
85   # Set up the page
86   grid.newpage()
87   pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))
88
89   # Make each plot, in the correct location
90   for (i in 1:numPlots) {
91     # Get the i,j matrix positions of the regions that contain this subplot
92     matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))
93
94     print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
95                                     layout.pos.col = matchidx$col))
96   }
97 }
98 }
99
100 #function for plotting VSURF Objects
101 plotVsurf = function(iVsurfOb,iStep,iCountry){
102   header_prefix = "not specified"
103   if(iStep == "thres"){
104     header_prefix = "Thresholding step"
105   }
106   if(iStep == "interp"){
107     header_prefix = "Interpretation step"
108   }
109
110   plot(iVsurfOb,step = iStep, var.names = FALSE,
111        nvar.interp = length(iVsurfOb$vselect.thres), main = paste(header_prefix,
112                               iCountry))
113 }

```



```

1 #calculate average price per food per month for the whole country
2 avgPriceFoodMonth = function(ds,cm_name,mp_price,mp_year,mp_month){
3   dsfoods = unique(ds[[cm_name]])
4   ds$avg_price_prod_month = 0
5   for (k in min(ds[[mp_year]]):max(ds[[mp_year]])){
6     print(paste('year is ',k))
7     for (j in 1:NROW(dsfoods)){
8       a <- ds[ds[[mp_year]] == k & ds[[cm_name]] == dsfoods[j],]
9       print(paste('food is ',dsfoods[j]))
10      for (i in 1:12){
11        b <- a[a[[mp_month]] == i,]
12        if (NROW(ds[ds[[mp_year]] == k & ds[[cm_name]] == dsfoods[j] & ds[[mp_month]]
13              == i,$avg_price_prod_month) >0) {
14          ds[ds[[mp_year]] == k & ds[[cm_name]] == dsfoods[j] & ds[[mp_month]] == i,]
15            $avg_price_prod_month = sum(b[[mp_price]])/ NROW(b)
16          print(paste('month is ',i))
17        }
18      }
19    }
20    return(ds)
21  }
22
23
24
25
26 #average price per year
27 avgPriceFoodYear = function(ds,cm_name,mp_year,avg_price_prod_month){
28   dsfoods = unique(ds[[cm_name]])
29   ds$avg_price_prod_year = 0
30   for (x in min(ds[[mp_year]]):max(ds[[mp_year]])){
31     print(paste('year is ',x))
32     for (y in 1:NROW(dsfoods)){
33       print(paste('food is ',dsfoods[y]))

```

```

34     if(NROW(ds[ds[[mp_year]] == x & ds[[cm_name]] == dsfoods[y],]$avg_price_prod_
35         year) > 0){
36         ds[ds[[mp_year]] == x & ds[[cm_name]] == dsfoods[y],]$avg_price_prod_year =
37             sum(ds[ds[[mp_year]] == x & ds[[cm_name]] == dsfoods[y],][[avg_price_prod_
38                 _month]]) / NROW(ds[ds[[mp_year]] == x & ds[[cm_name]] == dsfoods[y],][[
39                     avg_price_prod_month]])
40     }
41 }
42
43
44
45 # average rain and temp per quarter
46 avgRainTempQuarter = function(ds, month, mp_year, pr, tas){
47
48     if (is.na(ds[[pr]]) || is.na(ds[[tas]])) {
49         message(paste("No missing values allowed!"))
50     } else {
51
52         ds$tas_q1 = 0
53         ds$tas_q2 = 0
54         ds$tas_q3 = 0
55         ds$tas_q4 = 0
56         ds$pr_q1 = 0
57         ds$pr_q2 = 0
58         ds$pr_q3 = 0
59         ds$pr_q4 = 0
60
61
62         for(z in min(ds[[mp_year]]):max(ds[[mp_year]])){
63
64             ds[ds[[mp_year]] == z ,]$pr_q1 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in% c
65                 ("1", "2", "3"),][[pr]])/3
66             ds[ds[[mp_year]] == z ,]$pr_q2 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in% c
67                 ("4", "5", "6"),][[pr]])/3
68             ds[ds[[mp_year]] == z ,]$pr_q3 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in% c
69                 ("7", "8", "9"),][[pr]])/3
70             ds[ds[[mp_year]] == z ,]$pr_q4 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in% c
71                 ("10", "11", "12"),][[pr]])/3
72             ds[ds[[mp_year]] == z ,]$tas_q1 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in%
73                 c("1", "2", "3"),][[tas]])/3
74             ds[ds[[mp_year]] == z ,]$tas_q2 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in%
75                 c("4", "5", "6"),][[tas]])/3
76             ds[ds[[mp_year]] == z ,]$tas_q3 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in%
77                 c("7", "8", "9"),][[tas]])/3
78             ds[ds[[mp_year]] == z ,]$tas_q4 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in%
79                 c("10", "11", "12"),][[tas]])/3
80         }
81         return(ds)
82     }
83 }
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```



```

12 for (i in 1:NROW(colnames(explan_vars)) ){
13   temptarget = colnames(explan_vars)[i]
14   tempexpvars = paste(colnames(explan_vars[,!(colnames(explan_vars) %in% temptarget
15     )]),collapse = "+")
16   tempformula = paste(temptarget,"~", tempexpvars, collapse = " ")
17
18   tempresults[i,1] = temptarget
19   tempresults[i,2] = VIF(lm( tempformula,data = explan_vars))
20 }
21 print(tempresults[order(tempresults$vif),])
22 #remove variable with highest VIF, calculate new VIF for remaining variables until
23   all VIF are below cutoff value
24 while(max(tempresults$vif) >= cutoffval){
25   tempresults = tempresults[!tempresults$vif == max(tempresults$vif),]
26   tempremvars = tempresults$variable
27   for(j in 1: NROW(tempremvars)){
28     temptarget = tempremvars[j]
29     tempexpvars = paste(tempremvars[!tempremvars %in% temptarget],collapse = "+")
30     tempformula = paste(temptarget,"~", tempexpvars, collapse = " ")
31
32     tempresults[j,1] = temptarget
33     tempresults[j,2] = VIF(lm( tempformula,data = explan_vars))
34   }
35
36   print("Remaining variables:")
37   print(tempresults[order(tempresults$vif),])
38   cat("\n")
39 }
40 return(tempresults)
41 }
42
43
44
45 #calculate most important variables with LASSO for Lambda where MSE is minimal
46 if(!require("glmnet")) install.packages("glmnet"); library("glmnet")
47 if(!require("plyr")) install.packages("plyr"); library("plyr")
48
49 #load the data
50 india = readRDS("..\Qfolder2\Q2_india_fin.rds")
51
52 #select explanatory variables
53 exp_var_in = c("prod_price","pr_q1","pr_q2","pr_q3","pr_q4","tas_q1","tas_q2","tas_q3",
54   "tas_q4",
55   "prod_amount","daily_caloric_supply","exp_sug","exp_veg","exp_cer","
56   imp_sug","imp_veg","imp_cer",
57   "agri_gdp","gni_pc","cp_inflation","avg_p_barrel","population")
58 india = india[,colnames(india) %in% exp_var_in ]
59
60 #function for LASSO method
61 impVarsLasso = function(ds,targ){
62
63   #1. initial variable selection and normalization
64   val = ds[[targ]]
65   x = model.matrix(ds[[targ]]~.-1 , ds[!colnames(ds) %in% targ])
66
67   #2. Applying the Lasso technique
68   lasso = glmnet(x = x, y = val, standardize = TRUE, alpha = 1)
69
70   fit = cv.glmnet(x = x, y = val, standardize = TRUE, type.measure = "mse", alpha=1,
71     nfolds=3)
72
73   #3. Results
74   #with lambda.min

```

```

73 lambda_min = which(fit$lambda == fit$lambda.min)
74
75 #selecting coefficients of variables at lambda where mse is minimal
76 tempmincoefs = as.data.frame(fit$glmnet.fit$beta[, which(fit$lambda ==
77   fit$lambda.min)])
78 mincoefs = data.frame(matrix(ncol = 2, nrow = (NROW(tempmincoefs))
79   ))
80 mincoefs$variables = as.vector(as.character(labels(tempmincoefs)[[1]]))
81 mincoefs$coefs_minlambda = as.vector(tempmincoefs[[1]])
82 mincoefs$X1 = NULL
83 mincoefs$X2 = NULL
84
85 #get names in the decreasing order they appear in when lambda is minimal
86 names = names(coef(lasso)[,ncol(coef(lasso))[order(coef(lasso)[,ncol(
87   coef(lasso))],decreasing=TRUE)])
88 names = names[!names %in% c("(Intercept)")]
89 names = as.data.frame(names)
90 colnames(names) = "variables"
91
92 #add coefficient to names
93 disp_colors = join(names,mincoefs, by = "variables" )
94 disp_colors = disp_colors[!disp_colors$variables %in% c("(Intercept)"),]
95
96 #set colors for variables when displayed in a graph
97 disp_colors$colors = 0
98 if(NROW(disp_colors[disp_colors$coefs_minlambda >0,])>0){
99   disp_colors[disp_colors$coefs_minlambda >0,]$colors = c("green")
100 }
101 if(NROW(disp_colors[disp_colors$coefs_minlambda <0,])>0){
102   disp_colors[disp_colors$coefs_minlambda <0,]$colors = c("red")
103 }
104
105 #create a list to store the result
106 resultset = vector("list",3)
107 resultset[[1]] = lasso
108 resultset[[2]] = fit
109 resultset[[3]] = disp_colors
110
111 return(resultset)
112 }
113
114 #function for finding most important variables based on random forest
115 impVarsRf = function(ds,targ){
116   result_rf = VSURF(ds[[targ]] ~ ., data = ds[!colnames(ds) %in% targ], ntree = 2000,
117     nfor.thres = 50, nmin = 1, nfor.interp = 25, nsd = 1,
118     nfor.pred = 25, nmj = 1, parallel = FALSE, ncores = detectCores() -
119     1,
120     clusterType = "PSOCK")
121   #create a list to store the result
122   resultset = vector("list",2)
123   resultset[[1]] = result_rf
124   resultset[[2]] = colnames(ds[!colnames(ds) %in% targ])
125   return(resultset)
126 }
127
128 1 source("../Helper_functions\\preparation_functions.R")
129 2
130 3 data <- read.csv(file="../wfp_market_food_prices.csv",head=TRUE,sep=",")
131 4 rain <- read.csv(file="../rain_india.csv",head=TRUE,sep=";")
132 5 temp <- read.csv(file="../temp_india.csv",head=TRUE,sep=";")
133 6 prodcrops <- read.csv(file="../prodcrops_india.csv",head=TRUE,sep=";")
134 7 daycal <- read.csv(file="../per_capita_calories_india.csv",head=TRUE,sep=";")

```

```

9 exports <- read.csv(file=".\\india_exp.csv",head=TRUE,sep=";")
10 imports <- read.csv(file=".\\india_imp.csv",head=TRUE,sep=";")
11 agrigdp <- read.csv(file=".\\agri_gdp.csv",head=TRUE,sep=";")
12 gni_pc <- read.csv(file=".\\gni_pc_india.csv",head=TRUE,sep=";")
13 inflation <- read.csv(file=".\\inflation_india.csv",head=TRUE,sep=";")
14 oilprice <- read.csv(file=".\\opec-oil-price-annually.csv",head=TRUE,sep=";")
15 population <- read.csv(file=".\\population-india.csv",head=TRUE,sep=";")
16
17 #replace commas with dots where necessary so we can convert to numeric
18 rain$pr = as.numeric(gsub(",", ".", gsub("\\.", "", rain$pr)))
19 temp$tas = as.numeric(gsub(",", ".", gsub("\\.", "", temp$tas)))
20 exports$exp_sug = as.numeric(gsub(",", ".", gsub("\\.", "", exports$exp_sug)))
21 exports$exp_veg = as.numeric(gsub(",", ".", gsub("\\.", "", exports$exp_veg)))
22 exports$exp_cer = as.numeric(gsub(",", ".", gsub("\\.", "", exports$exp_sug)))
23 imports$imp_sug = as.numeric(gsub(",", ".", gsub("\\.", "", imports$imp_sug)))
24 imports$imp_veg = as.numeric(gsub(",", ".", gsub("\\.", "", imports$imp_veg)))
25 imports$imp_cer = as.numeric(gsub(",", ".", gsub("\\.", "", imports$imp_cer)))
26 agrigdp$agri_gdp = as.numeric(gsub(",", ".", gsub("\\.", "", agrigdp$agri_gdp)))
27 gni_pc$gni_pc = as.numeric(gsub(",", ".", gsub("\\.", "", gni_pc$gni_pc)))
28 inflation$cp_inflation = as.numeric(gsub(",", ".", gsub("\\.", "", inflation$cp_
    inflation)))
29 oilprice$avg_p_barrel = as.numeric(gsub(",", ".", gsub("\\.", "", oilprice$avg_p_
    barrel)))
30 prodcrops$prod_amount_y = as.numeric(levels(prodcrops$prod_amount_y)[prodcrops$prod_
    amount_y])
31
32
33
34
35 #we only look at crops falling under HS Code 2017 06-15
36 #https://www.foreign-trade.com/reference/hscod.htm?cat=2
37 #top 4 crops in terms of produced amount: sugarcane, rice, wheat, potatoes
38 #according to statistical yearbook of india 2017
39 #http://www.mospi.gov.in/statistical-year-book-india/2017/177
40 indiafoods = c("Sugar","Rice","Wheat","Potatoes")
41 india = data[data$adm0_name == 'India' & data$mp_year >= 2001,]
42 india = india[india$cm_name %in% indiafoods, ]
43 india$cm_name = as.character(india$cm_name)
44
45 #calculate average price per food per month for the whole country
46 india = avgPriceFoodMonth(india,"cm_name","mp_price","mp_year","mp_month")
47 #calculate average price per food per year
48 india = avgPriceFoodYear(india,"cm_name","mp_year","avg_price_prod_month")
49
50 #remove unnecessary variables
51 india$adm0_id = NULL
52 india$adm1_id = NULL
53 india$adm1_name = NULL
54 india$mkt_id = NULL
55 india$mkt_name = NULL
56 india$cm_id = NULL
57 india$cur_id = NULL
58 india$cur_name = NULL
59 india$pt_id = NULL
60 india$pt_name = NULL
61 india$mp_commoditysource = NULL
62 india$mp_price = NULL
63 india = unique(india)
64 india$month = india$mp_month
65 india$mp_month = NULL
66 india$year = india$mp_year
67 india$mp_year = NULL
68
69
70
71

```

```

72
73 #prepare rain and temp data: mean amount of rain / mean temp for each quarter year
74
75 rain$ISO3 = NULL
76 rain$ISO2 = NULL
77 rain$year = rain$X.Year
78 rain$X.Year = NULL
79 rain$month = rain$Month
80 rain$Month = NULL
81
82 temp$year = temp$X.Year
83 temp$X.Year = NULL
84 temp$month = temp$Month
85 temp$Month = NULL
86
87 raintemp = merge(rain,temp[c("tas","year","month")],by=c("month","year"))
88
89
90
91 # average rain and temp per quarter
92 raintemp = avgRainTempQuarter(raintemp,"month","year","pr","tas")
93
94 # we've decided to base our analysis on years, so we delete month related columns
95 india$month = NULL
96 india$avg_price_prod_month = NULL
97 india = unique(india)
98
99
100
101 india = merge(india,unique(raintemp[c("tas_q1","tas_q2","tas_q3","tas_q4","pr_q1","pr_q2","pr_q3","pr_q4","year")]),by=c("year"))
102 india = merge(india,prodcrops[c("year","cm_name","prod_amount_y")],by=c("year","cm_name"))
103 india = merge(india,daycal[c("year","daily_caloric_supply")],by=c("year"))
104 india = merge(india,exports,by=c("year"))
105 india = merge(india,imports,by=c("year"))
106 india = merge(india,agrigdp,by=c("year"))
107 india = merge(india,gni_pc,by=c("year"))
108 india = merge(india,inflation ,by=c("year"))
109 india = merge(india,oilprice ,by=c("year"))
110 india = merge(india,population ,by=c("year"))
111
112 #renaming columns that appear in every country's data set
113 colnames(india)[colnames(india) %in% "avg_price_prod_year"] = "prod_price"
114 colnames(india)[colnames(india) %in% "prod_amount_y"] = "prod_amount"
115 colnames(india)[colnames(india) %in% "cm_name"] = "prod_name"
116 colnames(india)[colnames(india) %in% "adm0_name" ] = "country"
117 colnames(india)[colnames(india) %in% "um_id" ] = "prod_uid"
118 colnames(india)[colnames(india) %in% "um_name" ] = "prod_unit"
119
120
121 #save our dataset for later
122 saveRDS(india, ("\\.\\Processed_ds\\india_fin.rds"))
123 #cleanup
124 rm(list = setdiff(ls(), lsf.str()))

1 library(data.table)
2 if(!require("plyr")) install.packages("plyr"); library("plyr")
3 if(!require("Hmisc")) install.packages("Hmisc"); library("Hmisc")
4 if(!require("corrplot")) install.packages("corrplot"); library("corrplot")
5 if(!require("ggplot2")) install.packages("ggplot2");library("ggplot2")
6 if(!require("grid")) install.packages("grid");library("grid")
7 if(!require("gridExtra")) install.packages("gridExtra");library("gridExtra")
8 if(!require("data.table")) install.packages("data.table");library("data.table")
9
10 source("~/sps_ws1718/Helper_functions/preparation_functions.R")

```

```

11
12
13
14 # http://www.fao.org/faostat/en/#data/PP
15 data <- read.csv("FoodPrices.csv", sep = ",", stringsAsFactors = FALSE)
16 # temprature and rainfall data from 1991 - 2015
17 #source: http://sdwebx.worldbank.org/climateportal/index.cfm?page=downscaled_data_
    download&menu=historical
18 rain <- read.csv(file="Philippines_rain.csv", head=TRUE, sep=",")
19 temp <- read.csv(file="Philippines_temp.csv", head=TRUE, sep=",")
20 # Source: https://www.statista.com/statistics/262858/change-in-opec-crude-oil-prices-
    since-1960/
21 oil_prices <- read.csv("OilPrices.csv", head = TRUE, sep = ";", stringsAsFactors =
    FALSE)
22 # Source: http://www.fao.org/faostat/en/#data/OA
23 population <- read.csv("Population.csv", head = TRUE, sep = ",", stringsAsFactors =
    FALSE)
24 # Source: http://www.fao.org/faostat/en/#data/OA
25 Production_amount <- read.csv("ProductionAmount.csv", head = TRUE, sep = ",",
    stringsAsFactors = FALSE)
26 # https://data.worldbank.org/indicator/NY.GNP.PCAP.KD?locations=RW
27 GNI <- read.csv("GNI.csv", head = TRUE, sep = ",", stringsAsFactors = FALSE)
28 # Source: http://www.fao.org/faostat/en/#data/OA
29 exchange_rate <- read.csv("ExchangeRate.csv", head = TRUE, sep = ",",
    stringsAsFactors = FALSE)
30 # https://data.worldbank.org/indicator/NY.GDP.MKTP.CD
31 GDP <- read.csv("GDP.csv", head = TRUE, sep = ",", stringsAsFactors = FALSE)
32 # https://data.worldbank.org/indicator/FP.CPI.TOTL.ZG?locations=RW
33 Inflation <- read.csv("Inflation.csv", head = TRUE, sep = ",", stringsAsFactors =
    FALSE)
34 # https://data.worldbank.org/indicator/NV.AGR.TOTL.KD?locations=RW
35 Agriculture_GDP <- read.csv("AgricultureGDP.csv", head = TRUE, sep = ",",
    stringsAsFactors = FALSE)
36 #https://psa.gov.ph/nap-press-release/data-charts
37 importData <- read.csv("imports.csv", head = TRUE, sep = ",", stringsAsFactors =
    FALSE)
38 #https://psa.gov.ph/nap-press-release/data-charts
39 exportData <- read.csv("exports.csv", head = TRUE, sep = ",", stringsAsFactors =
    FALSE)
40 # Source: https://ourworldindata.org/food-per-person
41 dpccs <- read.csv("daily-per-capita-supply-of-calories.csv", head = TRUE, sep = ",",
    stringsAsFactors = FALSE)
42
43
44 # I take only the important variables
45 data <- data[, c("Item", "Year", "Unit", "Value", "Flag", "Flag.Description")]
46
47 # choose a number of the most important products based of the production quantity
48 # Sweet potatoes Rice, paddy Potatoes Maize Cassava Bananas Beans, dry
49 data <- data[data$Item %in% c("Sugar cane", "Bananas", "Coconuts", "Rice, paddy"),]
50
51 ### rain and temp data
52 #replace commas with dots where necessary so we can convert to numeric
53 rain$pr = as.numeric(gsub(",", ".", gsub("\\.", "", rain$pr)))
54 temp$tas = as.numeric(gsub(",", ".", gsub("\\.", "", temp$tas)))
55
56 rain$ISO3 = NULL
57 rain$ISO2 = NULL
58 rain$year = rain$X.Year
59 rain$X.Year = NULL
60 rain$month = rain$Month
61 rain$Month = NULL
62
63 temp$year = temp$X.Year
64 temp$X.Year = NULL
65 temp$month = temp$Month

```

```

66 temp$Month = NULL
67
68 raintemp = merge(rain,temp[c("tas","Year","month")],by=c("month","Year"))
69
70 #calling the helper function
71 raintemp = avgRainTempQuarter(raintemp,"month","Year","pr","tas")
72
73
74 data = merge(data,unique(raintemp[c("tas_q1","tas_q2","tas_q3","tas_q4","pr_q1","pr_
    q2","pr_q3","pr_q4","Year")]),by=c("Year"))
75
76 #
    #####
77
78 ### read the oil prices data
79
80 colnames(oil_prices) <- c("Year", "oil_avarage_price_per_barrel")
81 #replace commas with dots where necessary so we can convert to numeric
82 oil_prices$oil_avarage_price_per_barrel = as.numeric(gsub(",",".", gsub("\\\\.", "",
    oil_prices$oil_avarage_price_per_barrel)))
83
84 data <- merge(x = data, y = oil_prices, by= "Year", all.x = TRUE)
85
86 #
    #####
87
88 ### read the Population data
89 population <- population[, c("Year", "Unit", "Value")]
90 population$Unit <- 1000
91 colnames(population) <- c("Year","PopulationUnit","PopulationValue")
92 data <- merge(x = data, y = population, by= "Year", all.x = TRUE)
93 #
    #####
94
95 ### Production Amount
96 Production_amount <- Production_amount[, c("Year", "Item", "Value")]
97 colnames(Production_amount)[3] <- "ProductionAmount"
98 #Production_amount$ProductionAmount = as.numeric(levels(data$Production_amount))[data
    $Production_amount]
99
100 data <- merge(x = data, y = Production_amount, by= c("Year", "Item"), all.x = TRUE)
101
102 #
    #####
103
104 # GNI per capita, Atlas method (current US$)
105 GNI <- GNI[GNI$Country.Name == "Philippines",]
106 GNI[1:35] <- NULL
107 GNI[c("X2017", "X")] <- NULL
108 colnames(GNI) <- c("1991", "1992", "1993", "1994", "1995", "1996", "1997", "1998", "
    1999", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008", "
    2009", "2010", "2011", "2012", "2013", "2014", "2015")
109 GNI <- as.data.frame(t(GNI))
110 setDT(GNI, keep.rownames = TRUE)[,]
111 colnames(GNI) <- c("Year", "GNI")
112
113 #GNI$GNI = as.numeric(gsub(",",".", gsub("\\\\.", "", GNI$GNI)))
114 #GNI$Year = as.numeric(gsub(",",".", gsub("\\\\.", "", GNI$Year)))
115
116
117 data <- merge(x = data, y = GNI, by= "Year", all.x = TRUE)
118

```

```

119 #
    #####

120 # Exchange rate
121 exchange_rate <- exchange_rate[, c("Year", "Value")]
122 colnames(exchange_rate)[2] <- "ExchangeRate"
123 data <- merge(x = data, y = exchange_rate, by= "Year", all.x = TRUE)
124
125 #
    #####

126 # GDP (current US$)
127 GDP <- GDP[GDP$Country.Name == "Philippines",]
128 GDP[1:35] <- NULL
129 GDP[c("X2016", "X2017", "X")] <- NULL
130 colnames(GDP) <- c("1991", "1992", "1993", "1994", "1995", "1996", "1997", "1998", "
    1999", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008", "
    2009", "2010", "2011", "2012", "2013", "2014", "2015")
131 GDP <- as.data.frame(t(GDP))
132 setDT(GDP, keep.rownames = TRUE)[]
133
134 colnames(GDP) <- c("Year", "GDP")
135 data <- merge(x = data, y = GDP, by= "Year", all.x = TRUE)
136
137 #
    #####

138 # Inflation, GDP deflator (annual %)
139 Inflation <- Inflation[Inflation$Country.Name == "Philippines",]
140 Inflation[1:35] <- NULL
141 Inflation[c("X2016", "X2017", "X")] <- NULL
142 colnames(Inflation) <- c("1991", "1992", "1993", "1994", "1995", "1996", "1997", "
    1998", "1999", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "
    2008", "2009", "2010", "2011", "2012", "2013", "2014", "2015")
143 Inflation <- as.data.frame(t(Inflation))
144 setDT(Inflation, keep.rownames = TRUE)[]
145
146 colnames(Inflation) <- c("Year", "Inflation")
147
148 #replace commas with dots where necessary so we can convert to numeric
149 #Inflation$Inflation = as.numeric(gsub(",", ".", gsub("\\.", "", Inflation$Inflation)
    ))
150
151 data <- merge(x = data, y = Inflation, by= "Year", all.x = TRUE)
152
153 #
    #####

154 # Agriculture GDP
155 # Agriculture, value added (constant 2010 US$)
156 Agriculture_GDP <- Agriculture_GDP[Agriculture_GDP$Country.Name == "Philippines",]
157 Agriculture_GDP[1:35] <- NULL
158 Agriculture_GDP[c("X2016", "X2017", "X")] <- NULL
159 colnames(Agriculture_GDP) <- c("1991", "1992", "1993", "1994", "1995", "1996", "1997"
    , "1998", "1999", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007",
    "2008", "2009", "2010", "2011", "2012", "2013", "2014", "2015")
160 Agriculture_GDP <- as.data.frame(t(Agriculture_GDP))
161 setDT(Agriculture_GDP, keep.rownames = TRUE)[]
162
163 colnames(Agriculture_GDP) <- c("Year", "Agriculture_GDP")
164
165 #replace commas with dots where necessary so we can convert to numeric
166 #Agriculture_GDP$Agriculture_GDP = as.numeric(gsub(",", ".", gsub("\\.", "",
    Agriculture_GDP$Agriculture_GDP)))
167
168 data <- merge(x = data, y = Agriculture_GDP, by= "Year", all.x = TRUE)

```



```

169
170 #
#####

171 #Import of Agricultural Products
172 #-- cereal
173 importData <- importData[importData$ITEM == "Cereals",]
174 importData[,c("X2016","ITEM")] <- NULL
175 colnames(importData) <- c("1998", "1999", "2000", "2001", "2002", "2003", "2004", "
2005", "2006", "2007", "2008", "2009", "2010", "2011", "2012", "2013", "2014", "
2015")
176 importData <- as.data.frame(t(importData))
177 setDT(importData, keep.rownames = TRUE)[]
178
179 colnames(importData) <- c("Year", "Import")
180
181 #replace commas with dots where necessary so we can convert to numeric
182 importData$Import = as.numeric(gsub(",", ".", gsub("\\.", "", importData$Import)))
183
184 data <- merge(x = data, y = importData, by= "Year", all.x = TRUE)
185
186 #
#####

187 #Export of Agricultural Products
188 #Bananas - Coconut Oil - Copra Oil, Coconut - Mango - Pineapple - Sugar
189 exportData <- exportData[exportData$ITEM == "AgriculturalProducts",]
190 exportData[,c("X2016","ITEMS")] <- NULL
191 colnames(exportData) <- c("1998", "1999", "2000", "2001", "2002", "2003", "2004", "
2005", "2006", "2007", "2008", "2009", "2010", "2011", "2012", "2013", "2014", "
2015")
192 exportData <- as.data.frame(t(exportData))
193 setDT(exportData, keep.rownames = TRUE)[]
194
195 colnames(exportData) <- c("Year", "Export")
196
197 #replace commas with dots where necessary so we can convert to numeric
198 exportData$Export = as.numeric(gsub(",", ".", gsub("\\.", "", exportData$Export)))
199
200 data <- merge(x = data, y = exportData, by= "Year", all.x = TRUE)
201
202 #
#####

203 # Average daily per capita caloric supply, measured in kilocalories per person per
day.
204
205 dpccs <- dpccs[dpccs$Entity == "Philippines", c("Year","Daily.caloric.supply..FAO
..2017....kcal.person.day.")]
206 colnames(dpccs)[2] <- "daily_caloric_supply"
207 data <- merge(x = data, y = dpccs, by= "Year", all.x = TRUE)
208
209 # 2014 and 2015 there is no data, replace with median
210 data$daily_caloric_supply[data$Year == "2014"] <- mean(unique(data$daily_caloric_
supply[data$Year %in% c("2012", "2011", "2010", "2009", "2008")]))
211 data$daily_caloric_supply[data$Year == "2015"] <- mean(unique(data$daily_caloric_
supply[data$Year %in% c("2013", "2012", "2011", "2010", "2009")]))
212
213 #removing rows with NAs in any column
214 data1 <- data[complete.cases(data),]
215
216 #renameing columns for final use
217 colnames(data1)[colnames(data1) %in% "Value"] = "prod_price"
218 colnames(data1)[colnames(data1) %in% "ProductionAmount" ] = "prod_amount"
219 colnames(data1)[colnames(data1) %in% "Year" ] = "year"
220 colnames(data1)[colnames(data1) %in% "Item" ] = "prod_name"

```



```

221 colnames(data1)[colnames(data1) %in% "PopulationValue" ] = "population"
222 colnames(data1)[colnames(data1) %in% "oil_avarage_price_per_barrel" ] = "avg_p_barrel"
223
223 colnames(data1)[colnames(data1) %in% "GNI" ] = "gni_pc"
224 colnames(data1)[colnames(data1) %in% "GDP" ] = "gdp"
225 colnames(data1)[colnames(data1) %in% "Inflation" ] = "cp_inflation"
226 colnames(data1)[colnames(data1) %in% "Agriculture_GDP" ] = "agri_gdp"
227 colnames(data1)[colnames(data1) %in% "Import" ] = "imp_cer"
228 colnames(data1)[colnames(data1) %in% "Export" ] = "exp_agri"
229 colnames(data1)[colnames(data1) %in% "ExchangeRate" ] = "exchange_rate"
230 colnames(data1)[colnames(data1) %in% "Flag" ] = "flag"
231 colnames(data1)[colnames(data1) %in% "Flag.Description" ] = "flag.description"
232 colnames(data1)[colnames(data1) %in% "PopulationUnit" ] = "population_unit"
233 colnames(data1)[colnames(data1) %in% "Unit" ] = "unit"
234
235
236 colnames(data1)
237
238 #save our dataset for later
239 saveRDS(data1, ("~/sps_ws1718/Processed_ds/philippines_fin.rds"))
240
241
242
243
244 #cleanup
245 rm(list = setdiff(ls(), lsf.str()))

1
2 source("../Helper_functions\\preparation_functions.R")
3
4 if(!require("plyr")) install.packages("plyr"); library("plyr")
5 if(!require("Hmisc")) install.packages("Hmisc"); library("Hmisc")
6 if(!require("corrplot")) install.packages("corrplot"); library("corrplot")
7 if(!require("ggplot2")) install.packages("ggplot2"); library("ggplot2")
8 if(!require("grid")) install.packages("grid"); library("grid")
9 if(!require("gridExtra")) install.packages("gridExtra"); library("gridExtra")
10 if(!require("data.table")) install.packages("data.table"); library("data.table")
11
12
13 # http://www.fao.org/faostat/en/#data/PP
14 data = read.csv("../Common_datasets\\Rwanda_datasets\\Food_prices.csv", sep = ",",
15 stringsAsFactors = FALSE)
16 # temprature and rainfall data from 1991 - 2015
17 #source: http://sdwebx.worldbank.org/climateportal/index.cfm?page=downscaled_data_
18 download&menu=historical
19 temp = read.csv("../Common_datasets\\Rwanda_datasets\\Rwanda_temp.csv", head = TRUE,
20 sep = ";", stringsAsFactors = FALSE)
21 rain = read.csv("../Common_datasets\\Rwanda_datasets\\Rwanda_rainfall.csv", head =
22 TRUE, sep = ";", stringsAsFactors = FALSE)
23 # Source: https://www.statista.com/statistics/262858/change-in-opec-crude-oil-prices-
24 since-1960/
25 oil_prices = read.csv("../Common_datasets\\Rwanda_datasets\\Oil_prices.csv", head =
26 TRUE, sep = ";", stringsAsFactors = FALSE)
27 # Source: http://www.fao.org/faostat/en/#data/OA
28 population = read.csv("../Common_datasets\\Rwanda_datasets\\Population.csv", head =
29 TRUE, sep = ";", stringsAsFactors = FALSE)
30 # Source: http://www.fao.org/faostat/en/#data/OA
31 Production_amount = read.csv("../Common_datasets\\Rwanda_datasets\\Production_Amount.
32 csv", head = TRUE, sep = ";", stringsAsFactors = FALSE)
33 # https://data.worldbank.org/indicator/NY.GNP.PCAP.KD?locations=RW
34 GNI = read.csv("../Common_datasets\\Rwanda_datasets\\GNI.csv", head = TRUE, sep = ";",
35 stringsAsFactors = FALSE)
36 # Source: http://www.fao.org/faostat/en/#data/OA
37 exchange_rate = read.csv("../Common_datasets\\Rwanda_datasets\\Exchange_rate.csv",
38 head = TRUE, sep = ";", stringsAsFactors = FALSE)
39 # https://data.worldbank.org/indicator/NY.GDP.MKTP.CD

```

```

30 GDP = read.csv(".\\Common_datasets\\Rwanda_datasets\\GDP.csv", head = TRUE, sep = ",",
  , stringsAsFactors = FALSE)
31 # https://data.worldbank.org/indicator/FP.CPI.TOTL.ZG?locations=RW
32 Inflation = read.csv(".\\Common_datasets\\Rwanda_datasets\\Inflation.csv", head =
  TRUE, sep = ",", stringsAsFactors = FALSE)
33 # https://data.worldbank.org/indicator/NV.AGR.TOTL.KD?locations=RW
34 Agriculture_GDP = read.csv(".\\Common_datasets\\Rwanda_datasets\\Agriculture_GDP.csv"
  , head = TRUE, sep = ",", stringsAsFactors = FALSE)
35 # Source: http://rwanda.opendataforafrica.org/UNCTADMTMEIWC2017/merchandise-trade-
  matrix-product-groups-exports-and-imports-in-thousands-of-dollars-annual
  -1995-2016
36 Vegetables = read.csv(".\\Common_datasets\\Rwanda_datasets\\Vegetables.csv", head =
  TRUE, sep = ",", stringsAsFactors = FALSE)
37 Cereals = read.csv(".\\Common_datasets\\Rwanda_datasets\\Cereal.csv", head = TRUE,
  sep = ",", stringsAsFactors = FALSE)
38 # Source: https://ourworldindata.org/food-per-person
39 dpccs = read.csv(".\\Common_datasets\\Rwanda_datasets\\daily-per-capita-caloric-
  supply.csv", head = TRUE, sep = ",", stringsAsFactors = FALSE)
40
41
42
43 # I take only the important variables
44
45 data = data[, c("Item", "Year", "Unit", "Value", "Flag", "Flag.Description")]
46
47 # choose a number of the most important products based of the production quantity
48 # Sweet potatoes Rice, paddy Potatoes Maize Cassava Bananas Beans, dry
49
50 data = data[data$Item %in% c("Cassava", "Bananas", "Beans, dry", "Maize", "Sweet
  potatoes", "Potatoes", "Rice, paddy"),]
51
52 ### rain and temp data
53
54 rain$pr = rain$i..pr
55 rain$Year = rain$X.Year
56 rain$month = rain$Month
57 rain[, c("IS03", "IS02", "X.Year", "Month", "i..pr")] = NULL
58
59 temp$Year = temp$X.Year
60 temp$tas = temp$i..tas
61 temp$month = temp$Month
62 temp[, c("X.Year", "i..tas", "Month")] = NULL
63
64 #replace commas with dots where necessary so we can convert to numeric
65 rain$pr = as.numeric(gsub(",", ".", gsub("\\.", "", rain$pr)))
66 temp$tas = as.numeric(gsub(",", ".", gsub("\\.", "", temp$tas)))
67 raintemp = merge(rain, temp[c("tas", "Year", "month")], by=c("month", "Year"))
68
69 # calling the function
70 raintemp = avgRainTempQuarter(raintemp, "month", "Year", "pr", "tas")
71
72 data = merge(data, unique(raintemp[c("tas_q1", "tas_q2", "tas_q3", "tas_q4", "pr_q1", "pr_
  q2", "pr_q3", "pr_q4", "Year")]), by=c("Year"))
73
74
75 #
  #####
76
77 ### read the oil prices data
78
79
80 colnames(oil_prices) = c("Year", "oil_avarage_price_per_barrel")
81 oil_prices$oil_avarage_price_per_barrel = as.numeric(gsub(",", ".", gsub("\\.", "",
  oil_prices$oil_avarage_price_per_barrel)))
82 data = merge(x = data, y = oil_prices, by= "Year", all.x = TRUE)

```

```

83
84 # #####
85
86 ### Population data
87
88 population = population[, c("Year", "Unit", "Value")]
89 colnames(population) = c("Year", "Population_Unit", "Population_Value")
90 data = merge(x = data, y = population, by= "Year", all.x = TRUE)
91
92 # #####
93
94 ### Production Amount
95
96 Production_amount = Production_amount[, c("Year", "Item", "Value")]
97 colnames(Production_amount)[3] = "Production_Amount"
98 data = merge(x = data, y = Production_amount, by= c("Year", "Item"), all.x = TRUE)
99
100
101 # #####
102
103 # GNI per capita, Atlas method (current US$)
104
105 GNI = GNI[GNI$Country.Name == "Rwanda",]
106 GNI[1:35] = NULL
107 GNI[c("X2017", "X")] = NULL
108 colnames(GNI) = c("1991", "1992", "1993", "1994", "1995", "1996", "1997", "1998", "
1999", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008", "
2009", "2010", "2011", "2012", "2013", "2014", "2015")
109 GNI = as.data.frame(t(GNI))
110 setDT(GNI, keep.rownames = TRUE)[]
111
112 colnames(GNI) = c("Year", "GNI")
113 data = merge(x = data, y = GNI, by= "Year", all.x = TRUE)
114
115 # #####
116
117 # Exchange rate
118
119 exchange_rate = exchange_rate[, c("Year", "Value")]
120 colnames(exchange_rate)[2] = "Exchange_Rate"
121 data = merge(x = data, y = exchange_rate, by= "Year", all.x = TRUE)
122
123 # #####
124
125 # GDP (current US$)
126
127 GDP = GDP[GDP$Country.Name == "Rwanda",]
128 GDP[1:35] = NULL
129 GDP[c("X2016", "X2017", "X")] = NULL
130 colnames(GDP) = c("1991", "1992", "1993", "1994", "1995", "1996", "1997", "1998", "
1999", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008", "
2009", "2010", "2011", "2012", "2013", "2014", "2015")
131 GDP = as.data.frame(t(GDP))
132 setDT(GDP, keep.rownames = TRUE)[]
133
134 colnames(GDP) = c("Year", "GDP")
135 data = merge(x = data, y = GDP, by= "Year", all.x = TRUE)
136
137 # #####

```

```

134
135 # Inflation, GDP deflator (annual %)
136 Inflation = Inflation[Inflation$Country.Name == "Rwanda",]
137 Inflation[1:35] = NULL
138 Inflation[c("X2016", "X2017", "X")] = NULL
139 colnames(Inflation) = c("1991", "1992", "1993", "1994", "1995", "1996", "1997", "1998",
    "1999", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007", "2008",
    "2009", "2010", "2011", "2012", "2013", "2014", "2015")
140 Inflation = as.data.frame(t(Inflation))
141 setDT(Inflation, keep.rownames = TRUE)[]
142
143 colnames(Inflation) = c("Year", "Inflation")
144 # the inflation data for the year 1994, 1995 where missing so we got them from
    another source: http://rwanda.opendataforafrica.org/rjirstd/cpi-by-country-
    statistics?country=Rwanda
145 Inflation$Inflation[Inflation$Year == "1994"] = 21.0
146 Inflation$Inflation[Inflation$Year == "1995"] = 56.0
147 data = merge(x = data, y = Inflation, by= "Year", all.x = TRUE)
148
149 #
    #####

150
151 # Agriculture GDP
152 # Agriculture, value added (constant 2010 US$)
153 Agriculture_GDP = Agriculture_GDP[Agriculture_GDP$Country.Name == "Rwanda",]
154 Agriculture_GDP[1:35] = NULL
155 Agriculture_GDP[c("X2016", "X2017", "X")] = NULL
156 colnames(Agriculture_GDP) = c("1991", "1992", "1993", "1994", "1995", "1996", "1997",
    "1998", "1999", "2000", "2001", "2002", "2003", "2004", "2005", "2006", "2007",
    "2008", "2009", "2010", "2011", "2012", "2013", "2014", "2015")
157 Agriculture_GDP = as.data.frame(t(Agriculture_GDP))
158 setDT(Agriculture_GDP, keep.rownames = TRUE)[]
159
160 colnames(Agriculture_GDP) = c("Year", "Agriculture_GDP")
161 data = merge(x = data, y = Agriculture_GDP, by= "Year", all.x = TRUE)
162
163 #
    #####

164
165 # import and export data for vegetables and Cereals
166 # thousand USD
167 colnames(Vegetables)[6] = "Year"
168 colnames(Cereals)[6] = "Year"
169 veg_import = Vegetables[Vegetables$flow == "Imports", c("Year", "Value")]
170 colnames(veg_import)[2] = "imp_veg"
171 veg_export = Vegetables[Vegetables$flow == "Exports", c("Year", "Value")]
172 colnames(veg_export)[2] = "exp_veg"
173 cer_import = Cereals[Cereals$flow == "Imports", c("Year", "Value")]
174 colnames(cer_import)[2] = "imp_cer"
175 cer_export = Cereals[Cereals$flow == "Exports", c("Year", "Value")]
176 colnames(cer_export)[2] = "exp_cer"
177
178 data = merge(x = data, y = veg_export, by= "Year", all.x = TRUE)
179 data = merge(x = data, y = cer_export, by= "Year", all.x = TRUE)
180 data = merge(x = data, y = veg_import, by= "Year", all.x = TRUE)
181 data = merge(x = data, y = cer_import, by= "Year", all.x = TRUE)
182
183 # now we have the import and export data is empty between 1991 - 1994, we will take
    the mean of each of the following five years
184
185 data$exp_veg[data$Year == "1991"] = mean(unique(data$exp_veg[data$Year %in% c("1995",
    "1996", "1997", "1998", "1999")]))
186 data$imp_veg[data$Year == "1991"] = mean(unique(data$imp_veg[data$Year %in% c("1995",

```

```

    "1996", "1997", "1998", "1999"))))
187 data$imp_cer[data$Year == "1991"] = mean(unique(data$imp_cer[data$Year %in% c("1995",
    "1996", "1997", "1998", "1999"))))
188 data$exp_cer[data$Year == "1991"] = mean(unique(data$exp_cer[data$Year %in% c("1995",
    "1996", "1997", "1998", "1999"))))
189
190 data$exp_veg[data$Year == "1992"] = mean(unique(data$exp_veg[data$Year %in% c("1996",
    "1997", "1998", "1999", "2000"))))
191 data$imp_veg[data$Year == "1992"] = mean(unique(data$imp_veg[data$Year %in% c("1996",
    "1997", "1998", "1999", "2000"))))
192 data$imp_cer[data$Year == "1992"] = mean(unique(data$imp_cer[data$Year %in% c("1996",
    "1997", "1998", "1999", "2000"))))
193 data$exp_cer[data$Year == "1992"] = mean(unique(data$exp_cer[data$Year %in% c("1996",
    "1997", "1998", "1999", "2000"))))
194
195 data$exp_veg[data$Year == "1993"] = mean(unique(data$exp_veg[data$Year %in% c("1997",
    "1998", "1999", "2000", "2001"))))
196 data$imp_veg[data$Year == "1993"] = mean(unique(data$imp_veg[data$Year %in% c("1997",
    "1998", "1999", "2000", "2001"))))
197 data$imp_cer[data$Year == "1993"] = mean(unique(data$imp_cer[data$Year %in% c("1997",
    "1998", "1999", "2000", "2001"))))
198 data$exp_cer[data$Year == "1993"] = mean(unique(data$exp_cer[data$Year %in% c("1997",
    "1998", "1999", "2000", "2001"))))
199
200 data$exp_veg[data$Year == "1994"] = mean(unique(data$exp_veg[data$Year %in% c("1998",
    "1999", "2000", "2001", "2002"))))
201 data$imp_veg[data$Year == "1994"] = mean(unique(data$imp_veg[data$Year %in% c("1998",
    "1999", "2000", "2001", "2002"))))
202 data$imp_cer[data$Year == "1994"] = mean(unique(data$imp_cer[data$Year %in% c("1998",
    "1999", "2000", "2001", "2002"))))
203 data$exp_cer[data$Year == "1994"] = mean(unique(data$exp_cer[data$Year %in% c("1998",
    "1999", "2000", "2001", "2002"))))
204
205 #
    #####

206
207 # Average daily per capita caloric supply, measured in kilocalories per person per
    day.
208 dpccs = dpccs[dpccs$Entity == "Rwanda", c("Year","X.kcal.person.day.")]
209 colnames(dpccs)[2] = "daily_caloric_supply"
210 data = merge(x = data, y = dpccs, by= "Year", all.x = TRUE)
211
212 # 2014 and 2015 there is no data, replace with median
213 data$daily_caloric_supply[data$Year == "2014"] = mean(unique(data$daily_caloric_
    supply[data$Year %in% c("2012", "2011", "2010", "2009", "2008")]))
214 data$daily_caloric_supply[data$Year == "2015"] = mean(unique(data$daily_caloric_
    supply[data$Year %in% c("2013", "2012", "2011", "2010", "2009")]))
215
216
217 #
    #####

218
219 # matching the data with the other datas
220 colnames(data) = c("year", "prod_name", "unit","prod_price", "flag", "flag.
    description", "tas_q1", "tas_q2", "tas_q3", "tas_q4", "pr_q1", "pr_q2", "pr_q3",
    "pr_q4", "avg_p_barrel","population_unit", "population", "prod_amount", "gni_pc",
    "exchange_rate", "gdp", "cp_inflation", "agri_gdp", "exp_veg", "exp_cer", "imp_
    veg", "imp_cer", "daily_caloric_supply")
221 data$prod_price = as.numeric(data$prod_price)
222 # saving the data
223 saveRDS(data, ".\\Processed_ds\\rwanda_fin.rds")

1 | if(!require("plotmo")) install.packages("plotmo"); library("plotmo")
2 | if(!require("Hmisc")) install.packages("Hmisc"); library("Hmisc")

```

```

3 if(!require("corrplot")) install.packages("corrplot"); library("corrplot")
4 if(!require("caret")) install.packages("caret"); library("caret")
5 if(!require("ggfortify")) install.packages("ggfortify"); library("ggfortify")
6
7 source("../Helper_functions\\exploration_functions.r")
8
9
10 #results for VIF
11 india_insig_vif = readRDS("../Results\\insign_in.rds")
12 india_nohc_vif = readRDS("../Results\\mod_varnohc_in.rds")
13 india_lo_vif = readRDS("../Results\\mod_lovif_in.rds")
14
15 rwanda_insig_vif = readRDS("../Results\\insign_rw.rds")
16 rwanda_nohc_vif = readRDS("../Results\\mod_varnohc_rw.rds")
17 rwanda_lo_vif = readRDS("../Results\\mod_lovif_rw.rds")
18
19 philippines_insig_vif = readRDS("../Results\\insign_ph.rds")
20 philippines_nohc_vif = readRDS("../Results\\mod_varnohc_ph.rds")
21 philippines_lo_vif = readRDS("../Results\\mod_lovif_ph.rds")
22
23
24 ##corrplot for all variables
25
26 jpeg("../Results//Rs_plots//india_corr.jpg", width = 1120, height = 1000, units = "px",
27       pointsize = 20,
28       quality = 100)
29 corrplot(india_insig_vif, type = "lower", order = "hclust",
30          tl.col = "black", tl.srt = 45)
31 title("Pairwise Correlations India")
32 dev.off()
33 highcorrcolnames_in = colnames(india_insig_vif)[findCorrelation(india_insig_vif,
34                        cutoff = 0.70)]
35 sink("../Results\\Rs_data\\hcvars_in.txt")
36 print(highcorrcolnames_in)
37 sink()
38
39 jpeg("../Results//Rs_plots//rwanda_corr.jpg", width = 1120, height = 1000, units = "px",
40       pointsize = 20,
41       quality = 100)
42 corrplot(rwanda_insig_vif, type = "lower", order = "hclust",
43          tl.col = "black", tl.srt = 45)
44 title("Pairwise Correlations Rwanda")
45 dev.off()
46 highcorrcolnames_rw = colnames(rwanda_insig_vif)[findCorrelation(rwanda_insig_vif,
47                        cutoff = 0.70)]
48 sink("../Results\\Rs_data\\hcvars_rw.txt")
49 print(highcorrcolnames_rw)
50 sink()
51
52 jpeg("../Results//Rs_plots//philippines_corr.jpg", width = 1120, height = 1000, units = "px",
53       pointsize = 20,
54       quality = 100)
55 corrplot(philippines_insig_vif, type = "lower", order = "hclust",
56          tl.col = "black", tl.srt = 45)
57 title("Pairwise Correlations Philippines")
58 dev.off()
59 highcorrcolnames_ph = colnames(philippines_insig_vif)[findCorrelation(philippines_insig_vif,
60                        cutoff = 0.70)]
61 sink("../Results\\Rs_data\\hcvars_ph.txt")
62 print(highcorrcolnames_ph)
63 sink()
64
65 ##significant variables left after removal of all variables part of a pair with

```

```

        correlation > 0.7
63
64 sink("...\\Results\\Rs_data\\nohc_mod_in.txt")
65 print(summary(india_nohc_vif))
66 sink()
67
68
69 sink("...\\Results\\Rs_data\\nohc_mod_rw.txt")
70 print(summary(rwanda_nohc_vif))
71 sink()
72
73 sink("...\\Results\\Rs_data\\nohc_mod_ph.txt")
74 print(summary(philippines_nohc_vif))
75 sink()
76
77
78 ##significant variables left after removal of multicorrelated variables by removeVif
   ()
79 sink("...\\Results\\Rs_data\\vif_mod_in.txt")
80 print(summary(india_lo_vif))
81 sink()
82
83
84 sink("...\\Results\\Rs_data\\vif_mod_rw.txt")
85 print(summary(rwanda_lo_vif))
86 sink()
87
88 sink("...\\Results\\Rs_data\\vif_mod_ph.txt")
89 print(summary(philippines_lo_vif))
90 sink()
91
92
93
94 #results for lasso
95 india_lasso_result = readRDS("...\\Results\\india_lasso.rds")
96 rwanda_lasso_result = readRDS("...\\Results\\rwanda_lasso.rds")
97 philippines_lasso_result = readRDS("...\\Results\\philippines_lasso.rds")
98
99
100
101 #India
102 #Optimal number of variables -> where MSE is minimal
103 jpeg("...\\Results\\Rs_plots\\india_lasso_mse.jpg", width = 600, height = 600, units =
      "px", pointsize = 20,
104       quality = 100)
105 autoplot(india_lasso_result[[2]], main = "Optimal number of variables by MSE for
      India")
106 dev.off()
107 #At what Lambda do variables enter the model
108 #jpeg("...\\Results\\Rs_plots\\india_lasso_lambda.jpg", width = 1200, height = 1400,
      units = "px", pointsize = 20,
109       quality = 100)
110 #plot_glmnet(india_lasso_result[[1]], s=india_lasso_result[[2]]$lambda.min, col =india_
      lasso_result[[3]]$colors, label = TRUE )
111 #dev.off()
112 sink("...\\Results\\Rs_data\\lasso_vars_in.txt")
113 print(india_lasso_result[[3]])
114 sink()
115
116
117 #Rwanda
118 #Optimal number of variables -> where MSE is minimal
119 jpeg("...\\Results\\Rs_plots\\rwanda_lasso_mse.jpg", width = 600, height = 600, units =
      "px", pointsize = 20,
120       quality = 100)
121 autoplot(rwanda_lasso_result[[2]], main = "Optimal number of variables by MSE for

```



```

    Rwanda")
122 dev.off()
123 #At what Lambda do variables enter the model
124 #jpeg("../Results//Rs_plots//rwanda_lasso_lambda.jpg", width = 1200, height = 1200,
    units = "px", pointsize = 20,
125     quality = 100)
126 #plot_glmnet(rwanda_lasso_result[[1]],s=rwanda_lasso_result[[2]]$lambda.min,col =
    rwanda_lasso_result[[3]]$colors, label = TRUE )
127 #dev.off()
128 sink("../Results\\Rs_data\\lasso_vars_rw.txt")
129 print(rwanda_lasso_result[[3]])
130 sink()
131
132
133 #Philippines
134 #Optimal number of variables -> where MSE is minimal
135 jpeg("../Results//Rs_plots//philippines_lasso_mse.jpg", width = 600, height = 600,
    units = "px", pointsize = 20,
136     quality = 100)
137 autoplot(philippines_lasso_result[[2]], main = "Optimal number of variables by MSE
    for Philippines")
138 dev.off()
139 #At what Lambda do variables enter the model
140 #jpeg("../Results//Rs_plots//philippines_lasso_lambda.jpg", width = 1200, height =
    1200, units = "px", pointsize = 20,
141     quality = 100)
142 #plot_glmnet(philippines_lasso_result[[1]],s=philippines_lasso_result[[2]]$lambda.min
    ,col =philippines_lasso_result[[3]]$colors, label = TRUE )
143 #dev.off()
144 sink("../Results\\Rs_data\\lasso_vars_ph.txt")
145 print(philippines_lasso_result[[3]])
146 sink()
147
148
149 #results for random forest based variable selection
150 india_rf_result = readRDS("../Results\\india_rf.rds")
151 rwanda_rf_result = readRDS("../Results\\rwanda_rf.rds")
152 philippines_rf_result = readRDS("../Results\\philippines_rf.rds")
153
154 #threshold step india
155 sink("../Results\\Rs_data\\rf_thres_in.txt")
156 print(india_rf_result[[2]][india_rf_result[[1]]$varselect.thres])
157 sink()
158
159 jpeg("../Results//Rs_plots//india_rf_thres.jpg", width = 1000, height = 700, units =
    "px", pointsize = 20,
160     quality = 100)
161 plotVsurf(india_rf_result[[1]], "thres", "India")
162 dev.off()
163
164 #interpretation step india
165 sink("../Results\\Rs_data\\rf_interp_in.txt")
166 print(india_rf_result[[2]][india_rf_result[[1]]$varselect.interp])
167 sink()
168
169 jpeg("../Results//Rs_plots//india_rf_interp.jpg", width = 1000, height = 700, units =
    "px", pointsize = 20,
170     quality = 100)
171 plotVsurf(india_rf_result[[1]], "interp", "India")
172 dev.off()
173
174
175 #threshold step rwanda
176 sink("../Results\\Rs_data\\rf_thres_rw.txt")
177 print(rwanda_rf_result[[2]][rwanda_rf_result[[1]]$varselect.thres])
178 sink()

```



```

179
180 jpeg("../Results//Rs_plots//rwanda_rf_thres.jpg", width = 1000, height = 700, units =
    "px", pointsize = 20,
181     quality = 100)
182 plotVsurf(rwanda_rf_result[[1]], "thres", "Rwanda")
183 dev.off()
184
185 #interpretation step rwanda
186 sink("../Results\\Rs_data\\rf_interp_rw.txt")
187 print(rwanda_rf_result[[2]][rwanda_rf_result[[1]]$vselect.interp])
188 sink()
189
190 jpeg("../Results//Rs_plots//rwanda_rf_interp.jpg", width = 1000, height = 700, units
    = "px", pointsize = 20,
191     quality = 100)
192 plotVsurf(rwanda_rf_result[[1]], "interp", "Rwanda")
193 dev.off()
194
195
196 #threshold step philippines
197 sink("../Results\\Rs_data\\rf_thres_ph.txt")
198 print(philippines_rf_result[[2]][philippines_rf_result[[1]]$vselect.thres])
199 sink()
200
201 jpeg("../Results//Rs_plots//philippines_rf_thres.jpg", width = 1000, height = 700,
    units = "px", pointsize = 20,
202     quality = 100)
203 plotVsurf(philippines_rf_result[[1]], "thres", "Philippines")
204 dev.off()
205
206 #interpretation step philippines
207 sink("../Results\\Rs_data\\rf_interp_ph.txt")
208 print(philippines_rf_result[[2]][philippines_rf_result[[1]]$vselect.interp])
209 sink()
210
211 jpeg("../Results//Rs_plots//philippines_rf_interp.jpg", width = 1000, height = 700,
    units = "px", pointsize = 20,
212     quality = 100)
213 plotVsurf(philippines_rf_result[[1]], "interp", "Philippines")
214 dev.off()
215
216
217 #cleanup
218 rm(list = setdiff(ls(), lsf.str()))

1 source("../Helper_functions\\variable_selection_functions.R")
2
3 rwanda = readRDS("../Processed_ds\\rwanda_fin.rds")
4 india = readRDS("../Processed_ds\\india_fin.rds")
5 philippines = readRDS("../Processed_ds\\philippines_fin.rds")
6 #select explanatory variables
7 exp_var_rw = c("pr_q1", "pr_q2", "pr_q3", "pr_q4", "tas_q1", "tas_q2", "tas_q3", "
    tas_q4",
8             "prod_amount", "daily_caloric_supply", "exp_veg", "exp_
    cer", "imp_veg", "imp_cer",
9             "agri_gdp", "gni_pc", "cp_inflation", "avg_p_barrel", "population"
    , "prod_price")
10 rwanda = rwanda[, colnames(rwanda) %in% exp_var_rw ]
11
12 exp_var_in = c("prod_price", "pr_q1", "pr_q2", "pr_q3", "pr_q4", "tas_q1", "tas_q2", "tas_q3
    ", "tas_q4",
13             "prod_amount", "daily_caloric_supply", "exp_sug", "exp_veg", "exp_cer", "
    imp_sug", "imp_veg", "imp_cer",
14             "agri_gdp", "gni_pc", "cp_inflation", "avg_p_barrel", "population")
15 india = india[, colnames(india) %in% exp_var_in ]
16

```

```

17 exp_var_ph = c("prod_price", "tas_q1", "tas_q2", "tas_q3", "tas_q4", "pr_q1", "pr_q2", "pr_
    q3",
18                 "pr_q4", "avg_p_barrel", "population", "prod_amount", "gni_pc", "
    exchange_rate",
19                 "gdp", "cp_inflation", "agri_gdp", "imp_cer", "exp_agri", "daily_
    caloric_supply")
20 philippines = philippines[, colnames(philippines) %in% exp_var_ph ]
21
22 #Get most important variables with Lasso method
23 rwanda_lasso_result = impVarsLasso(rwanda, "prod_price")
24 india_lasso_result = impVarsLasso(india, "prod_price")
25 philippines_lasso_result = impVarsLasso(philippines, "prod_price")
26
27 #save results
28 savestring = paste0(deparse(substitute(rwanda)), "_lasso.rds")
29 saveRDS(rwanda_lasso_result, (paste0(".\\Results\\", savestring)))
30
31 savestring = paste0(deparse(substitute(india)), "_lasso.rds")
32 saveRDS(india_lasso_result, (paste0(".\\Results\\", savestring)))
33
34 savestring = paste0(deparse(substitute(philippines)), "_lasso.rds")
35 saveRDS(philippines_lasso_result, (paste0(".\\Results\\", savestring)))
36
37
38 rm(list = setdiff(ls(), lsf.str()))

1 source(".\\Helper_functions\\variable_selection_functions.R")
2
3 india = readRDS(".\\Processed_ds\\india_fin.rds")
4 rwanda = readRDS(".\\Processed_ds\\rwanda_fin.rds")
5 philippines = readRDS(".\\Processed_ds\\philippines_fin.rds")
6
7
8
9 #initial variable selection and normalization
10 colselection_in = c("prod_price", "pr_q1", "pr_q2", "pr_q3", "pr_q4", "tas_q1", "tas_q2", "
    tas_q3", "tas_q4",
11                    "prod_amount", "daily_caloric_supply", "exp_sug", "exp_veg", "exp_cer
    ", "imp_sug", "imp_veg", "imp_cer",
12                    "agri_gdp", "gni_pc", "cp_inflation", "avg_p_barrel", "population")
13 india = as.data.frame(scale(india[colselection_in]))
14
15 colselection_rw = c("pr_q1", "pr_q2", "pr_q3", "pr_q4", "tas_q1", "tas_q2", "tas_q3"
    , "tas_q4",
16                    "prod_amount", "daily_caloric_supply", "exp_veg", "exp_cer", "
    imp_veg", "imp_cer",
17                    "agri_gdp", "gni_pc", "cp_inflation", "avg_p_barrel", "population
    ", "prod_price")
18 rwanda = as.data.frame(scale(rwanda[colselection_rw]))
19
20 colselection_ph = c("prod_price", "tas_q1", "tas_q2", "tas_q3", "tas_q4", "pr_q1", "pr_q2"
    , "pr_q3",
21                    "pr_q4", "avg_p_barrel", "population", "prod_amount", "gni_pc", "
    exchange_rate",
22                    "gdp", "cp_inflation", "agri_gdp", "imp_cer", "exp_agri", "daily_
    caloric_supply")
23 philippines = as.data.frame(scale(philippines[colselection_ph]))
24
25 #run the model
26 india_v_imp_rf = impVarsRf(india, "prod_price")
27 rwanda_v_imp_rf = impVarsRf(rwanda, "prod_price")
28 philippines_v_imp_rf = impVarsRf(philippines, "prod_price")
29
30
31 #save results
32 savestring = paste0(deparse(substitute(india)), "_rf.rds")

```

```

33 saveRDS(india_v_imp_rf, (paste0(".\\Results\\", savestring)))
34
35 savestring = paste0(deparse(substitute(rwanda)), "_rf.rds")
36 saveRDS(rwanda_v_imp_rf, (paste0(".\\Results\\", savestring)))
37
38 savestring = paste0(deparse(substitute(philippines)), "_rf.rds")
39 saveRDS(philippines_v_imp_rf, (paste0(".\\Results\\", savestring)))
40
41 #cleanup
42 rm(list = setdiff(ls(), lsf.str()))

1  if(!require("Hmisc")) install.packages("Hmisc"); library("Hmisc")
2  if(!require("corrplot")) install.packages("corrplot"); library("corrplot")
3  if(!require("caret")) install.packages("caret"); library("caret")
4
5  source(".\\Helper_functions\\variable_selection_functions.R")
6
7  india = readRDS(".\\Processed_ds\\india_fin.rds")
8  rwanda = readRDS(".\\Processed_ds\\rwanda_fin.rds")
9  philippines = readRDS(".\\Processed_ds\\philippines_fin.rds")
10
11 #initial variable selection and normalization
12 colselection_in = c("prod_price", "pr_q1", "pr_q2", "pr_q3", "pr_q4", "tas_q1", "tas_q2", "
    tas_q3", "tas_q4",
13                     "prod_amount", "daily_caloric_supply", "exp_sug", "exp_veg", "exp_cer", "
    imp_sug", "imp_veg", "imp_cer",
14                     "agri_gdp", "gni_pc", "cp_inflation", "avg_p_barrel", "population")
15 target_in = c("prod_price")
16 normalized_in = as.data.frame(scale(india[colselection_in]))
17 feats_in = normalized_in[, !(colnames(normalized_in) %in% target_in)]
18
19 colselection_rw = c("pr_q1", "pr_q2", "pr_q3", "pr_q4", "tas_q1", "tas_q2", "tas_q3"
    , "tas_q4",
20                     "prod_amount", "daily_caloric_supply", "exp_veg", "exp_cer", "
    imp_veg", "imp_cer",
21                     "agri_gdp", "gni_pc", "cp_inflation", "avg_p_barrel", "population"
    , "prod_price")
22 target_rw = c("prod_price")
23 normalized_rw = as.data.frame(scale(rwanda[colselection_rw]))
24 feats_rw = normalized_rw[, !(colnames(normalized_rw) %in% target_rw)]
25
26 colselection_ph = c("prod_price", "tas_q1", "tas_q2", "tas_q3", "tas_q4", "pr_q1", "pr_q2"
    , "pr_q3",
27                     "pr_q4", "avg_p_barrel", "population", "prod_amount", "gni_pc", "
    exchange_rate",
28                     "gdp", "cp_inflation", "agri_gdp", "imp_cer", "exp_agri", "daily_
    caloric_supply")
29 target_ph = c("prod_price")
30 normalized_ph = as.data.frame(scale(philippines[colselection_ph]))
31 feats_ph = normalized_ph[, !(colnames(normalized_ph) %in% target_ph)]
32
33 #Variable selection and modeling
34
35 #Model with all explanatory variables
36 insign_in = cor( feats_in, method = "pearson", use = "complete.obs")
37 insign_rw = cor( feats_rw, method = "pearson", use = "complete.obs")
38 insign_ph = cor( feats_ph, method = "pearson", use = "complete.obs")
39
40
41
42 # Discovering highly correlated explanatory variables
43 hicorvars_in = findCorrelation(cor(feats_in), cutoff = 0.70)
44 expvarsnohc_in = paste(colnames(feats_in[,-hicorvars_in]), collapse = "+")
45 formulanohc_in = paste(target_in, "~", expvarsnohc_in, collapse = "+")
46 mod_varnohc_in = lm(formulanohc_in, data = normalized_in)
47

```

```

48 | hicorvars_rw = findCorrelation(cor(feats_rw), cutoff = 0.70)
49 | expvarsnohc_rw = paste(colnames(feats_rw[, -hicorvars_rw]), collapse = "+")
50 | formulanohc_rw = paste(target_rw, "~", expvarsnohc_rw, collapse = "+")
51 | mod_varnohc_rw = lm(formulanohc_rw, data = normalized_rw)
52 |
53 | hicorvars_ph = findCorrelation(cor(feats_ph), cutoff = 0.70)
54 | expvarsnohc_ph = paste(colnames(feats_ph[, -hicorvars_ph]), collapse = "+")
55 | formulanohc_ph = paste(target_ph, "~", expvarsnohc_ph, collapse = "+")
56 | mod_varnohc_ph = lm(formulanohc_ph, data = normalized_ph)
57 |
58 |
59 | #Multicollinearity removal
60 | # for highly correlated variables
61 | varslovifhc_in = removeVif(feats_in[, hicorvars_in], 8)
62 | varslovifhc_rw = removeVif(feats_rw[, hicorvars_rw], 8)
63 | varslovifhc_ph = removeVif(feats_ph[, hicorvars_ph], 8)
64 | # the rest
65 | varslovifnohc_in = removeVif(feats_in[, -hicorvars_in], 8)
66 | varslovifnohc_rw = removeVif(feats_rw[, -hicorvars_rw], 8)
67 | varslovifnohc_ph = removeVif(feats_ph[, -hicorvars_ph], 8)
68 | #Model without multicollinearity
69 | expvars_lovif_in = paste(paste(varslovifhc_in$variable, collapse = "+"), "+", paste(
70 |   varslovifnohc_in$variable, collapse = "+"), collapse = "+")
71 | formula_lovif_in = paste(target_in, "~", expvars_lovif_in, collapse = "+")
72 | mod_lovif_in = lm(formula_lovif_in, data = normalized_in)
73 |
74 | expvars_lovif_rw = paste(paste(varslovifhc_rw$variable, collapse = "+"), "+", paste(
75 |   varslovifnohc_rw$variable, collapse = "+"), collapse = "+")
76 | formula_lovif_rw = paste(target_rw, "~", expvars_lovif_rw, collapse = "+")
77 | mod_lovif_rw = lm(formula_lovif_rw, data = normalized_rw)
78 |
79 | expvars_lovif_ph = paste(paste(varslovifhc_ph$variable, collapse = "+"), "+", paste(
80 |   varslovifnohc_ph$variable, collapse = "+"), collapse = "+")
81 | formula_lovif_ph = paste(target_ph, "~", expvars_lovif_ph, collapse = "+")
82 | mod_lovif_ph = lm(formula_lovif_ph, data = normalized_ph)
83 | #save results
84 | savesuffix = deparse(substitute(insign_in))
85 | savestring = paste0(savesuffix, ".rds")
86 | saveRDS(insign_in, (paste0(".\\Results\\", savestring)))
87 |
88 | savesuffix = deparse(substitute(mod_varnohc_in))
89 | savestring = paste0(savesuffix, ".rds")
90 | saveRDS(mod_varnohc_in, (paste0(".\\Results\\", savestring)))
91 |
92 | savesuffix = deparse(substitute(mod_lovif_in))
93 | savestring = paste0(savesuffix, ".rds")
94 | saveRDS(mod_lovif_in, (paste0(".\\Results\\", savestring)))
95 |
96 | savesuffix = deparse(substitute(insign_rw))
97 | savestring = paste0(savesuffix, ".rds")
98 | saveRDS(insign_rw, (paste0(".\\Results\\", savestring)))
99 |
100 | savesuffix = deparse(substitute(mod_varnohc_rw))
101 | savestring = paste0(savesuffix, ".rds")
102 | saveRDS(mod_varnohc_rw, (paste0(".\\Results\\", savestring)))
103 |
104 | savesuffix = deparse(substitute(mod_lovif_rw))
105 | savestring = paste0(savesuffix, ".rds")
106 | saveRDS(mod_lovif_rw, (paste0(".\\Results\\", savestring)))
107 |
108 | savesuffix = deparse(substitute(insign_ph))
109 | savestring = paste0(savesuffix, ".rds")
110 | saveRDS(insign_ph, (paste0(".\\Results\\", savestring)))

```

```

111 saveRDS(mod_varnohc_ph, (paste0(".\\Results\\", savestring)))
112
113 savesuffix = deparse(substitute(mod_lovif_ph))
114 savestring = paste0(savesuffix, ".rds")
115 saveRDS(mod_lovif_ph, (paste0(".\\Results\\", savestring)))
116
117
118 #cleanup
119 rm(list = setdiff(ls(), lsf.str()))

1 #loading the food prices for India
2 india = readRDS(".\\Qfolder1\\Q1_india_prices.rds")
3
4 #calculate average price per food per month for the whole country
5 #Define the function for food price per month across all markets
6 avgPriceFoodMonth = function(ds, cm_name, mp_price, mp_year, mp_month){
7   dsfoods = unique(ds[[cm_name]])
8   ds$avg_price_prod_month = 0
9   for (k in min(ds[[mp_year]]):max(ds[[mp_year]])){
10     print(paste('year is ', k))
11     for (j in 1:NROW(dsfoods)){
12       a <- ds[ds[[mp_year]] == k & ds[[cm_name]] == dsfoods[j],]
13       print(paste('food is ', dsfoods[j]))
14       for (i in 1:12){
15         b <- a[a[[mp_month]] == i,]
16         if(NROW(ds[ds[[mp_year]] == k
17           & ds[[cm_name]] == dsfoods[j]
18           & ds[[mp_month]] == i,]$avg_price_prod_month) > 0) {
19           ds[ds[[mp_year]] == k
20             & ds[[cm_name]] == dsfoods[j]
21             & ds[[mp_month]] == i,]$avg_price_prod_month
22             = sum(b[[mp_price]])/ NROW(b)
23           print(paste('month is ', i))
24         }
25       }
26     }
27   }
28 }
29 return(ds)
30 }
31
32
33 india = avgPriceFoodMonth(india, "cm_name", "price", "year", "month")
34
35 #calculate average price per food per year
36 #Define function for average price per year
37 avgPriceFoodYear = function(ds, cm_name, mp_year, avg_price_prod_month){
38   dsfoods = unique(ds[[cm_name]])
39   ds$avg_price_prod_year = 0
40   for (x in min(ds[[mp_year]]):max(ds[[mp_year]])){
41     print(paste('year is ', x))
42     for(y in 1:NROW(dsfoods)){
43       print(paste('food is ', dsfoods[y]))
44       if(NROW(ds[ds[[mp_year]] == x & ds[[cm_name]] == dsfoods[y],]$avg_price_prod_
45         year) > 0){
46         ds[ds[[mp_year]] == x & ds[[cm_name]] == dsfoods[y],]$avg_price_prod_year =
47           sum(ds[ds[[mp_year]] == x & ds[[cm_name]] == dsfoods[y],][[avg_price_prod_
48             _month]]) / NROW(ds[ds[[mp_year]] == x & ds[[cm_name]] == dsfoods[y],)[[
49             avg_price_prod_month]])
46       }
47     }
48   }
49 }
50 return(ds)
51 }
52

```

```

53
54 india = avgPriceFoodYear(india,"cm_name","year","avg_price_prod_month")
55
56 # we've decided to base our analysis on years, so we delete month related columns and
    other columns we don'd need
57 india$month = NULL
58 india$avg_price_prod_month = NULL
59 india$adm1_name = NULL
60 india$mkt_name = NULL
61 india$price = NULL
62 india = unique(india)
63
64 #renaming columns due to convention
65 colnames(india)[colnames(india) %in% "avg_price_prod_year"] = "prod_price"
66 colnames(india)[colnames(india) %in% "adm0_name" ] = "country"
67 colnames(india)[colnames(india) %in% "cm_name"] = "prod_name"
68 colnames(india)[colnames(india) %in% "um_id" ] = "prod_uid"
69 colnames(india)[colnames(india) %in% "um_name" ] = "prod_unit"
70
71
72 #save our dataset for later
73 saveRDS(india, ("\\.\\Qfolder1\\Q1_india_wip.rds"))
74
75 #cleanup
76 rm(list = setdiff(ls(), lsf.str()))

1 #prepare rain and temp data: mean amount of rain and mean temperature for each
    quarter year
2 rain = readRDS("\\Qfolder2\\Q2_india_rain.rds")
3 temp = readRDS("\\Qfolder2\\Q2_india_temp.rds")
4
5 rain$ISO3 = NULL
6 rain$ISO2 = NULL
7 rain$year = rain$X.Year
8 rain$X.Year = NULL
9 rain$month = rain$Month
10 rain$Month = NULL
11
12 temp$year = temp$X.Year
13 temp$X.Year = NULL
14 temp$month = temp$Month
15 temp$Month = NULL
16
17 raintemp = merge(rain,temp[c("tas","year","month")],by=c("month","year"))
18
19 # Calculate average rain and temp per quarter
20 # Define function for average rain and temp per quarter
21 avgRainTempQuarter = function(ds,month,mp_year,pr,tas){
22   if (is.na(ds[[pr]]) || is.na(ds[[tas]])) {
23     message(paste("No missing values allowed!"))
24   } else {
25     ds$tas_q1 = 0
26     ds$tas_q2 = 0
27     ds$tas_q3 = 0
28     ds$tas_q4 = 0
29     ds$pr_q1 = 0
30     ds$pr_q2 = 0
31     ds$pr_q3 = 0
32     ds$pr_q4 = 0
33
34     for(z in min(ds[[mp_year]]):max(ds[[mp_year]])){
35
36       ds[ds[[mp_year]] == z ,]$pr_q1 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in%
        c("1","2","3"),)[[pr]])/3
37       ds[ds[[mp_year]] == z ,]$pr_q2 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in%
        c("4","5","6"),)[[pr]])/3

```

```

38     ds[ds[[mp_year]] == z ,]$pr_q3 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in
    % c("7","8","9"),)[[pr]])/3
39     ds[ds[[mp_year]] == z ,]$pr_q4 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in
    % c("10","11","12"),)[[pr]])/3
40     ds[ds[[mp_year]] == z ,]$tas_q1 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in
    % c("1","2","3"),)[[tas]])/3
41     ds[ds[[mp_year]] == z ,]$tas_q2 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in
    % c("4","5","6"),)[[tas]])/3
42     ds[ds[[mp_year]] == z ,]$tas_q3 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in
    % c("7","8","9"),)[[tas]])/3
43     ds[ds[[mp_year]] == z ,]$tas_q4 = sum(ds[ds[[mp_year]] == z & ds[[month]] %in
    % c("10","11","12"),)[[tas]])/3
44 }
45 return(ds)
46 }
47 }
48 raintemp = avgRainTempQuarter(raintemp,"month","year","pr","tas")
49
50 #load the india dataset and the remaining variables
51 india_wip = readRDS(".\\Qfolder1\\Q1_india_wip.rds")
52 rest = readRDS(".\\Qfolder2\\Q2_india_rest.rds")
53
54
55 #merge india with wheater data
56 india_wip = merge(india_wip,unique(raintemp[c("tas_q1","tas_q2","tas_q3","tas_q4","pr
    _q1","pr_q2","pr_q3","pr_q4","year")]),by=c("year"))
57
58 #join the datasets
59 india_fin = merge(india_wip, rest, by=c("prod_price")) #prod_price is unique,
60                                                         #therefore it can be used as
61                                                         a key for the merge
62
63 #save dataset
64 saveRDS(india_fin, (".\\Qfolder2\\Q2_india_fin.rds"))
65
66 #cleanup
67 rm(list = setdiff(ls(), lsf.str()))

1 if(!require("fmsb")) install.packages("fmsb"); library("fmsb")
2 if(!require("Hmisc")) install.packages("Hmisc"); library("Hmisc")
3 if(!require("caret")) install.packages("caret"); library("caret")
4
5
6 #loading the data
7 india = readRDS(".\\Qfolder2\\Q2_india_fin.rds")
8
9 #initial variable selection and normalization
10 colselection_in = c("prod_price","pr_q1","pr_q2","pr_q3","pr_q4","tas_q1","tas_q2","
    tas_q3","tas_q4",
11                    "prod_amount","daily_caloric_supply","exp_sug","exp_veg","exp_cer
    ","imp_sug","imp_veg","imp_cer",
12                    "agri_gdp","gni_pc","cp_inflation","avg_p_barrel","population")
13 target_in = c("prod_price")
14 normalized_in = as.data.frame(scale(india[colselection_in]))
15 feats_in = normalized_in[, !(colnames(normalized_in) %in% target_in)]
16
17
18 #Variable selection and modeling
19
20 #Model with all explanatory variables
21 insign_in = cor( feats_in, method = "pearson", use = "complete.obs")
22
23
24
25

```

```

26 # Discovering highly correlated explanatory variables
27 hicorvars_in = findCorrelation(cor(feats_in), cutoff = 0.70)
28 expvarsnohc_in = paste(colnames(feats_in)[-hicorvars_in], collapse = "+")
29 formulanohc_in = paste(target_in, "~", expvarsnohc_in, collapse = "+")
30 mod_varnohc_in = lm(formulanohc_in, data = normalized_in)
31
32
33
34 #Multicollinearity removal
35 #function for VIF based stepwise removal of multicorrelated variables
36 removeVif = function(explan_vars, cutoffval=10){
37
38     tempresults = as.data.frame(matrix(ncol = 2, nrow = 0))
39     colnames(tempresults) = c("variable", "vif")
40     #initially calculate VIF for each explanatory variable
41     for (i in 1:NROW(colnames(explan_vars)) ){
42         temptarget = colnames(explan_vars)[i]
43         tempexpvars = paste(colnames(explan_vars[,!(colnames(explan_vars) %in% temptarget
44             )]), collapse = "+")
45         tempformula = paste(temptarget, "~", tempexpvars, collapse = " ")
46
47         tempresults[i,1] = temptarget
48         tempresults[i,2] = VIF(lm( tempformula, data = explan_vars))
49     }
50     print(tempresults[order(tempresults$vif),])
51     #remove variable with highest VIF, calculate new VIF for remaining variables until
52     #all VIF are below cutoff value
53     while(max(tempresults$vif) >= cutoffval){
54         tempresults = tempresults[!tempresults$vif == max(tempresults$vif),]
55         tempremvars = tempresults$variable
56         for(j in 1: NROW(tempremvars)){
57             temptarget = tempremvars[j]
58             tempexpvars = paste(tempremvars[!tempremvars %in% temptarget], collapse = "+")
59             tempformula = paste(temptarget, "~", tempexpvars, collapse = " ")
60
61             tempresults[j,1] = temptarget
62             tempresults[j,2] = VIF(lm( tempformula, data = explan_vars))
63         }
64
65         print("Remaining variables:")
66         print(tempresults[order(tempresults$vif),])
67         cat("\n")
68     }
69     return(tempresults)
70 }
71
72 # for highly correlated variables
73 varslovifhc_in = removeVif(feats_in[,hicorvars_in],8)
74 # for lower correlated variables
75 varslovifnohc_in = removeVif(feats_in[, -hicorvars_in],8)
76 #Model without multicollinearity
77 expvars_lovif_in = paste(paste(varslovifhc_in$variable, collapse = "+"), "+", paste(
78     varslovifnohc_in$variable, collapse = "+"), collapse = "+")
79 formula_lovif_in = paste(target_in, "~", expvars_lovif_in, collapse = "+")
80 mod_lovif_in = lm(formula_lovif_in, data = normalized_in)
81
82 #save results
83 savesuffix = deparse(substitute(insign_in))
84 savestring = paste0(".\Qfolder3\\", "Q3_", savesuffix, ".rds")
85 saveRDS(insign_in, savestring)
86
87 savesuffix = deparse(substitute(mod_varnohc_in))
88 savestring = paste0(".\Qfolder3\\", "Q3_", savesuffix, ".rds")

```



```

89 saveRDS(mod_varnohc_in, savestring)
90
91 savesuffix = deparse(substitute(mod_lovif_in))
92 savestring = paste0(".*\\Qfolder3\\", "Q3_", savesuffix, ".rds")
93 saveRDS(mod_lovif_in, savestring)
94
95
96 #cleanup
97 rm(list = setdiff(ls(), lsf.str()))

1  if(!require("glmnet")) install.packages("glmnet"); library("glmnet")
2  if(!require("plyr")) install.packages("plyr"); library("plyr")
3
4  #load the data
5  india = readRDS(".*\\Qfolder2\\Q2_india_fin.rds")
6
7  #select explanatory variables
8  exp_var_in = c("prod_price", "pr_q1", "pr_q2", "pr_q3", "pr_q4", "tas_q1", "tas_q2", "tas_q3",
9               "tas_q4",
10               "prod_amount", "daily_caloric_supply", "exp_sug", "exp_veg", "exp_cer", "
               imp_sug", "imp_veg", "imp_cer",
11               "agri_gdp", "gni_pc", "cp_inflation", "avg_p_barrel", "population")
12  india = india[, colnames(india) %in% exp_var_in ]
13
14  #function for LASSO method
15  impVarsLasso = function(ds, targ){
16      #1. initial variable selection and normalization
17
18      val = ds[[targ]]
19      x = model.matrix(ds[[targ]]~.-1 , ds[!colnames(ds) %in% targ])
20
21      #2. Applying the Lasso technique
22      lasso = glmnet(x = x, y = val, standardize = TRUE, alpha = 1)
23
24      fit = cv.glmnet(x = x, y = val, standardize = TRUE, type.measure = "mse", alpha=1,
25                    nfolds=3)
26
27      #3. Results
28      #with lambda.min
29      lambda_min = which(fit$lambda == fit$lambda.min)
30
31      #selecting coefficients of variables at lambda where mse is minimal
32      tempmincoefs = as.data.frame(fit$glmnet.fit$beta[, which(fit$lambda ==
33                    fit$lambda.min)])
34      mincoefs = data.frame(matrix(ncol = 2, nrow = (NROW(tempmincoefs))
35                    ))
36      mincoefs$variables = as.vector(as.character(labels(tempmincoefs)[[1]]))
37      mincoefs$coefs_minlambda = as.vector(tempmincoefs[[1]])
38      mincoefs$X1 = NULL
39      mincoefs$X2 = NULL
40
41      #get names in the decreasing order they appear in when lambda is minimal
42      names = names(coef(lasso)[,ncol(coef(lasso))[order(coef(lasso)[,ncol(
43                    coef(lasso))], decreasing=TRUE)])
44      names = names[!names %in% c("(Intercept)")]
45      names = as.data.frame(names)
46      colnames(names) = "variables"
47
48      #add coefficient to names
49      disp_colors = join(names, mincoefs, by = "variables" )
50      disp_colors = disp_colors[!disp_colors$variables %in% c("(Intercept)"),]
51
52      #set colors for variables when displayed in a graph
53      disp_colors$colors = 0
54      if(NROW(disp_colors[disp_colors$coefs_minlambda >0,])>0){

```

```

51     disp_colors[disp_colors$coefs_minlambda >0,]$colors = c("green")
52 }
53 if(NROW(disp_colors[disp_colors$coefs_minlambda <0,])>0){
54     disp_colors[disp_colors$coefs_minlambda <0,]$colors = c("red")
55 }
56
57
58 #create a list to store the result
59 resultset = vector("list",3)
60 resultset[[1]] = lasso
61 resultset[[2]] = fit
62 resultset[[3]] = disp_colors
63
64 return(resultset)
65 }
66
67
68 #Get most important variables with Lasso function
69 india_lasso_result = impVarsLasso(india,"prod_price")
70
71 #save results
72 savestring = paste0(deparse(substitute(india)),"_lasso.rds")
73 saveRDS(india_lasso_result, (paste0(".\\Qfolder4\\","Q4_",savestring)))
74
75 #cleanup
76 rm(list = setdiff(ls(), lsf.str()))

1  if(!require("VSURF")) install.packages("VSURF"); library("VSURF")
2  #load the data
3  india = readRDS(".\\Processed_ds\\india_fin.rds")
4
5  #initial variable selection and normalization
6  colselection_in = c("prod_price","pr_q1","pr_q2","pr_q3","pr_q4","tas_q1","tas_q2","
7      tas_q3","tas_q4",
8      "prod_amount","daily_caloric_supply","exp_sug","exp_veg","exp_cer",
9      "imp_sug","imp_veg","imp_cer",
10     "agri_gdp","gni_pc","cp_inflation","avg_p_barrel","population")
11 india = as.data.frame(scale(india[colselection_in]))
12
13 #function for finding most important variables based on random forest
14 impVarsRf = function(ds,targ){
15     result_rf = VSURF(ds[[targ]] ~ ., data = ds[!colnames(ds) %in% targ], ntree = 2000,
16         nfor.thres = 50, nmin = 1, nfor.interp = 25, nsd = 1,
17         nfor.pred = 25, nmj = 1, parallel = FALSE, ncores = detectCores()
18         - 1,
19         clusterType = "PSOCK")
20     #create a list to store the result
21     resultset = vector("list",2)
22     resultset[[1]] = result_rf
23     resultset[[2]] = colnames(ds[!colnames(ds) %in% targ])
24     return(resultset)
25 }
26
27 #apply the function
28 india_v_imp_rf = impVarsRf(india,"prod_price")
29
30 #save results
31 savestring = paste0(deparse(substitute(india)),"_rf.rds")
32 saveRDS(india_v_imp_rf, (paste0(".\\Qfolder5\\","Q5_",savestring)))
33
34 #cleanup
35 rm(list = setdiff(ls(), lsf.str()))

1  if(!require("plotmo")) install.packages("plotmo"); library("plotmo")

```

```

2  if(!require("Hmisc")) install.packages("Hmisc"); library("Hmisc")
3  if(!require("corrplot")) install.packages("corrplot"); library("corrplot")
4  if(!require("caret")) install.packages("caret"); library("caret")
5  if(!require("ggfortify")) install.packages("ggfortify"); library("ggfortify")
6
7
8
9
10
11
12  #loading results for correlation / VIF based variable selection
13  india_insig_vif = readRDS(".\\Qfolder3\\Q3_insign_in.rds")
14  india_nohc_vif = readRDS(".\\Qfolder3\\Q3_mod_varnohc_in.rds")
15  india_lo_vif = readRDS(".\\Qfolder3\\Q3_mod_lovif_in.rds")
16
17  #create corrplots for all variables and save plot and variable names to file
18  jpeg("../Qfolder6/Q6_india_corr.jpg", width = 1120, height = 1000, units = "px",
19        pointsize = 20,
20        quality = 100)
21  corrplot(india_insig_vif, type = "lower", order = "hclust",
22           tl.col = "black", tl.srt = 45)
23  title("Pairwise Correlations India")
24  dev.off()
25  highcorrcolnames_in = colnames(india_insig_vif)[findCorrelation(india_insig_vif,
26                           cutoff = 0.70)]
27  sink("../Qfolder6/Q6_hcvars_in.txt")
28  print(highcorrcolnames_in)
29  sink()
30
31  #significant variables left after removal of all variables part of a pair with
32  #correlation >0.7
33  #save results to file
34  sink("../Qfolder6/Q6_nohc_mod_in.txt")
35  print(summary(india_nohc_vif))
36  sink()
37
38  #significant variables left after removal of multicorrelated variables by removeVif()
39  #save results to file
40  sink("../Qfolder6/Q6_vif_mod_in.txt")
41  print(summary(india_lo_vif))
42  sink()
43
44
45  #load results for lasso
46  india_lasso_result = readRDS(".\\Qfolder4\\Q4_india_lasso.rds")
47
48  #Optimal number of variables -> where MSE is minimal
49  #save plot and variable names to file
50  jpeg("../Qfolder6/Q6_india_lasso_mse.jpg", width = 600, height = 600, units = "px",
51        pointsize = 20,
52        quality = 100)
53  autoplot(india_lasso_result[[2]], main = "Optimal number of variables by MSE for
54           India")
55  dev.off()
56  #At what Lambda do variables enter the model
57  #jpeg("../Qfolder6/Q6_india_lasso_lambda.jpg", width = 1200, height = 1400, units =
58  #      "px", pointsize = 20,
59  #      quality = 100)
60  #plot_glmnet(india_lasso_result[[1]], s=india_lasso_result[[2]]$lambda.min, col = india_
61  #lasso_result[[3]]$colors, label = TRUE)
62  #dev.off()
63  sink("../Qfolder6/Q6_lasso_vars_in.txt")
64  print(india_lasso_result[[3]])

```

```

61 sink()
62
63
64
65
66 #loading results for random forest based variable selection
67 india_rf_result = readRDS("..\Qfolder5\Q5_india_rf.rds")
68
69 #function for plotting VSURF Objects
70 plotVsurf = function(iVsurfOb,iStep,iCountry){
71   header_prefix = "not specified"
72   if(iStep == "thres"){
73     header_prefix = "Thresholding step"
74   }
75   if(iStep == "interp"){
76     header_prefix = "Interpretation step"
77   }
78
79   plot(iVsurfOb,step = iStep, var.names = FALSE,
80        nvar.interp = length(iVsurfOb$varselect.thres), main = paste(header_prefix,
81                                iCountry))
82
83   #threshold step
84   #save variables and plot to file
85   sink("..\Qfolder6\Q6_rf_thres_in.txt")
86   print(india_rf_result[[2]][india_rf_result[[1]]$varselect.thres])
87   sink()
88
89   jpeg("../Qfolder6//Q6_india_rf_thres.jpg", width = 1000, height = 700, units = "px",
90        pointsize = 20,
91        quality = 100)
92   plotVsurf(india_rf_result[[1]],"thres","India")
93   dev.off()
94
95   #interpretation step
96   sink("..\Qfolder6\Q6_rf_interp_in.txt")
97   print(india_rf_result[[2]][india_rf_result[[1]]$varselect.interp])
98   sink()
99
100  jpeg("../Qfolder6//Q6_india_rf_interp.jpg", width = 1000, height = 700, units = "px",
101       pointsize = 20,
102       quality = 100)
103  plotVsurf(india_rf_result[[1]],"interp","India")
104  dev.off()
105
106
107 #cleanup
108 rm(list = setdiff(ls(), lsf.str()))

```



```

1 #reading the data
2 world_population = read.csv("../Qfolder7\world_population.csv", stringsAsFactors =
3   FALSE, sep = ",", header = TRUE)
4 world_production = read.csv("../Qfolder7\world_production.csv", stringsAsFactors =
5   FALSE, sep = ",", header = TRUE)
6 price_index = read.csv("../Qfolder7\Food_price_indices_data.csv", stringsAsFactors =
7   FALSE, sep = ",")
8
9
10 # libraries
11 if(!require("reshape2")) install.packages("reshape2");library("reshape2")

```

```

12 if(!require("ggplot2")) install.packages("ggplot2");library("ggplot2")
13 if(!require("data.table")) install.packages("data.table");library("data.table")
14 if(!require("zoo")) install.packages("zoo");library("zoo")
15
16
17
18
19 #calculating the percentage of the change in a column's value on a fixed base year
20 calcPercFixBaseyear = function(ds, areacol, areaname, yearcol, baseyear, valuecol,
    perccol){
21   base = ds[ds[[yearcol]] == baseyear & ds[[areacol]] %in% areaname, ][[valuecol]]
22   if(is.null(ds[[perccol]])){
23     ds[[perccol]] = 0
24   }
25   #
26   for(i in baseyear: max(ds[[yearcol]])){
27     later = ds[ds[[yearcol]]==i & ds[[areacol]] %in% areaname, ][[valuecol]]
28     sub = later - base
29     ds[ds[[yearcol]] == i & ds[[areacol]] %in% areaname, ][[perccol]] = (sub / later)
        * 100
30   }
31   return(ds)
32 }
33
34
35 # calculate the prcentage of changes in a colname value in a predefined year, where
the base is for every change is the value from the previous year.
36 calcPercPreBaseyear = function(ds, areacol, areaname, yearcol, valuecol){
37   for(i in unique(ds[[yearcol]])){
38     base = ds[ds[[yearcol]] == i & ds[[areacol]] %in% areaname, ][[valuecol]]
39     later = ds[ds[[yearcol]] == i+1 & ds[[areacol]] %in% areaname, ][[valuecol]]
40     #if(length(later) == 0L) break
41     if(i == 2015L) break
42     sub = later - base
43     if(length(sub) == 0L)next
44     ds[ds[[yearcol]] == i+1 & ds[[areacol]] %in% areaname , paste(valuecol, "Percent"
        , sep = "_")]= (sub / later) * 100
45   }
46   ds[ds[[yearcol]] == min(ds[[yearcol]]) & ds[[areacol]] %in% areaname, paste(
        valuecol, "Percent", sep = "_")]= 0
47   return(ds)
48 }
49
50 # To plot production data with specific area and itmes
51 prodPlot = function(ds, area, items){
52   p = ggplot(data=ds[ds$Area == area & ds$Item %in% items,], aes(x=Year, y=Percentage
        , colour=Item)) +
53     geom_line() +
54     geom_point()+
55     ylim(-30, 75)+
56     ggtitle(label=area)+
57     ylab(label="Percentage Production Change") +
58     xlab("Year")
59   return(p)
60 }
61
62 # bar plot for product price change
63
64 prodBarPlot = function(d, dname){
65   ggplot(d[d$year %in% c(2010:2015) ,c("year", "prod_name", "prod_price_Percent")],
        aes(x = year, y = prod_price_Percent)) +
66     geom_bar(aes(fill = prod_name), position = "dodge", stat="identity") +
67     ggtitle(label=dname)+
68     ylab(label="price change based on previous year") +
69     xlab("Year")
70 }

```

```

71
72 # Multiple plot function
73 #
74 # ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)
75 # - cols: Number of columns in layout
76 # - layout: A matrix specifying the layout. If present, 'cols' is ignored.
77 #
78 # If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),
79 # then plot 1 will go in the upper left, 2 will go in the upper right, and
80 # 3 will go all the way across the bottom.
81 #
82 multiplot = function(..., plotlist=NULL, file, cols=1, layout=NULL) {
83   require(grid)
84
85   # Make a list from the ... arguments and plotlist
86   plots = c(list(...), plotlist)
87
88   numPlots = length(plots)
89
90   # If layout is NULL, then use 'cols' to determine layout
91   if (is.null(layout)) {
92     # Make the panel
93     # ncol: Number of columns of plots
94     # nrow: Number of rows needed, calculated from # of cols
95     layout = matrix(seq(1, cols * ceiling(numPlots/cols)),
96                     ncol = cols, nrow = ceiling(numPlots/cols))
97   }
98
99   if (numPlots==1) {
100     print(plots[[1]])
101
102   } else {
103     # Set up the page
104     grid.newpage()
105     pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))
106
107     # Make each plot, in the correct location
108     for (i in 1:numPlots) {
109       # Get the i,j matrix positions of the regions that contain this subplot
110       matchidx = as.data.frame(which(layout == i, arr.ind = TRUE))
111
112       print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row,
113                                       layout.pos.col = matchidx$col))
114     }
115   }
116 }
117
118
119 # 1. plotting the population for the all countries and the world
120
121 # stacks a set of columns into a single column of data to be able to process it
122 world_population = melt(world_population, id=c("Year"), value.name = "population")
123
124 # calculating the percentage of the change in the population
125 for(i in unique(world_population$variable)){
126   # i is the name of the land
127   #print(i)
128   world_population = calcPercFixBaseyear(world_population, "variable", i, "Year", 1991, "
    population", "percentage")
129 }
130
131 # creating and saving the plot
132 jpeg("../Qfolder7//population_plot.jpg", width = 800, height = 480, units = "px",
    pointsize = 12,
133     quality = 75)
134 ggplot(world_population) + geom_line(aes(x=Year, y=percentage, colour=variable), size

```

```

    =1.2) +
135   scale_colour_manual(values=c("red","green","blue", "grey")) +
136   ylab(label="Growth Precentage") +
137   xlab("Year")
138   dev.off()
139
140   #
141   #####
142
143   # 2. plotting the production amount of the selected products for the specified
144   # countries compared to the world
145
146   # preparing the dataset
147   world_production$X = NULL
148   colnames(world_production) = c("Area", "Item", "1991", "1992", "1993", "1994", "1995",
149   "1996", "1997", "1998", "1999", "2000", "2001", "2002", "2003", "2004", "2005",
150   "2006", "2007", "2008", "2009", "2010", "2011", "2012", "2013", "2014", "2015")
151
152   world_production = melt(world_production, id=c("Area","Item"), value.name = "
153   Production_Amount")
154   # select the intersting items
155   world_production = world_production[world_production$Item %in% c("Sugar cane", "Rice,
156   paddy", "Wheat", "Potatoes",
157   "Bananas", "Coconuts",
158   "Cassava", "Beans,
159   dry", "Maize", "
160   Sweet potatoes")
161   ,]
162   # remove some unintersting itmes specified by a land
163   world_production = world_production[!(world_production$Area == "India" & world_
164   production$Item %in% c("Cassava", "Bananas", "Beans, dry", "Maize", "Sweet
165   potatoes", "Coconuts")),]
166
167   world_production = world_production[!(world_production$Area == "Philippines" & world_
168   production$Item %in% c("Cassava", "Wheat", "Beans, dry", "Maize", "Sweet potatoes",
169   "Potatoes")),]
170
171   world_production = world_production[!(world_production$Area == "Rwanda" & world_
172   production$Item %in% c("Wheat", "Sugar cane", "Coconuts")),]
173   colnames(world_production)[3] = "Year"
174   world_production$Year = as.numeric(levels(world_production$Year))[world_production$
175   Year]
176   # normalize the production amount
177   #world_production$Production_Amount = scale(world_production$Production_Amount)
178   h = data.frame()
179   for(i in unique(world_production$Item)){
180     d = world_production[world_production$Item == i,]
181     for(j in unique(d$Area)){
182       # i is the name of the land
183       #print(i)
184       d = calcPercFixBaseyear(d,"Area",j,"Year",1991,"Production_Amount", "Percentage")
185     }
186     h = rbind(h,d)
187   }
188   world_production = h
189
190   # calling the plot function and get the plot
191   p1 = prodPlot(world_production, "India", c("Sugar cane", "Rice, paddy", "Wheat", "
192   Potatoes"))
193   p2 = prodPlot(world_production, "World", c("Sugar cane", "Rice, paddy", "Wheat", "
194   Potatoes"))
195   p3 = prodPlot(world_production, "Philippines", c("Sugar cane", "Bananas", "Coconuts",

```

```

    "Rice, paddy"))
181 p4 = prodPlot(world_production, "World", c("Sugar cane", "Bananas", "Coconuts", "Rice
    , paddy"))
182 # p5 = prodPlot(world_production, "Rwanda", c("Cassava", "Bananas", "Beans, dry", "
    Maize", "Sweet potatoes", "Potatoes", "Rice, paddy"))
183 # p6 = prodPlot(world_production, "World", c("Cassava", "Bananas", "Beans, dry", "
    Maize", "Sweet potatoes", "Potatoes", "Rice, paddy"))
184 p5 = prodPlot(world_production, "Rwanda", c("Cassava", "Bananas", "Maize", "Sweet
    potatoes"))
185 p6 = prodPlot(world_production, "World", c("Cassava", "Bananas", "Maize", "Sweet
    potatoes"))
186
187 # plot in one screen and save the image
188 jpeg("../Qfolder7//production.jpg", width = 1200, height = 800, units = "px",
    pointsize = 12,
189     quality = 75)
190 multiplot(p1, p3, p5,p2, p4,p6, cols=2)
191 dev.off()
192
193 #
    #####

194
195 # 3. Plotting the price index
196 price_index[,8:16] = NULL
197 price_index$Date = as.yearmon(price_index$Date, format = "%m/%Y")
198 price_index = price_index[!price_index$Date %in% c(1990,2016,2017,2018),]
199
200 # creating and saving the plot
201 jpeg("../Qfolder7//price_index.jpg", width = 800, height = 480, units = "px",
    pointsize = 12,
202     quality = 75)
203 ggplot(price_index, aes(x = Date)) +
204   geom_line(aes(y = Food.Price.Index, colour="Food")) +
205   geom_line(aes(y = Cereals.Price.Index, colour="Cereals")) +
206   geom_line(aes(y = Oils.Price.Index, colour="Oils")) +
207   geom_line(aes(y = Sugar.Price.Index, colour="Sugar")) +
208   scale_x_yearmon(format="%m/%Y", n=5)+
209   ylab(label="Price Index") +
210   xlab("Year")
211 dev.off()
212
213 #
    #####

214 # 4. Plotting a bar chart for each item form 2010 - 2015
215
216 for(i in unique(rdata$prod_name)){
217   rdata = calcPercPreBaseyear(rdata, "prod_name", i, "year", "prod_price")
218 }
219 for(i in unique(pdata$prod_name)){
220   pdata = calcPercPreBaseyear(pdata, "prod_name", i, "year", "prod_price")
221 }
222 for(i in unique(idata$prod_name)){
223   idata = calcPercPreBaseyear(idata, "prod_name", i, "year", "prod_price")
224 }
225 idata[idata$year == 2012 & idata$prod_name == "Potatoes", "prod_price_Percent"] = 0
226 b1 = prodBarPlot(rdata, "Rwanda")
227 b2 = prodBarPlot(pdata, "Philippines")
228 b3 = prodBarPlot(idata, "India")
229
230 jpeg("../Qfolder7//barplot_price_change.jpg", width = 1200, height = 800, units = "px
    ", pointsize = 12,
231     quality = 75)
232 multiplot(b1,b2,b3, cols=1)
233 dev.off()

```



```
234 |  
235 |  
236 | #cleanup  
237 | rm(list = setdiff(ls(), lsf.str()))
```

Declaration of Authorship

We hereby confirm that we have authored this Seminar paper independently and without use of others than the indicated sources. All passages which are literally or in general matter taken out of publications or other sources are marked as such.

Berlin, 30.03.2018, Benjamin Jaidi, Stefan Ramakrishnan, Raiber Alkurdi