Data Mining & Predictive Modelling Assignment

Pablo Benayas Penas

25/02/2020

- These are the different sections I am going to cover in this assignment:
  - Firstly, I am going to work with data from the Spanish National Elections. I will clean the dataset and then, will create linear and logistic regression models to predict varaibles thereof.
  - Secondly, Time Series Analysis will be conducted with US_monthly_unemployment_rate data from 2007-08 to present moment.
  - Finally, it will carry out clustering analysis with the aforementioned Spanish National Elections Dataset.

PART I To begin with, let's clean the dataset:

Columns are renamed

```
options(warn=-1)

elec=readxl::read_excel('C:/Users/pablo/Desktop/DatosEleccionesEsp.xlsx')
elec=as.data.frame(elec)

names(elec)=c('Name', 'ProvinceCode', 'Autonomous_Community', 'Population', 'TotalCensus',
              'AbstentionPtge', 'High_Abstention_rate', 'Left_wing_Pct', 'Right_wing_Pct',
              'Others_Pct', 'Left_wing', 'Right_wing', 'Age_0_4_Ptge', 'Age_under19_Ptge',
              'Age_19_65_pct', 'Age_over65_pct', 'FemalePopulationPtge', 'ForeignersPtge',
              'SameAutomComPtge', 'SameAutonComDiffProvPtge', 'DifAutonComPtge',
              'UnemployLess25_Ptge', 'Unemploy25_40_Ptge', 'UnemployMore40_Ptge',
              'AgricultureUnemploymentPtge', 'IndustryUnemploymentPtge',
              'ConstructionUnemploymentPtge', 'ServicesUnemploymentPtge', 'totalCompanies',
              'Industry', 'ConstructionInd', 'commerceNhostelry', 'ServiceInd',
              'MainActivity', 'RealProperty', 'Pob2010', 'SurfaceArea', 'Density',
              'PopChange_pct','People_RealProp_ratio', 'officesNfacilities')
```

If variable has few unique values, it should be converted to factor

```
sapply(elec, function(x) length(unique(x)))
```

```
##                          Name                  ProvinceCode
##                          8102                            52
##          Autonomous_Community                    Population
##                            19                          3597
##                   TotalCensus               AbstentionPtge
##                          3310                          5675
##          High_Abstention_rate                 Left_wing_Pct
##                             2                          6569
##                Right_wing_Pct                    Others_Pct
##                          6682                          4319
##                     Left_wing                    Right_wing
##                             2                             2
##                  Age_0_4_Ptge              Age_under19_Ptge
##                          3761                          5891
##                 Age_19_65_pct               Age_over65_pct
##                          6215                          6778
##          FemalePopulationPtge               ForeignersPtge
##                          4524                          2329
##              SameAutomComPtge      SameAutonComDiffProvPtge
##                          6151                          4207
##               DifAutonComPtge             UnemployLess25_Ptge
##                          5574                          2342
##            Unemploy25_40_Ptge             UnemployMore40_Ptge
##                          2681                          2751
##   AgricultureUnemploymentPtge      IndustryUnemploymentPtge
##                          2525                          2538
## ConstructionUnemploymentPtge      ServicesUnemploymentPtge
##                          2505                          2904
##                totalCompanies                      Industry
##                          1226                           308
##               ConstructionInd             commerceNhostelry
##                           457                           803
##                    ServiceInd                  MainActivity
##                           758                             5
##                  RealProperty                       Pob2010
##                          3088                          3625
##                   SurfaceArea                       Density
##                          8110                             4
##                 PopChange_pct         People_RealProp_ratio
##                          3049                           283
##            officesNfacilities
##                           758
```

I set less than 10 unique values as benchmark for factor conversion

```
to_factor=elec[,sapply(elec, function(x) length(unique(x))) < 10]
elec[,sapply(elec, function(x) length(unique(x))) < 10] = lapply(to_factor, factor)

head(elec[,sapply(elec, function(x) length(unique(x))) < 10],2)
```

```
##   High_Abstention_rate Left_wing Right_wing MainActivity Density
## 1                    0         1          0         Otro MuyBaja
## 2                    0         1          0         Otro MuyBaja
```

for the sake of file length, dfplot() is not going to be displayed

```r
# function provided by my teacher
#dfplot <- function(data.frame){
#  df <- data.frame
#  ln <- length(names(data.frame))
#  for(i in 1:ln){
#    if(is.factor(df[,i])){
#      plot(df[,i],main=names(df)[i])} # 'main' argument: an overall title for the plot

#    # names(df)[i] returns its column name
#    else{hist(df[,i],main=names(df)[i])
#   boxplot(df[,i],main=names(df)[i])}
#  }
#}

#dfplot(elec)
```

Let's check the structure of the dataset

```r
str(elec)
```

```
## 'data.frame':    8119 obs. of  41 variables:
##  $ Name                        : chr  "Abadía" "Abertura" "Acebo" "Acehúche" ...
##  $ ProvinceCode                : num  10 10 10 10 10 10 10 10 10 10 ...
##  $ Autonomous_Community        : chr  "Extremadura" "Extremadura" "Extremadura" "Extremadura" ...
##  $ Population                  : num  336 429 569 822 623 ...
##  $ TotalCensus                 : num  282 364 569 704 540 ...
##  $ AbstentionPtge              : num  20.2 25.3 27.2 30.1 30.2 ...
##  $ High_Abstention_rate        : Factor w/ 2 levels "0","1": 1 1 1 2 2 1 2 1 1 1 ...
##  $ Left_wing_Pct               : num  60.4 54.8 44.2 50.8 44.6 ...
##  $ Right_wing_Pct              : num  35.6 44.1 53.1 45.3 49.9 ...
##  $ Others_Pct                  : num  1.778 0.368 0.966 0 0.796 ...
##  $ Left_wing                   : Factor w/ 2 levels "0","1": 2 2 1 2 1 2 2 1 1 1 ...
##  $ Right_wing                  : Factor w/ 2 levels "0","1": 1 1 2 1 2 1 1 2 2 2 ...
##  $ Age_0_4_Ptge                : num  3.87 1.63 1.23 4.26 3.53 ...
##  $ Age_under19_Ptge            : num  18.16 13.05 9.14 14.96 15.57 ...
##  $ Age_19_65_pct               : num  55.1 56.6 54.8 60.1 59.4 ...
##  $ Age_over65_pct              : num  26.8 30.3 36 24.9 25 ...
##  $ FemalePopulationPtge        : num  44 50.1 49 51.1 48.2 ...
##  $ ForeignersPtge              : num  0.89 1.63 0.7 0.12 0.64 0.56 0.98 3.56 2.04 1.95 ...
##  $ SameAutomComPtge            : num  79.8 90.9 78.9 93.9 93.3 ...
##  $ SameAutonComDiffProvPtge    : num  0.298 2.797 0.703 0.487 0.161 ...
##  $ DifAutonComPtge             : num  19.34 7.23 18.1 5.11 4.17 ...
##  $ UnemployLess25_Ptge         : num  2.38 16.22 8.2 7.41 15.38 ...
##  $ Unemploy25_40_Ptge          : num  54.8 32.4 36.1 61.1 48.1 ...
##  $ UnemployMore40_Ptge         : num  42.9 51.4 55.7 31.5 36.5 ...
##  $ AgricultureUnemploymentPtge : num  4.76 8.11 22.95 16.67 21.15 ...
##  $ IndustryUnemploymentPtge    : num  9.52 8.11 9.84 5.56 0 ...
##  $ ConstructionUnemploymentPtge: num  11.9 10.8 13.1 16.7 11.5 ...
##  $ ServicesUnemploymentPtge    : num  73.8 67.6 49.2 59.3 61.5 ...
##  $ totalCompanies              : num  15 11 49 50 22 90 45 26 82 7 ...
##  $ Industry                    : num  0 0 0 0 0 5 0 0 9 0 ...
##  $ ConstructionInd             : num  0 0 0 0 0 18 0 0 14 0 ...
##  $ commerceNhostelry           : num  0 0 0 0 0 56 0 0 32 0 ...
##  $ ServiceInd                  : num  0 0 0 0 0 11 0 0 27 0 ...
##  $ MainActivity                : Factor w/ 5 levels "ComercTTEHosteleria",..: 4 4 4 4 4 1 4 4 1 4 ...
##  $ RealProperty                : num  216 382 918 599 394 ...
##  $ Pob2010                     : num  326 459 674 842 625 ...
##  $ SurfaceArea                 : num  4508 6271 5702 9106 4008 ...
##  $ Density                     : Factor w/ 4 levels "?","Alta","Baja",..: 4 4 4 4 4 4 4 1 4 4 ...
##  $ PopChange_pct               : num  3.07 -6.54 -15.58 -2.38 -0.32 ...
##  $ People_RealProp_ratio       : num  1.56 1.12 0.62 1.37 1.58 1.39 2.18 0.83 1.26 0.78 ...
##  $ officesNfacilities          : num  28 67 74 66 96 ...
```

Let's check number of NAs per varaible

```r
sapply(elec, function(x) sum(is.na(x)))
```

```
##                          Name                 ProvinceCode
##                             0                            0
##           Autonomous_Community                   Population
##                             0                            0
##                   TotalCensus              AbstentionPtge
##                             0                            0
##          High_Abstention_rate                Left_wing_Pct
##                             0                            0
##                Right_wing_Pct                   Others_Pct
##                             0                            0
##                     Left_wing                   Right_wing
##                             0                            0
##                  Age_0_4_Ptge             Age_under19_Ptge
##                             0                            0
##                 Age_19_65_pct               Age_over65_pct
##                             0                            0
##          FemalePopulationPtge               ForeignersPtge
##                             0                            0
##               SameAutomComPtge     SameAutonComDiffProvPtge
##                             0                            0
##                DifAutonComPtge          UnemployLess25_Ptge
##                             0                            0
##              Unemploy25_40_Ptge          UnemployMore40_Ptge
##                             0                            0
##   AgricultureUnemploymentPtge     IndustryUnemploymentPtge
##                             0                            0
## ConstructionUnemploymentPtge     ServicesUnemploymentPtge
##                             0                            0
##                totalCompanies                     Industry
##                             5                          188
##               ConstructionInd            commerceNhostelry
##                           139                            9
##                    ServiceInd                 MainActivity
##                            62                            0
##                  RealProperty                      Pob2010
##                           138                            7
##                   SurfaceArea                      Density
##                             9                            0
##                 PopChange_pct        People_RealProp_ratio
##                             7                          138
##             officesNfacilities
##                             0
```

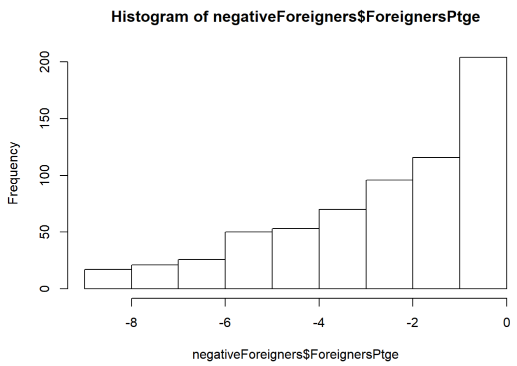
- OBSERVATIONS:
    - 1. Check if 100 % is a valid max value for ‘Others_Pct’ (it actualy is)
    - 2. ‘ForeignersPtge’ has negative values. Decide whether those values should be converted to positive or NA.
    - 3. Check if 127 % is a valid max value for ‘SameAutomComPtge’
    - 4. ‘Density’ has ‘?’ values. Convert them to NAs
    - 5. Check if 138.46 is a valid max value for ‘SameAutomComPtge’
    - 6. ‘Offices&Facilities’: max value is ‘99999’, while median is 52
    - 7. Explain the 100 0 0 unemployment thing

```
# LET'S BEGIN
# 1) Check if 100 % is a valid max value for 'Others_Pct' (it actualy is)
# Apart from national-oriented political parties, there are regional parties too.
# These political parties are particularly popular in Catolonia and the Basque Country.
# Thus, if max value of 'Other_Pct' is not a locality from the aforementioned Autonomous
# Communities, I will treat it as NA

elec[elec$Others_Pct==max(elec$Others_Pct),c('Name','ProvinceCode',
                                             'Autonomous_Community','Others_Pct')]
```

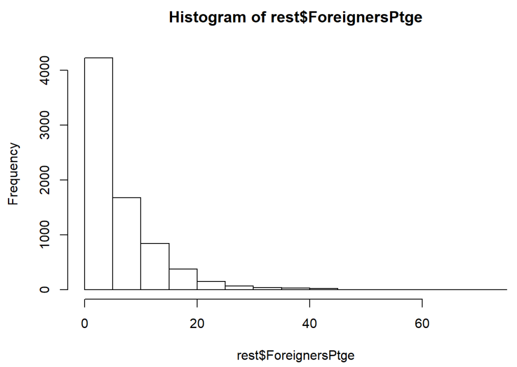
```
##                          Name ProvinceCode Autonomous_Community Others_Pct
## 7699 Santa Maria de Merlès            8             Cataluña        100
```

```
# As it is located in Catalonia, I assume this value is valid
```

```
# 2) 'ForeignersPtge' has negative values. Decide whether those values should be converted to
# positive or NA.
negativeForeigners=elec[elec$ForeignersPtge<0,c('Name',
                             'ProvinceCode','Autonomous_Community','ForeignersPtge')]

rest=elec[elec$ForeignersPtge>=0,c('Name',
                             'ProvinceCode','Autonomous_Community','ForeignersPtge')]

hist(negativeForeigners$ForeignersPtge)
```




```
hist(rest$ForeignersPtge)
```




```
# I have decided to convert those values to positive since the distribution of both 'negative'
# and 'rest' are similar enough
elec$ForeignersPtge=ifelse(elec$ForeignersPtge<0, -1*elec$ForeignersPtge, elec$ForeignersPtge)
```

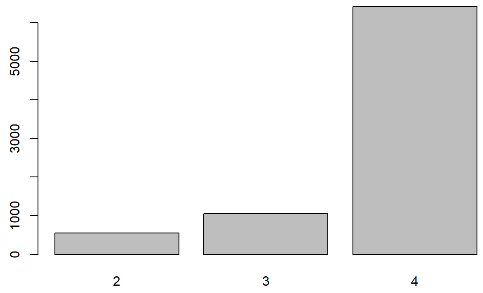
```
# 3) Check if 127 % is a valid max value for 'SameAutomComPtge'

elec[elec$SameAutomComPtge>100,c('Name','Population',
                             'ProvinceCode','Autonomous_Community','SameAutomComPtge')]
```

```
##                  Name Population ProvinceCode Autonomous_Community
## 1148 Arenas del Rey       1241           18            Andalucía
## 1220       Iznalloz       5158           18            Andalucía
## 6883          Berja      12370            4            Andalucía
##      SameAutomComPtge
## 1148          127.156
## 1220          106.805
## 6883          102.401
```

```
# I assume percentages cannot be greater than 100% for 'SameAutomComPtge'.
# Values greater than 100 are converted to NA
elec$SameAutomComPtge=ifelse(elec$SameAutomComPtge>100, NA, elec$SameAutomComPtge)
summary(elec$SameAutomComPtge)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   75.80   84.49   81.62   90.46  100.00       3
```

```
# 4) 'Density' has '?' values. Convert them to NAs

elec$Density=as.factor(ifelse(elec$Density=='?', NA, elec$Density))
plot(elec$Density)
```

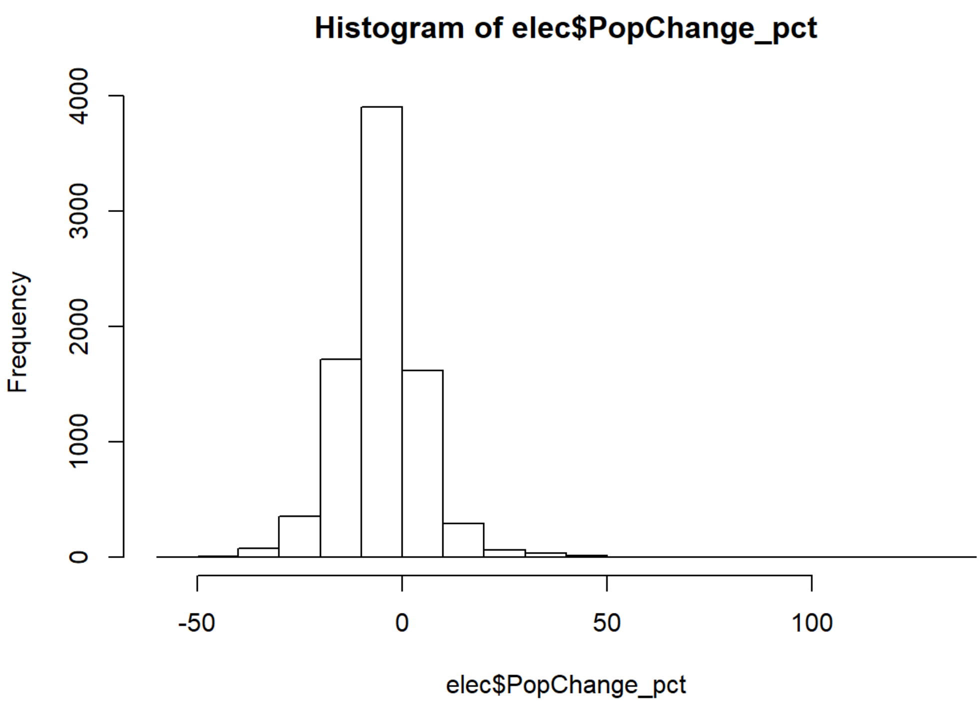


We have to pay special attention to outliers in this dataset. This is because some outliers are completely valid. For example, looking at the 'Population' variable there are two clear outliers ('Madrid and Barcelona') that should be included in the analysis. However, I cannot find any reasonable argument justifying a 138.46% population change in one year. Thus, we are going to modify those values.

IMPORTANT: We are not going to convert continuous variables to categorical. This is because we have several continuous variables and factorization will dramatically increase computational costs for 'lm' and 'glm' models. The main drawback of keeping continuous variables is that we might miss out non-linear relationships in our data.

Later on, we will see that it is possible to fix this problem by conducting mathematical transformations (e.g., x^2, log(x), …) in our continuous variables. By this way, linearity might be created with the target variable

```
# 5) Check if 138.46 is a valid max value for 'PobChange_pct'
hist(elec$PopChange_pct)
```




```
summary(elec$PopChange_pct)
```

```
##      Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's 
## -52.2700 -10.4000  -4.9600  -4.8974   0.0925 138.4600        7
```

```
# The following function counts the number of outliers and  transform it to missings
# (Function provided by my teacher)
outliersToMissing<-function(varaux){
  if (abs(skew(varaux))<1){
    criteria1<-abs((varaux-mean(varaux,na.rm=T))/sd(varaux,na.rm=T))>3
  } else {
    criteria1<-abs((varaux-median(varaux,na.rm=T))/mad(varaux,na.rm=T))>8
  }
  qnt <- quantile(varaux, probs=c(.25, .75), na.rm = T)
  H <- 3 * IQR(varaux, na.rm = T)
  criteria2<-(varaux<(qnt[1] - H))|(varaux>(qnt[2] + H))
  varaux[criteria1&criteria2]<-NA
  return(list(varaux,sum(criteria1&criteria2,na.rm=T)))
}


# (Function provided by my teacher)
quantVarNA_conversion<-function(vv,type){#type only accepts three possible values:'mean', 'median'
                                                                                  # or 'random'
  if (type=="mean"){
    vv[is.na(vv)]<-round(mean(vv,na.rm=T),4)
  } else if (type=="median"){
    vv[is.na(vv)]<-round(median(vv,na.rm=T),4)
  } else if (type=="random"){
    dd<-density(vv,na.rm=T,from=min(vv,na.rm = T),to=max(vv,na.rm = T))
    vv[is.na(vv)]<-round(approx(cumsum(dd$y)/sum(dd$y),dd$x,runif(sum(is.na(vv))))$y,4)
  }
  vv
}



elec$PopChange_pct=outliersToMissing(elec$PopChange_pct)[[1]]
elec$PopChange_pct=quantVarNA_conversion(elec$PopChange_pct,'random')
summary(elec$PopChange)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
## -52.270 -10.420  -4.970  -5.018   0.055  54.050
```

```
# 6) 'officesNfacilities': max value is '99999', while median is 52
unique(elec[elec$officesNfacilities>4000,c('Name','Population','ProvinceCode',
                                           'Autonomous_Community',
                                           'officesNfacilities')]$officesNfacilities)
```

```
## [1] 99999  4006  4759
```

```
# We see there is a clear significant difference between the greatest value (99999) and
# the second greatest (4759).
# We are not going to use the previous outlier-detector function, but only convert 99999 values to
# NA

elec$officesNfacilities=ifelse(elec$officesNfacilities==99999, median(elec$officesNfacilities),
                               elec$officesNfacilities)
summary(elec$officesNfacilities)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##     1.0    22.0    52.0   120.6   124.0  4759.0
```

```
# 7) Having a 100% Unemployment level at a specific age range is suspicious.
# It is worth mentioning that if one of these variables ('UnemployLess25_Ptge',
# 'Unemploy25_40_Ptge', 'UnemployMore40_Ptge') have 100% as value, the remaining ones will be zero
# We are going to convert 100% rows to NAs and then pass the 'quantVarNA_conversion' function
# to those NA values.

elec[(elec$UnemployLess25_Ptge==100|elec$Unemploy25_40_Ptge==100|elec$UnemployMore40_Ptge==100)
      | (elec$UnemployLess25_Ptge==0 & elec$Unemploy25_40_Ptge==0 & elec$UnemployMore40_Ptge==0),
                       c('UnemployLess25_Ptge', 'Unemploy25_40_Ptge','UnemployMore40_Ptge')] = NA


elec$UnemployLess25_Ptge=quantVarNA_conversion(elec$UnemployLess25_Ptge,'median')
elec$Unemploy25_40_Ptge=quantVarNA_conversion(elec$Unemploy25_40_Ptge,'median')
# elec$UnemployMore40_Ptge=quantVarNA_conversion(elec$UnemployMore40_Ptge,'random')

More40NA=elec[is.na(elec$UnemployMore40_Ptge),c('UnemployLess25_Ptge', 'Unemploy25_40_Ptge',
                                       'UnemployMore40_Ptge')]
More40NA=apply(More40NA[,c('UnemployLess25_Ptge', 'Unemploy25_40_Ptge')], 1,
               function(x) 100-sum(x)) #so that Unem_less_25 + Unem_25_40 + Unem_more_40 = 100

elec[is.na(elec$UnemployMore40_Ptge),c('UnemployMore40_Ptge')]=More40NA

summary(elec[,c('UnemployLess25_Ptge', 'Unemploy25_40_Ptge','UnemployMore40_Ptge')])
```

```
##  UnemployLess25_Ptge Unemploy25_40_Ptge UnemployMore40_Ptge
##  Min.   : 0.000      Min.   : 0.00      Min.   : 0.00
##  1st Qu.: 4.040      1st Qu.:35.00      1st Qu.:45.45
##  Median : 7.005      Median :40.93      Median :52.06
##  Mean   : 8.050      Mean   :40.32      Mean   :51.63
##  3rd Qu.:10.361      3rd Qu.:45.83      3rd Qu.:57.14
##  Max.   :80.000      Max.   :87.50      Max.   :94.74
```

```
#SAME IDEA WITH INDUSTRY FUNCTION
               # We are going to convert this values to NAs and then pass the 'quantVarNA_conversion' functi
on
# to those NA values.

elec[(elec$AgricultureUnemploymentPtge==100|elec$IndustryUnemploymentPtge==100|elec$ConstructionUnemployment
Ptge==100|elec$ServicesUnemploymentPtge==100)
      | (elec$AgricultureUnemploymentPtge==0 & elec$IndustryUnemploymentPtge==0 & elec$ConstructionUnemploym
entPtge==0 & elec$ServicesUnemploymentPtge==0),
                       c('AgricultureUnemploymentPtge','IndustryUnemploymentPtge',
                         'ConstructionUnemploymentPtge','ServicesUnemploymentPtge')] = NA



elec$ServicesUnemploymentPtge = quantVarNA_conversion(elec$ServicesUnemploymentPtge,'median')
elec$IndustryUnemploymentPtge = quantVarNA_conversion(elec$IndustryUnemploymentPtge,'median')
elec$ConstructionUnemploymentPtge = quantVarNA_conversion(elec$ConstructionUnemploymentPtge,'median')

AgricultureNA=elec[is.na(elec$AgricultureUnemploymentPtge),c('IndustryUnemploymentPtge',
                         'ConstructionUnemploymentPtge','ServicesUnemploymentPtge','AgricultureUnemployment
Ptge')]

AgricultureNA=apply(AgricultureNA[,c('IndustryUnemploymentPtge','ConstructionUnemploymentPtge',
                                     'ServicesUnemploymentPtge')], 1, function(x) 100-sum(x))

elec[is.na(elec$AgricultureUnemploymentPtge),c('AgricultureUnemploymentPtge')]=AgricultureNA

summary(elec[,c('AgricultureUnemploymentPtge','IndustryUnemploymentPtge',
                'ConstructionUnemploymentPtge','ServicesUnemploymentPtge')])
```

```
##  AgricultureUnemploymentPtge IndustryUnemploymentPtge
##  Min.   : 0.000              Min.   : 0.000
##  1st Qu.: 1.573              1st Qu.: 4.617
##  Median : 7.692              Median : 8.929
##  Mean   :10.813              Mean   :10.794
##  3rd Qu.:19.317              3rd Qu.:14.286
##  Max.   :90.909              Max.   :82.000
##  ConstructionUnemploymentPtge ServicesUnemploymentPtge
##  Min.   : 0.000               Min.   : 0.00
##  1st Qu.: 6.365               1st Qu.:53.85
##  Median : 9.849               Median :61.91
##  Mean   :11.703               Mean   :60.34
##  3rd Qu.:14.118               3rd Qu.:68.42
##  Max.   :86.486               Max.   :95.65
```

```
# let's make sure there are no NAs in our debugged dataset
sapply(elec[,sapply(elec, function(x) sum(is.na(x))>0)],
                                     function(x) is.numeric(x)) #We make distiction between
```

```
##      SameAutomComPtge        totalCompanies              Industry
##                  TRUE                  TRUE                  TRUE
##       ConstructionInd     commerceNhostelry            ServiceInd
##                  TRUE                  TRUE                  TRUE
##          RealProperty               Pob2010           SurfaceArea
##                  TRUE                  TRUE                  TRUE
##               Density People_RealProp_ratio
##                 FALSE                  TRUE
```

```
                                                                #numeric and non_numberic vars
```

variables with NAs are split into factor variables and numeric ones

```
quantNAvars=rownames(data.frame(sapply(elec[,sapply(elec, function(x) sum(is.na(x))>0)],
function(x) is.numeric(x))))[!(rownames(data.frame(sapply(elec[,sapply(elec,
function(x) sum(is.na(x))>0)], function(x) is.numeric(x)))) %in% 'Density')]

elec[,quantNAvars]=sapply(quantNAvars, function(x) quantVarNA_conversion(elec[,x],'median'))

categNAvars='Density'
elec$Density[is.na(elec$Density)] = '4'
elec$Density=as.factor(elec$Density)

sapply(elec, function(x) sum(is.na(x)))
```

```
##                          Name                  ProvinceCode
##                             0                             0
##           Autonomous_Community                    Population
##                             0                             0
##                   TotalCensus                AbstentionPtge
##                             0                             0
##           High_Abstention_rate                Left_wing_Pct
##                             0                             0
##                 Right_wing_Pct                    Others_Pct
##                             0                             0
##                      Left_wing                    Right_wing
##                             0                             0
##                    Age_0_4_Ptge               Age_under19_Ptge
##                             0                             0
##                   Age_19_65_pct                Age_over65_pct
##                             0                             0
##          FemalePopulationPtge               ForeignersPtge
##                             0                             0
##               SameAutomComPtge      SameAutonComDiffProvPtge
##                             0                             0
##                DifAutonComPtge           UnemployLess25_Ptge
##                             0                             0
##              Unemploy25_40_Ptge          UnemployMore40_Ptge
##                             0                             0
##   AgricultureUnemploymentPtge      IndustryUnemploymentPtge
##                             0                             0
## ConstructionUnemploymentPtge      ServicesUnemploymentPtge
##                             0                             0
##                 totalCompanies                      Industry
##                             0                             0
##                ConstructionInd             commerceNhostelry
##                             0                             0
##                     ServiceInd                  MainActivity
##                             0                             0
##                   RealProperty                       Pob2010
##                             0                             0
##                    SurfaceArea                       Density
##                             0                             0
##                  PopChange_pct         People_RealProp_ratio
##                             0                             0
##             officesNfacilities
##                             0
```

```
#There should be no NAs
```

save file as RDS

```
saveRDS(elec, 'Debugged_elect_dataset')
```

LINEAR MODELS

LET'S DEFINE A BINARY TARGET VARIABLE FOR LOGISTIC REGRESSION AND CONTINUOUS TARGET VARIABLE FOR LINEAR REGRESSION

BINARY -> Right_wing CONTINUOUS -> AbstentionPtge

Let's get rid of the other potential target variables that we are not going to use We are going to remove 'Name' since each observation has a unique name.

```
options(warn=-1)

rowdata=as.data.frame(readRDS('C:/Users/pablo/Desktop/Debugged_elect_dataset'))

df=rowdata[,!(names(rowdata) %in% c('Left_wing_Pct','Right_wing_Pct','Others_Pct',
                                    'High_Abstention_rate','Left_wing','Right_wing','Name',
                                    'AbstentionPtge'))]
total_lm=as.data.frame(cbind(AbstentionPtge=rowdata$AbstentionPtge,df))

# head(total_lm,2)
```

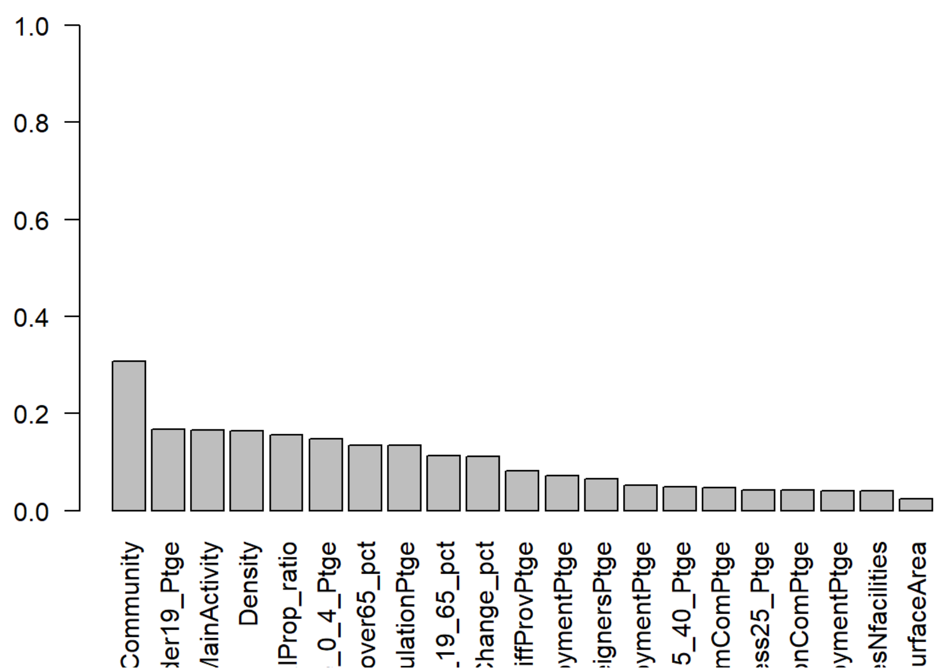
IMPORTANT: there are collinearity issues in both unemployment and industry variables:

```r
# In order to avoid collinearity, We just have to remove one of the variables.
total_lm=total_lm[, !(names(total_lm) %in% c('UnemployMore40_Ptge','AgricultureUnemploymentPtge'))]
total_lm[1:2, 14:20] #Let's make sure they have been successfully removed
```

```
##   DifAutonComPtge UnemployLess25_Ptge Unemploy25_40_Ptge
## 1          19.345               2.381             54.762
## 2           7.226              16.216             32.432
##   IndustryUnemploymentPtge ConstructionUnemploymentPtge
## 1                    9.524                       11.905
## 2                    8.108                       10.811
##   ServicesUnemploymentPtge totalCompanies
## 1                   73.810             15
## 2                   67.568             11
```

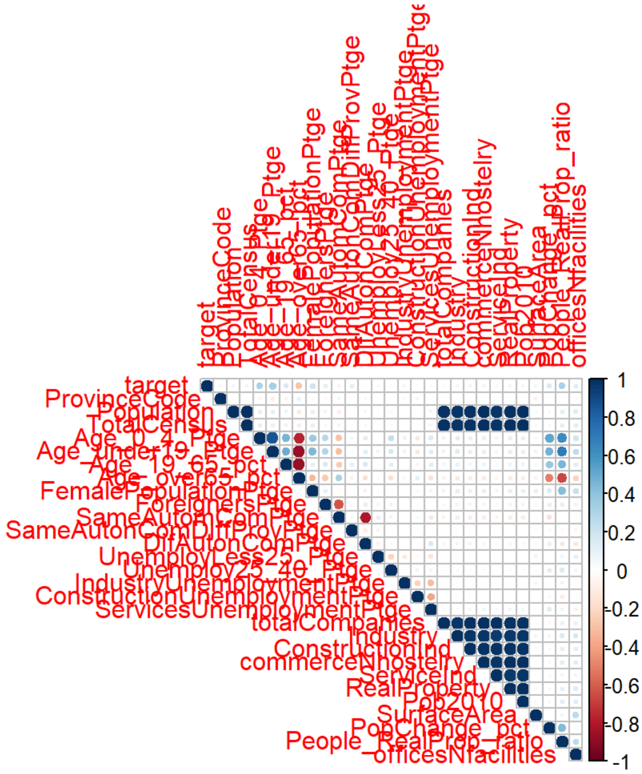
Significance of variables

```r
# functions provided by my teacher
Vcramer<-function(v,target){
  if (is.numeric(v)){
    v<-cut(v,5)
  }
  if (is.numeric(target)){
    target<-cut(target,5)
  }
  cramer.v(table(v,target))
}

VcramerGraph<-function(matrix, target){
  outputVcramer<-sapply(matrix,function(x) Vcramer(x,target))
  barplot(sort(outputVcramer,decreasing =T),las=2,ylim=c(0,1))
}

VcramerGraph(total_lm[,3:ncol(total_lm)], total_lm$AbstentionPtge)
```



correlation among variables (only for numeric ones)

```r
input=total_lm[,!(names(total_lm) %in% c('AbstentionPtge','Right_wing'))]
target=total_lm$AbstentionPtge

corrplot(cor(cbind(target,Filter(is.numeric, input)), use="pairwise",
             method="pearson"), method = "circle",type = "upper")
```



IMPORTANT: We have not converted continuous variables to categorical. This is because we have several continuous variables and factorization will dramatically increase computational costs for ‘lm’ and ‘glm’ models. The main drawback of keeping variables continuous is that we might miss out non-linear relationships in our data. it is possible to fix this problem by conducting mathematical transformations (e.g., x^2, log(x), …) in our continuous variables. This is exactly what it is done in the following lines of code:

```
# function provided by my teacher
bestTransfCorr<-function(vv,target){
  vv<-scale(vv)
  vv<-vv+abs(min(vv,na.rm=T))*1.0001
  posiblesTransf<-data.frame(x=vv,logx=log(vv),expx=exp(vv),sqrx=vv^2,sqrtx=sqrt(vv),cuartax=vv^4,raiz4=vv^(
1/4))
  return(list(colnames(posiblesTransf)[which.max(abs(cor(target,posiblesTransf, use="complete.obs")))],posib
lesTransf[,which.max(abs(cor(target,posiblesTransf, use="complete.obs")))]))
}

only_numeric_vars=total_lm[,as.vector(sapply(total_lm, function(x) is.numeric(x))) & (!(names(total_lm) %in%
'AbstentionPtge'))]
transformed_total_lm=total_lm
transformed_total_lm[,as.vector(sapply(transformed_total_lm, function(x) is.numeric(x))) & (!(names(transfor
med_total_lm) %in% 'AbstentionPtge'))] =
sapply(only_numeric_vars, function(x) bestTransfCorr(x,total_lm$AbstentionPtge)[[2]])

#head(transformed_total_lm,2)
#head(total_lm,2)
```

Let’s check how useful these mathematical transformations have been. To do so, R_square of both normal and transformed variables will be evaluated. ‘transformed_data’ will be used from now on

```
# MY FIRST MODEL
# Function provided by my teacher
Rsq<-function(model,target,data){
  testpredicted<-predict(model, data)
  testReal<-data[,target]
  sse <- sum((testpredicted - testReal) ^ 2)
  sst <- sum((testReal - mean(testReal)) ^ 2)
  1 - sse/sst
}

model1=lm(total_lm$AbstentionPtge ~ ., total_lm)
paste('R_sq_normal_data', round(Rsq(model1,'AbstentionPtge', total_lm), digits=3), '%')
```

```
## [1] "R_sq_normal_data 0.382 %"
```

```r
transformedModel=lm(total_lm$AbstentionPtge ~ ., transformed_total_lm)
paste('R_sq_transformed_data',round(Rsq(transformedModel,'AbstentionPtge',
                                         transformed_total_lm), digits=3), '%')
```

```
## [1] "R_sq_transformed_data 0.403 %"
```

Let's create all possible combinations of variables

```r
# function provided by teacher
#It generates all possible interactions
varInteractions<-function(data,position){ #position refers to index where target var is
  list_of_factors<-c()
  list_of_interactions<-paste(names(data)[position],'~')
  names<-names(data)
  for (i in (1:length(names))[-position]){
    list_of_interactions<-paste(list_of_interactions,names[i],'+')
    if (class(data[,i])=="factor"){
      list_of_factors<-c(list_of_factors,i)
      for (j in ((1:length(names))[-c(position,list_of_factors)])){
        list_of_interactions<-paste(list_of_interactions,names[i],':',names[j],'+')
      }
    }
  }
  list_of_interactions<-substr(list_of_interactions, 1, nchar(list_of_interactions)-1)
  list_of_interactions
}

transformed_interactions=varInteractions(transformed_total_lm, 1)
```

Let's check R_sq of model including all possible interactions

```r
# summary(lm(transformed_interactions, data_train))
Rsq(lm(transformed_interactions, transformed_total_lm), 'AbstentionPtge', transformed_total_lm) # a little b
it better
```

```
## [1] 0.4392983
```

This model only includes the most significant variables with no interactions

```r
handmadeModel=lm(AbstentionPtge ~ Autonomous_Community + TotalCensus + FemalePopulationPtge + SameAutomComPt
ge + ConstructionUnemploymentPtge + totalCompanies + Industry + ConstructionInd + commerceNhostelry + Servic
eInd + MainActivity + RealProperty + SurfaceArea + PopChange_pct, transformed_total_lm)
# I only include the most significant variables with no interactions
# summary(handmadeModel)
Rsq(handmadeModel,'AbstentionPtge', transformed_total_lm)
```

```
## [1] 0.3944459
```

Most significant variables. Now including interactions

```r
handmadeModel2=lm(AbstentionPtge ~ Autonomous_Community + TotalCensus + FemalePopulationPtge + SameAutomComP
tge + ConstructionUnemploymentPtge + totalCompanies + Industry + ConstructionInd + commerceNhostelry + Servi
ceInd + MainActivity + RealProperty + SurfaceArea + PopChange_pct + MainActivity : Population + MainActivity
: TotalCensus + MainActivity : totalCompanies + MainActivity : PopChange_pct, transformed_total_lm)

# Most significant variables. Now including interactions
# summary(handmadeModel2)
Rsq(handmadeModel2,'AbstentionPtge', transformed_total_lm)
```

```
## [1] 0.4060081
```

OUTPUT OF THIS CELL IS TOO LONG. THAT'S WHY IT IS NOT GOING TO BE RUN

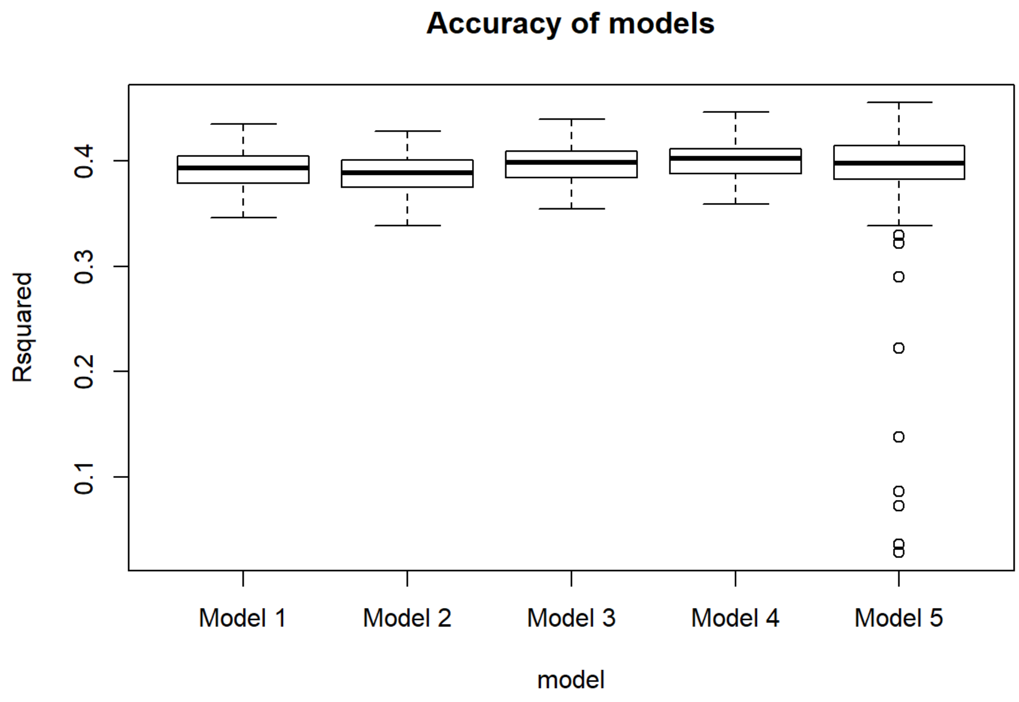
```
# nullmodel=lm(AbstentionPtge ~ 1, transformed_total_lm)
# fullmodel=lm(transformed_interactions, transformed_total_lm)

# BICmodel=step(nullmodel, scope=list(lower=nullmodel, upper=fullmodel), direction='both',
#               k=log(nrow(transformed_total_lm)))
# # summary(BICmodel)
# # Rsq(BICmodel, 'AbstentionPtge', transformed_total_lm) >>>0.407283270714246

# AICmodel=step(nullmodel, scope=list(lower=nullmodel, upper=fullmodel), direction='both')
# # summary(AICmodel)
# # Rsq(AICmodel, 'AbstentionPtge', transformed_total_lm) >>>0.420388579593511
```

THESE ARE THE LIST OF THE POTENTIAL MODELS:

```
nullmodel=lm(AbstentionPtge ~ 1, transformed_total_lm)
fullmodel=lm(transformed_interactions, transformed_total_lm)

model1=lm(AbstentionPtge ~ ., transformed_total_lm)

handmadeModel=lm(AbstentionPtge ~ Autonomous_Community + TotalCensus + FemalePopulationPtge +
                 SameAutomComPtge + ConstructionUnemploymentPtge + totalCompanies + Industry +
                 ConstructionInd + commerceNhostelry + ServiceInd + MainActivity + RealProperty +
                 SurfaceArea + PopChange_pct, transformed_total_lm)

handmadeModel2=lm(AbstentionPtge ~ Autonomous_Community + TotalCensus + FemalePopulationPtge +
                  SameAutomComPtge + ConstructionUnemploymentPtge + totalCompanies + Industry +
                  ConstructionInd + commerceNhostelry + ServiceInd + MainActivity +
                  RealProperty + SurfaceArea + PopChange_pct + MainActivity : Population +
                  MainActivity : TotalCensus + MainActivity : totalCompanies +
                  MainActivity : PopChange_pct, transformed_total_lm)

BICmodel=lm(AbstentionPtge ~ Autonomous_Community + Density + FemalePopulationPtge +
    Age_19_65_pct + Industry + People_RealProp_ratio + SameAutonComDiffProvPtge +
    Population + MainActivity + SurfaceArea + Density:People_RealProp_ratio +
    FemalePopulationPtge:MainActivity, transformed_total_lm)



AICmodel=lm(AbstentionPtge ~ Autonomous_Community + RealProperty + MainActivity +
    FemalePopulationPtge + Age_19_65_pct + ProvinceCode + SameAutonComDiffProvPtge +
    People_RealProp_ratio + Population + SurfaceArea + Industry +
    SameAutomComPtge + Age_under19_Ptge + ConstructionInd + Age_over65_pct +
    Autonomous_Community:MainActivity + RealProperty:MainActivity +
    MainActivity:FemalePopulationPtge + MainActivity:SameAutonComDiffProvPtge +
    MainActivity:Age_19_65_pct + MainActivity:Age_under19_Ptge, transformed_total_lm)
```

boxplot of models' accuracy.Data split method: cross validation * BEST MODEL (in terms of R_squared_mean): MODEL4 -> BICmodel * BEST MODEL (in terms of R_squared_standard_deviation): MODEL3 -> handmadeModel2

```
data=transformed_total_lm
# data$AbstentionPtge=make.names(data$AbstentionPtge) #this step is mandatory, otherwise
# train() will return an error

models=sapply(list(model1, handmadeModel, handmadeModel2, BICmodel, AICmodel),
              function(x) formula(x))

total=c()
for (i in 1:length(models)) {
    set.seed(652)
    cross_validation=train(as.formula(models[[i]]), data=data, method='lm',
                           trControl=trainControl(method="repeatedcv", number=5, repeats=20,
                                                  returnResamp="all"))
  total<-rbind(total,data.frame(cross_validation$resample,
                                model=rep(paste("Model",i),nrow(cross_validation$resample))))
}

# head(total,3)
boxplot(Rsquared~model,data=total,main="Accuracy of models")
```




```
sort_by_mean=as.data.frame(aggregate(Rsquared ~ model, data=total, mean))
by_mean=sort_by_mean[order(-sort_by_mean$Rsquared),]
names(by_mean)=c('Model','Rsquared_mean')
by_mean
```

```
##     Model Rsquared_mean
## 4 Model 4     0.4008136
## 3 Model 3     0.3975049
## 1 Model 1     0.3924913
## 2 Model 2     0.3884753
## 5 Model 5     0.3809804
```

```
sort_by_SD=as.data.frame(aggregate(Rsquared ~ model, data=total, sd))
by_SD=sort_by_SD[order(sort_by_SD$Rsquared),]
names(by_SD)=c('Model','Rsquared_SD')
by_SD
```

```
##     Model Rsquared_SD
## 3 Model 3  0.01807718
## 2 Model 2  0.01842427
## 1 Model 1  0.01849136
## 4 Model 4  0.01867398
## 5 Model 5  0.07810699
```

Logistic regression models BINARY (our targer var for glm) -> Right_wing CONTINUOUS (previous lm model) -> AbstentionPtge

```
### LOGISTIC REGRESSION MODEL FOR BINARY TARGET VARIABLE
# Let's get rid of the other potential target variables we havent used
noTargetVars=rowdata[,!(names(rowdata) %in% c('Name','Left_wing_Pct','Right_wing_Pct','Others_Pct',
                                             'High_Abstention_rate','Left_wing','AbstentionPtge',
                                             'Right_wing'))]

# BINARY (our targer var for glm) -> Right_wing
# CONTINUOUS (previous lm model) -> AbstentionPtge

total_glm=cbind(Right_wing=rowdata$Right_wing, noTargetVars)
#head(total_glm,2)
```

(We use Vcramer as metric for quantitative variable transformations in logistic regressions because it is based on Pearson’s chi-squared statistic)

```r
# function provided by my teacher
bestTransfVcramer<-function(vv,target){
  vv<-scale(vv)
  vv<-vv+abs(min(vv,na.rm=T))*1.0001
  possibleTransf<-data.frame(x=vv,logx=log(vv),expx=exp(vv),sqrx=vv^2,sqrtx=sqrt(vv),cuartax=vv^4,raiz4=vv^(
1/4))
  return(list(colnames(possibleTransf)[which.max(apply(possibleTransf,2,function(x) cramersV(x,target)))],po
ssibleTransf[,which.max(apply(possibleTransf,2,function(x) cramersV(x,target)))]))
}

only_numeric_vars=total_glm[,as.vector(sapply(total_glm, function(x) is.numeric(x))) & (!(names(total_glm) %
in% 'Right_wing'))]


transfData=total_glm
transfData[,as.vector(sapply(total_glm, function(x) is.numeric(x))) & (!(names(total_glm) %in% 'Right_wing')
)]=sapply(only_numeric_vars,
                     function(x) bestTransfVcramer(x,total_glm$Right_wing)[[2]])
```

We obtain same accuracy with transfData. We will use ‘transfData’ from now on, even though it does not matter using ‘total_glm’

```r
# Function provided by my teacher
#Evaluate logistic_regression_pseudo-R2 in any data set
pseudoR2<-function(modelo,dd,nombreVar){
  pred.out.link <- predict(modelo, dd, type = "response")
  mod.out.null <- glm(as.formula(paste(nombreVar,"~1")), family = binomial, data = dd)
  pred.out.linkN <- predict(mod.out.null, dd, type = "response")
  1-sum((dd[,nombreVar]==1)*log(pred.out.link)+log(1 -pred.out.link)*(1-(dd[,nombreVar]==1)))/
    sum((dd[,nombreVar]==1)*log(pred.out.linkN)+log(1 -pred.out.linkN)*(1-(dd[,nombreVar]==1)))
}

previousModel=glm(Right_wing ~ ., data=total_glm, family=binomial)
paste('pseudoR2_normal_data',round(pseudoR2(previousModel,total_glm,"Right_wing"),
                                    digits=3), '%')
```

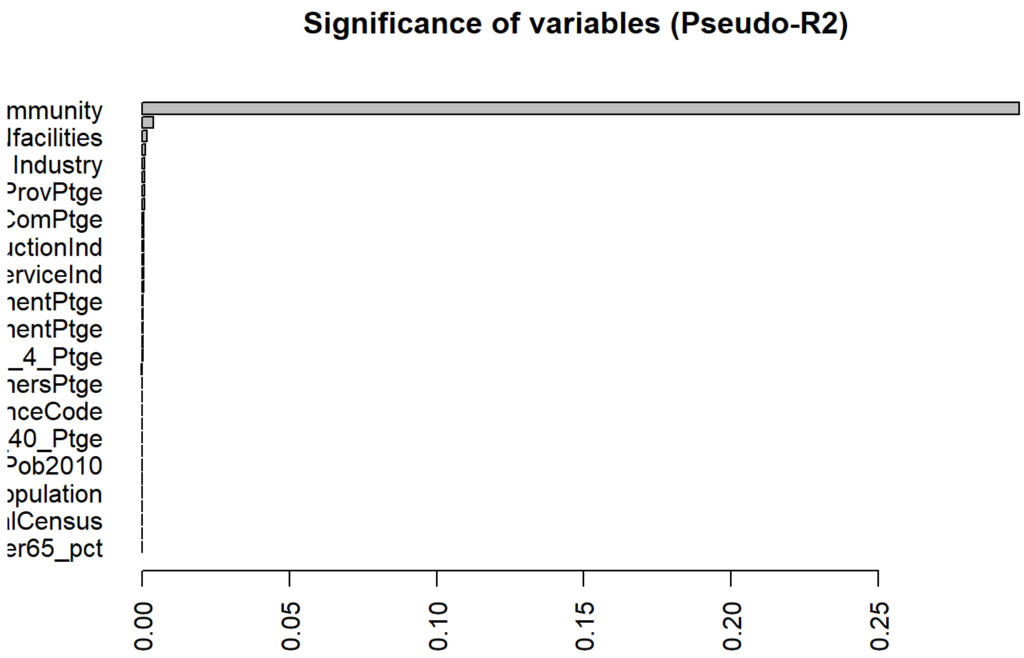
```
## [1] "pseudoR2_normal_data 0.449 %"
```

```r
transfModel=glm(Right_wing ~ ., data=transfData, family=binomial)
paste('pseudoR2_transfData',round(pseudoR2(transfModel,transfData,"Right_wing"),
                                    digits=3), '%')
```

```
## [1] "pseudoR2_transfData 0.449 %"
```

Graph displyaing input features’ significance in logistic regression

```r
# function provided by my teacher
impVariablesLog<-function(modelo,nombreVar,dd=data_train){
  null<-glm(as.formula(paste(nombreVar,"~1")),data=dd,family=binomial)
  aux2<-capture.output(aux<-step(modelo, scope=list(lower=null, upper=modelo), direction="backward",k=0,step
s=1))
  aux3<-read.table(textConnection(aux2[grep("-",aux2)]))[,c(2,5)]
  aux3$V5<-(aux3$V5-modelo$deviance)/modelo$null.deviance
  barplot(aux3$V5,names.arg = aux3$V2,las=2,horiz=T,main="Significance of variables (Pseudo-R2)")
}

impVariablesLog(transfModel, 'Right_wing', transfData)
```



As may be inferred, 'Autonomous_Community' is clearly the most significant variable (This pseudoR2 confirms what we have already observed in the 'impVariablesLog' graph)

```
AACCmodel=glm(Right_wing ~ Autonomous_Community, data=transfData, family=binomial)
pseudoR2(AACCmodel, transfData, 'Right_wing')
```

```
## [1] 0.4279396
```

Best score at the moment. Then main drawback is that it includes a lot of variables and we are not looking for super complex models. We want something easier to interpret.

```
transformed_interactions=varInteractions(transfData, 1)
fullmodel=glm(transformed_interactions, family=binomial, data=transfData)
pseudoR2(fullmodel, transfData, 'Right_wing')
```

```
## [1] 0.4930704
```

handmadeModel: including those interactions with the lowest p-value. (Selected interactions at least have '**' as significance level)

```
formula=as.formula('Right_wing ~ . + totalCompanies:MainActivity + Industry:Density + ConstructionInd:Densit
y + SameAutonComDiffProvPtge:Density +
UnemployLess25_Ptge:MainActivity + Unemploy25_40_Ptge:MainActivity + UnemployMore40_Ptge:MainActivity')
handmadeModel=glm(formula, data=transfData, family=binomial)
# handmadeModel
pseudoR2(handmadeModel, transfData, 'Right_wing')
```

```
## [1] 0.4572729
```

The following cell is not going to be run, since output gets too long

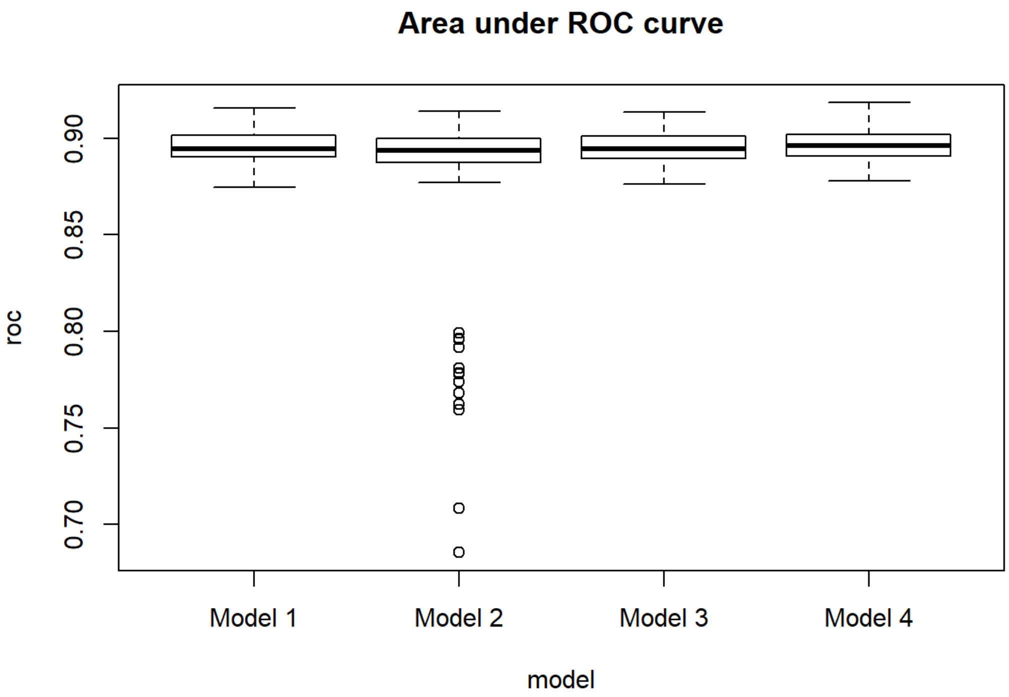
```
# nullmodel=glm(Right_wing ~ 1, family=binomial, data=transfData)

# fullmodel=glm(transformed_interactions, family=binomial, data=transfData)


# BICmodel=step(nullmodel, scope=list(lower=nullmodel, upper=fullmodel), direction='both',
#               k=log(nrow(transfData)))

# AICmodel=step(nullmodel, scope=list(lower=nullmodel, upper=fullmodel), direction='both')
```

MODELS:

```r
firstmodelFormula=as.formula('Right_wing ~ .')

handmadeModelFormula=as.formula('Right_wing ~ . + totalCompanies:MainActivity + Industry:Density +
    ConstructionInd:Density + SameAutonComDiffProvPtge:Density +
    UnemployLess25_Ptge:MainActivity + Unemploy25_40_Ptge:MainActivity +
    UnemployMore40_Ptge:MainActivity')

BICmodelFormula=as.formula('Right_wing ~ Autonomous_Community + Age_19_65_pct +
    officesNfacilities + MainActivity + ServiceInd + RealProperty + ForeignersPtge +
    commerceNhostelry')

AICmodelFormula=as.formula('Right_wing ~ Autonomous_Community + MainActivity + ServiceInd +
    Age_19_65_pct + ForeignersPtge + RealProperty + officesNfacilities +
    commerceNhostelry + SameAutonComDiffProvPtge + People_RealProp_ratio +
    PopChange_pct + UnemployLess25_Ptge + SurfaceArea + ServicesUnemploymentPtge +
    IndustryUnemploymentPtge + Autonomous_Community:MainActivity +
    MainActivity:ForeignersPtge + MainActivity:officesNfacilities +
    MainActivity:UnemployLess25_Ptge + MainActivity:SameAutonComDiffProvPtge')

model1=glm(firstmodelFormula, family=binomial, data=transfData)
model2=glm(handmadeModelFormula, family=binomial, data=transfData)
model3=glm(BICmodelFormula, family=binomial, data=transfData)
model4=glm(AICmodelFormula, family=binomial, data=transfData)
```

boxplot of models' accuracy. (Data split mehtod: cross validation) * BEST MODEL: MODEL4 -> AICmodel

```r
data=transfData
# data$AbstentionPtge=make.names(data$AbstentionPtge) #this step is mandatory, otherwise
# train() will return an error

data$Right_wing=make.names(data$Right_wing) #convert target to character: values -> 'X0' and 'X1'

models=sapply(list(model1, model2, model3, model4), function(x) formula(x))

total=c()
for (i in 1:length(models)) {
    set.seed(192)
    cross_validation=train(as.formula(models[[i]]), data=data, method='glm', family='binomial',
                           metric='ROC', trControl=trainControl(method="repeatedcv", number=5,
                                                                repeats=20,
                                                                summaryFunction=twoClassSummary,
                                                                classProbs=TRUE,
                                                                returnResamp="all"))

  total<-rbind(total,data.frame(roc=cross_validation$resample[,1],
                                model=rep(paste("Model",i),nrow(cross_validation$resample))))
}

boxplot(roc~model,data=total,main="Area under ROC curve")
```




```
sort_by_mean=as.data.frame(aggregate(roc ~ model, data=total, mean))
sort_by_mean=sort_by_mean[order(-sort_by_mean$roc),]
names(sort_by_mean)=c('Model','Rsquared_mean')
sort_by_mean
```

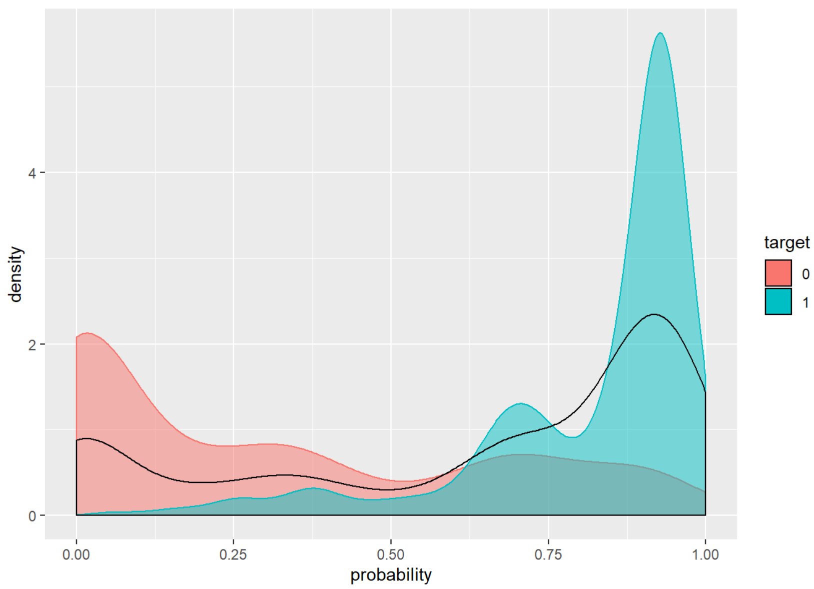
```
##     Model Rsquared_mean
## 4 Model 4     0.8963969
## 1 Model 1     0.8956563
## 3 Model 3     0.8950435
## 2 Model 2     0.8790268
```

```
sort_by_SD=as.data.frame(aggregate(roc ~ model, data=total, sd))
sort_by_SD=sort_by_SD[order(sort_by_SD$roc),]
names(sort_by_SD)=c('Model','Rsquared_sd')
sort_by_SD
```

```
##     Model Rsquared_sd
## 4 Model 4 0.007483708
## 3 Model 3 0.007617589
## 1 Model 1 0.007639005
## 2 Model 2 0.045436108
```

distribution of values of the binary target variable (Right_wing)

```
options(repr.plot.width=4, repr.plot.height=3)
data=transfData

trainIndex=createDataPartition(data$Right_wing, p=0.8, list=FALSE)  #we make this transformation before visu
alizing models' boxplots
data_train=data[trainIndex,]
data_test=data[-trainIndex,]

winner_model=model4


# function provided by my teacher
hist_binaryTarget=function(var, target, title_of_x_axis) {
    values=data.frame(variable=var, target=target)
    ggplot(values, aes(x=var)) +
        geom_density(aes(colour=target, fill=target), alpha=0.5) +
        geom_density(lty=1) +
        xlab(title_of_x_axis)
}

set.seed(4545)
trainIndex=createDataPartition(data$Right_wing, p=0.8, list=FALSE)
data_train=data[trainIndex,]
data_test=data[-trainIndex,]

y_pred=predict(winner_model, newdata=data_test, type='response')
hist_binaryTarget(y_pred, data_test$Right_wing,'probability')
```



The by default partition point is 0.5 in logistic regressions. Let’s check whether 0.5 is actually the optimal partition value

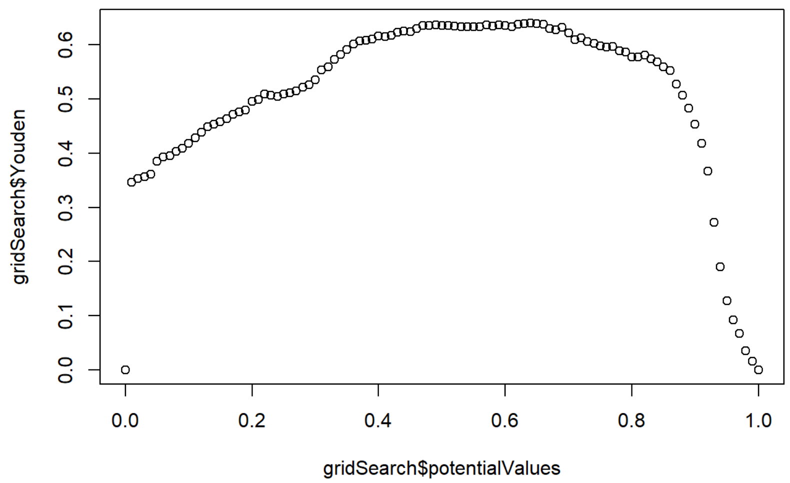
```r
# function provided by my teacher
partitionPointMetrics=function(model, dataset, target_var_as_string,
                               partitionValue, positive_class) {
    probs=predict(model, newdata=dataset, type='response')
    cm=confusionMatrix(data=factor(ifelse(probs > partitionValue, 1, 0)),
                       reference=dataset[, target_var_as_string], positive=positive_class)
    c(cm$overall[1], cm$byClass[1:4])
}

# Let's test values
potentialValues=seq(0,1,0.01) #sequence of 0.01 evenly splitted values between 0 and 1.

# data_test[,'Right_wing']
gridSearch=data.frame(cbind(potentialValues, t(sapply(potentialValues, function(x)
    partitionPointMetrics(winner_model, data_test, 'Right_wing',x,'1')))))


gridSearch$Youden=gridSearch$Sensitivity + gridSearch$Specificity - 1 #we have included a new
                                                                      #accuracy method

# optimal values
gridSearch[gridSearch$Youden==max(gridSearch$Youden),c('potentialValues','Youden')]
```

```
##    potentialValues    Youden
## 65            0.64 0.6393099
```

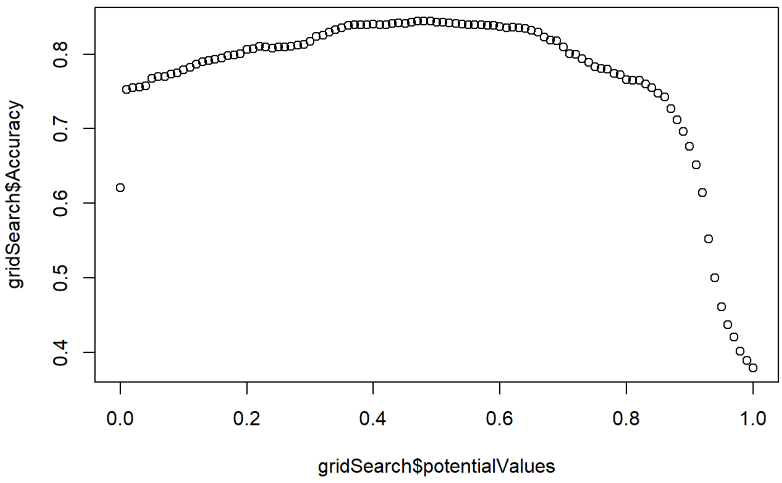
```r
gridSearch[gridSearch$Accuracy==max(gridSearch$Accuracy),c('potentialValues','Accuracy')]
```

```
##    potentialValues  Accuracy
## 48            0.47 0.8441158
## 49            0.48 0.8441158
## 50            0.49 0.8441158
```

```r
# GRAPHS
plot(gridSearch$potentialValues, gridSearch$Youden)
```




```r
plot(gridSearch$potentialValues, gridSearch$Accuracy) #We are going to maintain 0.5 as partition
```




```
#value
```

PART II Time Series: US monthly unemployment rate from 2007-08 to present moment

```
options(warn=-1)

data=read.csv('C:/Users/pablo/Desktop/US_unemployment.csv')
names(data)=c('Date','Unemployment_rate')
data=data[(nrow(data)-149):nrow(data),] #we want 150 observations
nrow(data)
```

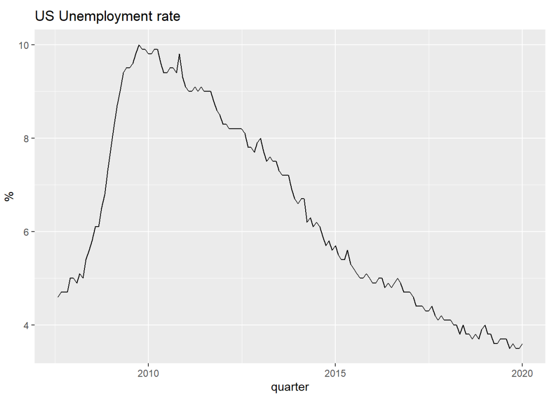
```
## [1] 150
```

```
head(data,2)
```

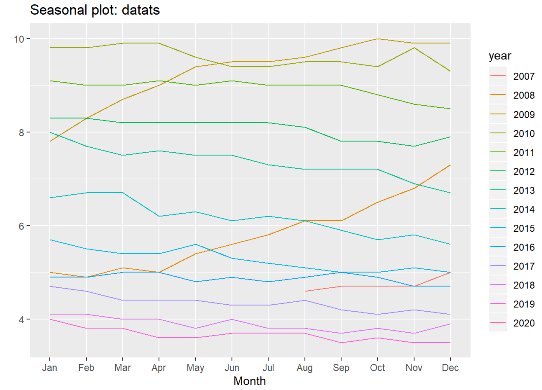
```
##           Date Unemployment_rate
## 716 2007-08-01               4.6
## 717 2007-09-01               4.7
```

Convert dataframe to timeseries object As may be inferred in ggseasonplot(), there is no clear seasonal behaviour

```
datats=ts(data[,-1], start=c(2007,8), frequency=12) #monthly frequency

autoplot(datats) + ggtitle('US Unemployment rate') + xlab('quarter') + ylab('%')
```

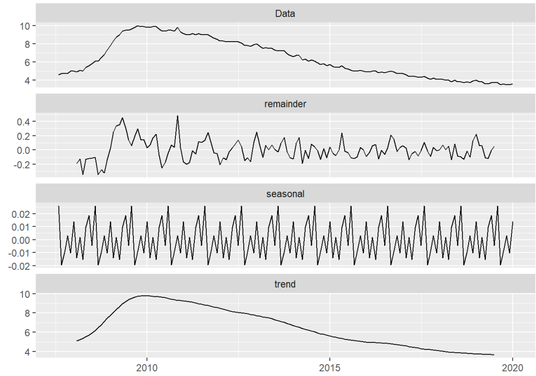



```
ggseasonplot(datats)
```



This time series is clearly non-seasonal

```
data_desc=decompose(datats)#,type=c("multiplicative"))
autoplot(data_desc)
```

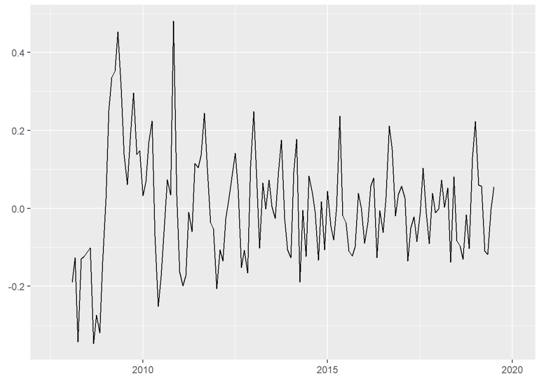



```
data_desc$figure
```

```
##  [1]  0.026060080 -0.019394465 -0.009167193  0.002954019 -0.010303556
##  [6]  0.013938868 -0.013933607  0.001691393 -0.015322496  0.009330282
## [11]  0.018705282 -0.004558607
```

Normality test for residuals: small p-values -> errors dont follow normal distribution. Thus, there is no white noise

```
autoplot(data_desc$random)
```



```
ks.test(data_desc$random,'pnorm')
```

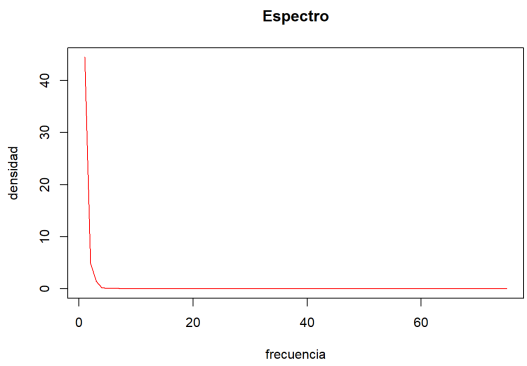
```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  data_desc$random
## D = 0.38232, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
shapiro.test(data_desc$random) #small p-value. errors dont follow a normal distribution.
```

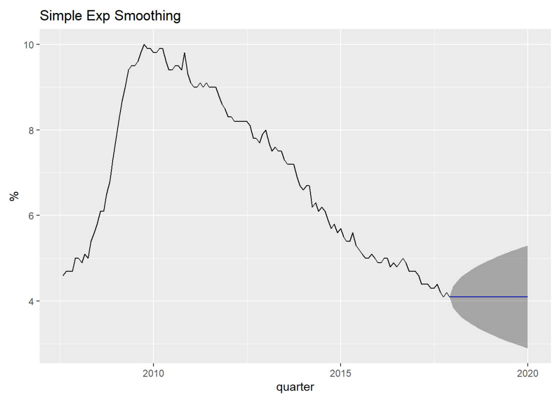
```
##
##  Shapiro-Wilk normality test
##
## data:  data_desc$random
## W = 0.97707, p-value = 0.02
```

A periodogram calculates the significance of different frequencies in time-series. Its main goal is identifying any intrinsic periodic signal.

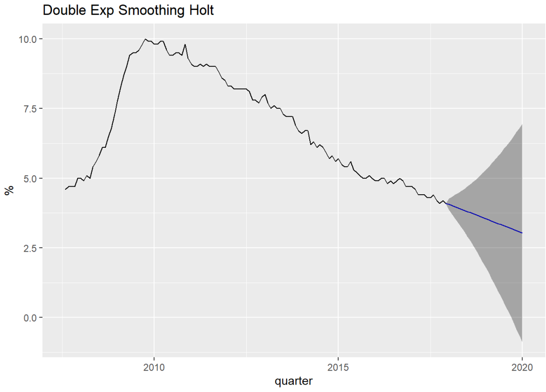
```
# peridiogram
gperiodograma(datats)
```



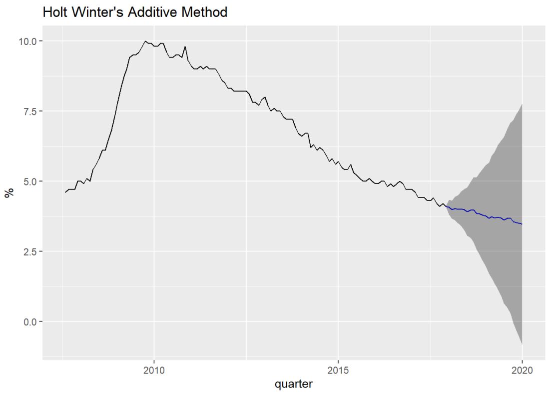

```
# TRAIN TEST SPLIT
datats_tr<-as.ts(window(x = datats, end = c(2017,12)))
datats_tst<-as.ts(window(x = datats, start = c(2018,1)))



# ### SIMPLE EXPONENTIAL SMOOTHING
datats_s1=ses(datats_tr, h=length(datats_tst))


###  DOUBLE EXPOENENTIAL SMOOTHING HOLT
datats_sh <- holt(datats_tr, h=length(datats_tst))


#### Holt-Winters

# ADDITIVE AND MULTIPLICATIVE
datats_hw_add <- hw(datats_tr, h=length(datats_tst),level = c(80, 95))
datats_hw_mul <- hw(datats_tr, h=length(datats_tst), seasonal="multiplicative",level = c(80, 95))


autoplot(datats_s1) + ggtitle('Simple Exp Smoothing') + xlab('quarter') + ylab('%')
```



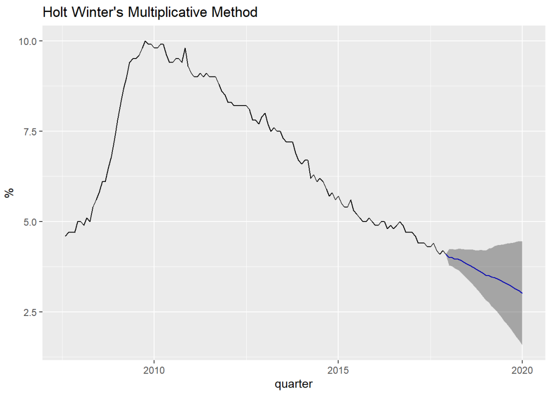

```
autoplot(datats_sh) + ggtitle("Double Exp Smoothing Holt") + xlab('quarter') + ylab('%')
```



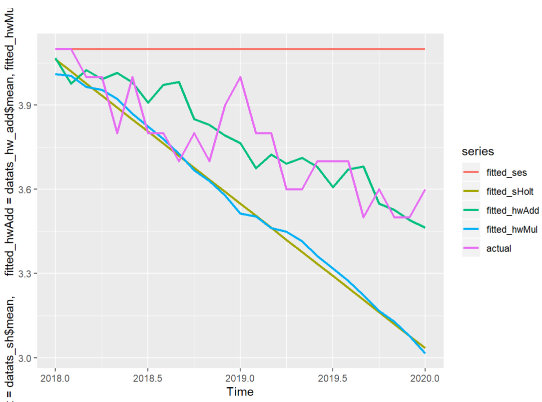

```
autoplot(datats_hw_add) + ggtitle("Holt Winter's Additive Method") + xlab('quarter') +
    ylab('%')
```




```
autoplot(datats_hw_mul) + ggtitle("Holt Winter's Multiplicative Method") + xlab('quarter') +
    ylab('%')
```

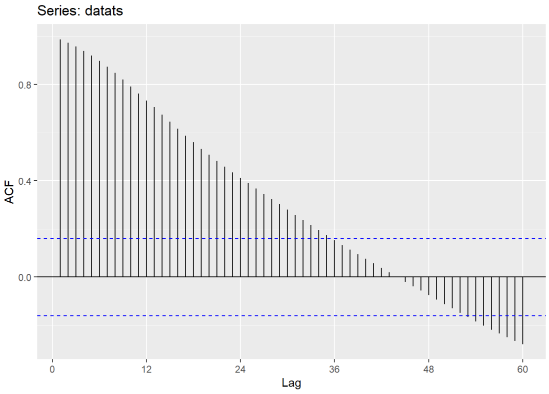



```
autoplot(cbind(fitted_ses=datats_s1$mean, fitted_sHolt=datats_sh$mean,
               fitted_hwAdd=datats_hw_add$mean, fitted_hwMul=datats_hw_mul$mean,
               actual=datats_tst), size=1)
```

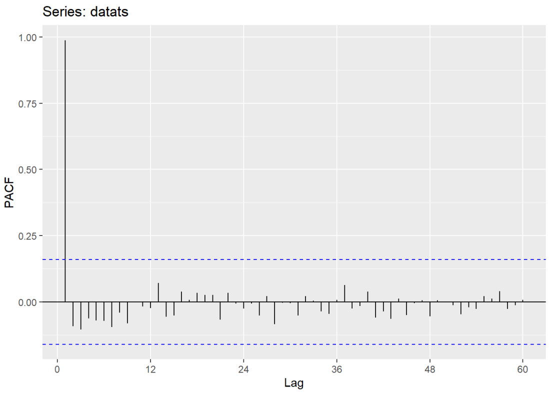


ACF & PACF confirm that datats does not have seasonal patters

```
ggAcf(datats, lag=60)
```

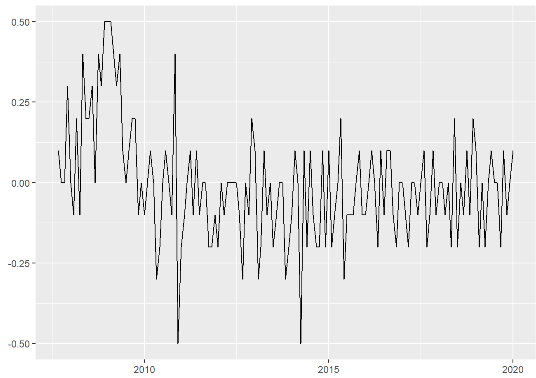



```
ggPacf(datats, lag=60)
```

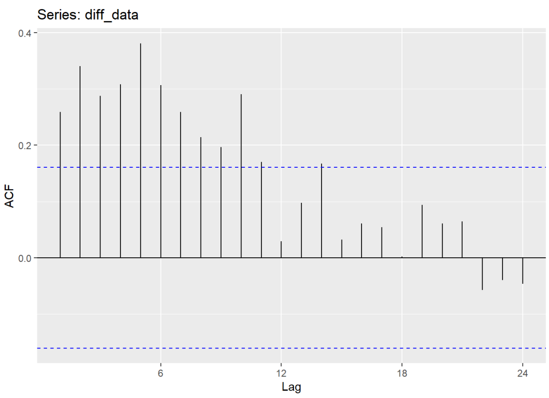


First order difference (t-1) even though there are spikes in lag 12 & 24 in PACF. We dont have any spike at that lags in ACF. * Spikes occurs at the first lags in both PACF and ACF. For that reason, my proposed ARIMA model is ARIMA(2,1,2)(0,0,0)

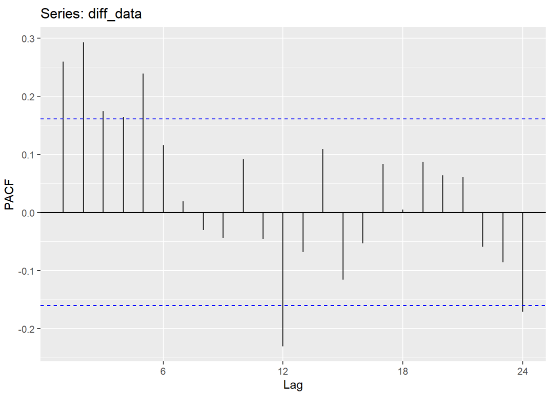
```
diff_data = datats %>% diff()
autoplot(diff_data)
```



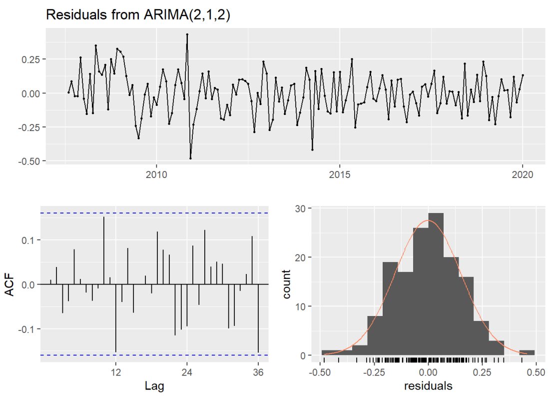

```
ggAcf(diff_data)
```



```
ggPacf(diff_data)
```



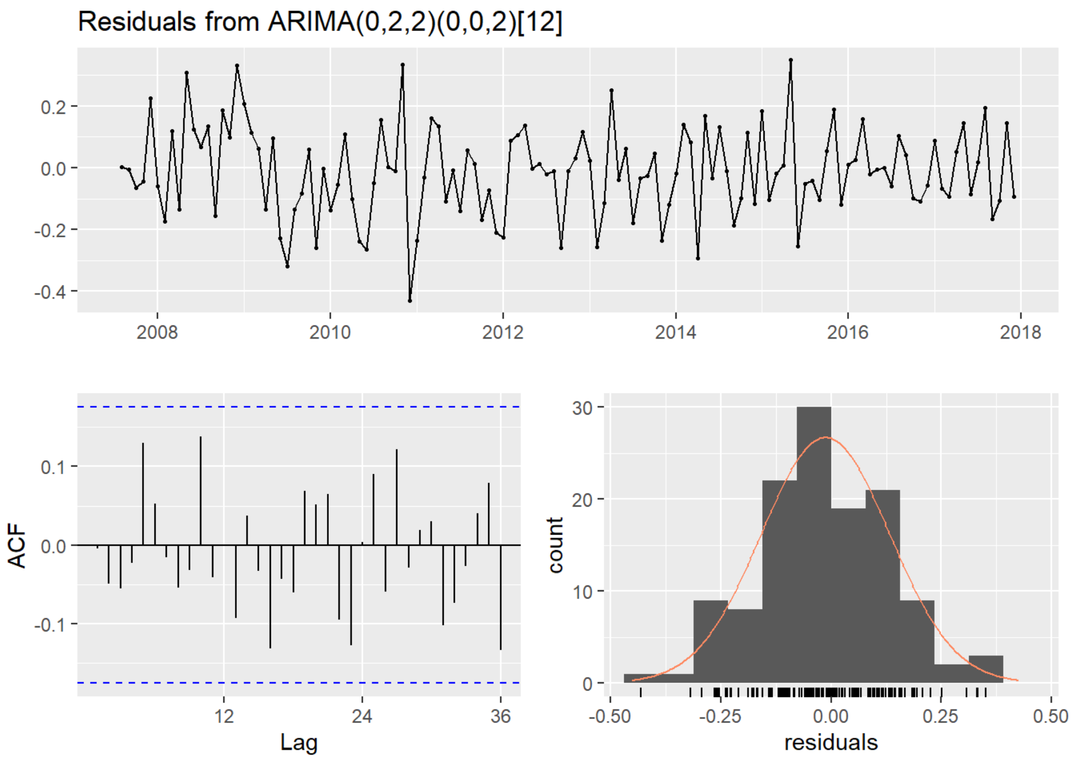

```
checkresiduals(datats %>% Arima(order=c(2,1,2), seasonal=c(0,0,0)))
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,2)
## Q* = 22.463, df = 20, p-value = 0.3159
##
## Model df: 4.   Total lags used: 24
```

LET'S CREATE AN AUTOARIMA MODEL AND CHECK IF IT OUTPERFORMS MY PROPOSED MODELS:

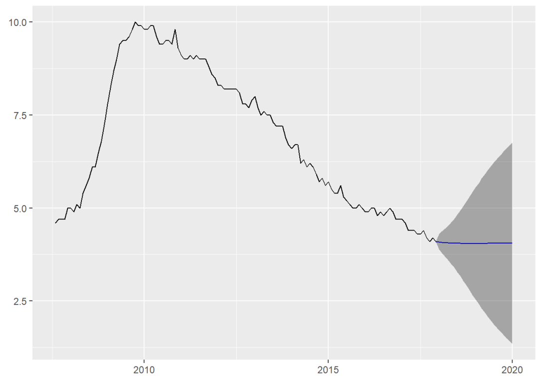
```
model1=datats_tr %>% Arima(order=c(2,1,2), seasonal=c(0,0,0))
model2=datats_tr %>% Arima(order=c(1,1,1), seasonal=c(0,0,0))

# datats_tr %>% Arima(order=c(2,1,2), seasonal=c(0,0,0))
# checkresiduals(datats_tr %>% Arima(order=c(1,1,1), seasonal=c(0,0,0)))

autoModel=datats_tr %>% auto.arima(seasonal=T)
checkresiduals(autoModel)
```



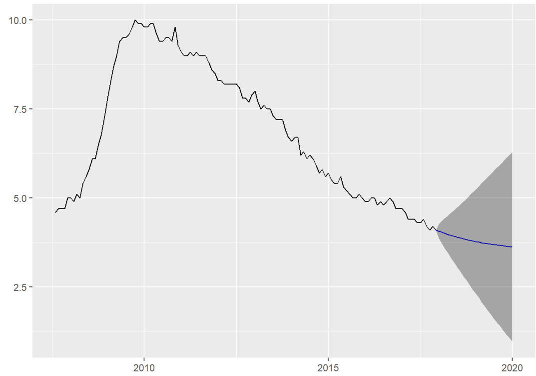

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,2,2)(0,0,2)[12]
## Q* = 17.387, df = 20, p-value = 0.6277
##
## Model df: 4.   Total lags used: 24
```

VISUALIZATION OF ARIMA MODELS' PERFORMANCE IN TEST SET

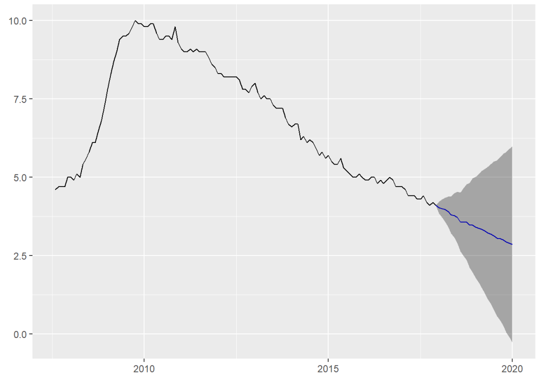
```
pred1=forecast(model1, h=length(datats_tst))
pred2=forecast(model2, h=length(datats_tst))
pred3=forecast(autoModel, h=length(datats_tst))

autoplot(pred1)
```



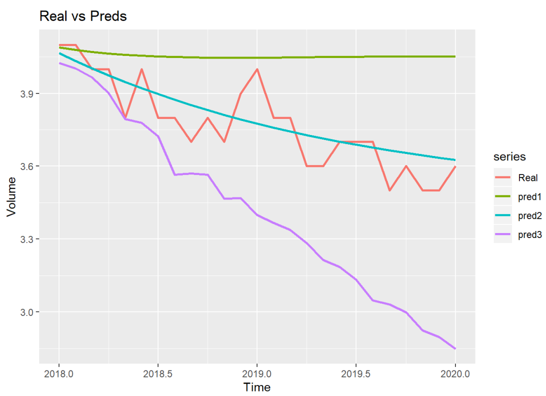

```
autoplot(pred2)
```




```
autoplot(pred3)
```




```
autoplot(cbind(Real=datats_tst, pred1=pred1$mean, pred2=pred2$mean, pred3=pred3$mean),
         size=1, ylab='Volume') + ggtitle('Real vs Preds')
```



Last Section: CLUSTERING (Dataset: Again, Spanish National elections) 10 numeric variables are going to be selected for this analysis.

```
allColumns=readRDS('Debugged_elect_dataset')
data=allColumns[,c('Autonomous_Community', 'Population', 'Left_wing_Pct', 'Right_wing_Pct',
                'ForeignersPtge', 'SurfaceArea' ,'FemalePopulationPtge', 'Others_Pct',
                'Age_19_65_pct', 'Age_over65_pct', 'totalCompanies')]
```

```
library(dplyr)
```

Let's group values by Autonomous_Community to calculate its mean and sum. This is going to be our dataset from now on

```
agg = data %>% group_by(Autonomous_Community) %>% summarise_all(list(~mean(.), ~sum(.))) %>%
         mutate_at(vars(2:ncol(data)), function(x) round(x, digits=3))

head(agg,2)
```

```
## # A tibble: 2 x 21
##   Autonomous_Comm~ Population_mean Left_wing_Pct_m~ Right_wing_Pct_~
##   <chr>                      <dbl>            <dbl>            <dbl>
## 1 Andalucía                 10858.             55.2             41.4
## 2 Aragón                     1803.             41.6             54.7
## # ... with 17 more variables: ForeignersPtge_mean <dbl>,
## #   SurfaceArea_mean <dbl>, FemalePopulationPtge_mean <dbl>,
## #   Others_Pct_mean <dbl>, Age_19_65_pct_mean <dbl>, Age_over65_pct_mean <dbl>,
## #   totalCompanies_mean <dbl>, Population_sum <dbl>, Left_wing_Pct_sum <dbl>,
## #   Right_wing_Pct_sum <dbl>, ForeignersPtge_sum <dbl>, SurfaceArea_sum <dbl>,
## #   FemalePopulationPtge_sum <dbl>, Others_Pct_sum <dbl>,
## #   Age_19_65_pct_sum <dbl>, Age_over65_pct_sum <dbl>, totalCompanies_sum <dbl>
```

Normalization of values

```
z = as.data.frame(agg[,-1])

rownames(z) = agg$Autonomous_Community

means = apply(z,2,mean) # '2' because we want to use 'apply' by columns
sd = apply(z,2,sd)

z = scale(z, means, sd)
#head(z,3)
```

Distance is going to measured by the Euclidean method It is worth mentionning that if Euclidean distance is chosen, then observations with high values of features will be clustered together. The same holds true for observations with low values of features. If we want to identify clusters of observations with the same overall profiles regardless of their magnitudes, then we should go with correlation-based distance as a dissimilarity measure.

```
distance=dist(z, method='euclidean')

#print(distance, digits=3)
```

LET'S SELECT WHICH LINKAGE METHOD IS THE BEST for K=4

```
methods <- c("single", "complete", "average", "mcquitty", "ward.D2")
st = as.data.frame(t((sapply(methods,
        function(x) cluster.stats(distance, cutree(hclust(distance, method= x), 4))))))
# head(st,2)
```

DEPENDING ON THE NUMBER OF CLUSTERS SELECTED, WE MIGHT GET DIFFERENT SOLUTIONS. THUS WE HAVE TO REPEAT THIS PROCESS, ONCE THE K-MEANS wss (elbow method) plot is done.

```
both=as.data.frame(cbind(avg.silwidth=as.numeric(st$avg.silwidth),
                         within.cluster.ss=as.numeric(st$within.cluster.ss)))
rownames(both)= rownames(st)
round(both, digits=3)
```

```
##          avg.silwidth within.cluster.ss
## single          0.167           192.386
## complete        0.173           163.601
## average         0.167           192.386
## mcquitty        0.167           192.386
## ward.D2         0.190           160.834
```

The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. BEST LINKAGE METHODS: 'ward.D2' for number of clusters = 4

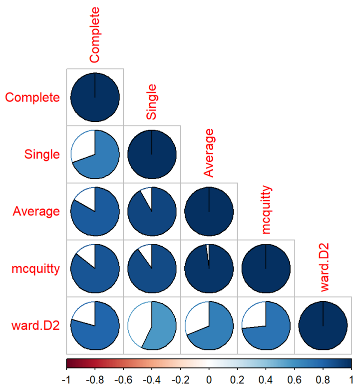
```
library(dendextend)
```

Correlation matrix of dendograms

```
# Create multiple dendrograms by chaining
dend1 <- z %>% dist %>% hclust("complete") %>% as.dendrogram
dend2 <- z %>% dist %>% hclust("single") %>% as.dendrogram
dend3 <- z %>% dist %>% hclust("average") %>% as.dendrogram
dend4 <- z %>% dist %>% hclust("mcquitty") %>% as.dendrogram
dend5 <- z %>% dist %>% hclust("ward.D2") %>% as.dendrogram
# Compute correlation matrix
dend_list <- dendlist("Complete" = dend1, "Single" = dend2,
"Average" = dend3, "mcquitty" = dend4, "ward.D2" = dend5)
cors <- cor.dendlist(dend_list)
# Print correlation matrix
round(cors, 2)
```

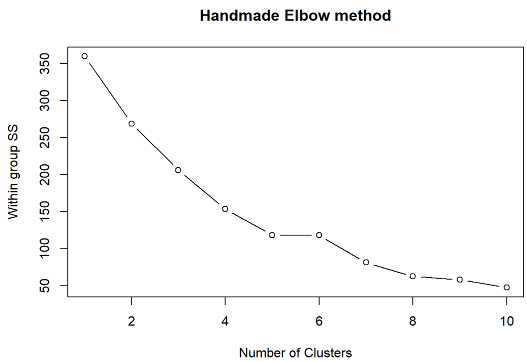
```
##           Complete Single Average mcquitty ward.D2
## Complete      1.00   0.70    0.83     0.86    0.79
## Single        0.70   1.00    0.92     0.90    0.57
## Average       0.83   0.92    1.00     0.98    0.69
## mcquitty      0.86   0.90    0.98     1.00    0.73
## ward.D2       0.79   0.57    0.69     0.73    1.00
```

```
corrplot(cors, "pie", "lower")
```

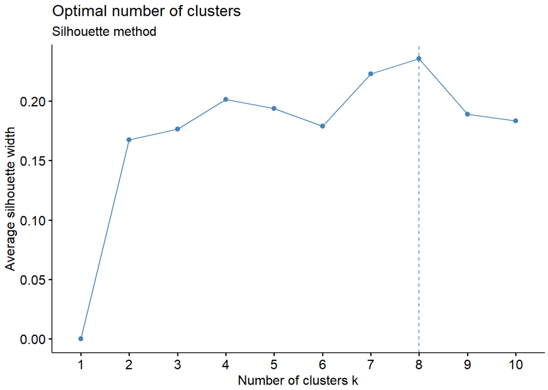


Withing group SS (elbow) method and Silhouette method

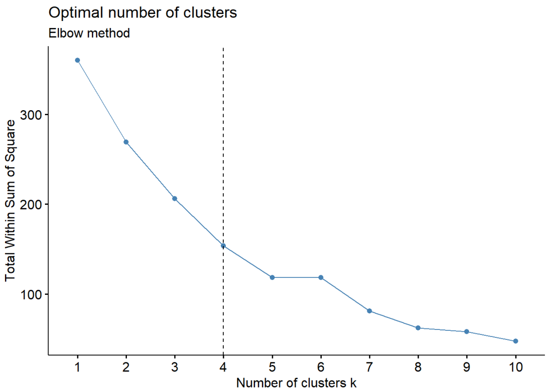
```
set.seed(123)
# Scree Plot
z=as.data.frame(z)
wss=(nrow(z)-1)*sum(apply(z,2,var))

# for (i in 1:(nrow(z)-1)) wss[i] = sum(kmeans(z, centers=i)$withinss)
# plot(1:(nrow(z)-1),wss, type='b',xlab='Number of Clusters',
#      ylab='Within group SS', main='Handmade Elbow method')

for (i in 1:10) wss[i] = sum(kmeans(z, centers=i)$withinss) #maximum number of clusters created:10
plot(1:10,wss, type='b',xlab='Number of Clusters',
     ylab='Within group SS', main='Handmade Elbow method')
```




```
fviz_nbclust(z, kmeans, method = "silhouette")+
labs(subtitle = "Silhouette method")
```




```
fviz_nbclust(z, kmeans, method = "wss") +
geom_vline(xintercept = 4, linetype = 2)+
labs(subtitle = "Elbow method")
```



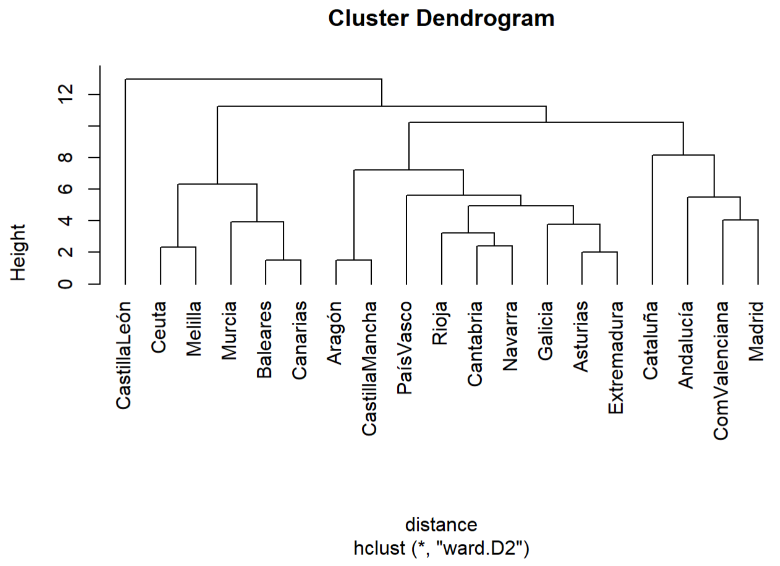
I SELECT K=4 BECAUSE IT IS THE BEST VALUE ACCORDING TO THE ‘ELBOW METHOD’. ADDITIONALY, K=4 HAS THE GREATEST AVERAGE SILHOUETTE WIDTH FOR THE FIRST 6 ‘number of clusters k’. It is worth mentioning that there are less than 20 observations and, consequently, we are not interested in creating many groups. LET’S IDENTIFY THE MOST SUITABLE LINKAGE METHOD

```
methods <- c("single", "complete", "average", "mcquitty", "ward.D2")
st = as.data.frame(t((sapply(methods,
        function(x) cluster.stats(distance, cutree(hclust(distance, method= x), 4))))))

both=as.data.frame(cbind(avg.silwidth=as.numeric(st$avg.silwidth),
                         within.cluster.ss=as.numeric(st$within.cluster.ss)))
rownames(both)= rownames(st)
round(both, digits=3)   #ward.D2 is the best linkage for k=4
```

```
##            avg.silwidth within.cluster.ss
## single            0.167           192.386
## complete          0.173           163.601
## average           0.167           192.386
## mcquitty          0.167           192.386
## ward.D2           0.190           160.834
```

Dendogram using ‘ward.D2’ as linkage method

```
plot(hclust(distance, method='ward.D2'), labels=agg$Autonomous_Community, hang=-1)
```



hybrid_k_means method with parameters we have already selected vs kmeans with k=4

```
set.seed(443)
hybrid=hkmeans(x=z, k=4, hc.metric = "euclidean", hc.method = "ward.D2")
random_Kmeans=kmeans(z,4)

paste('hybrid_withinss_mean: ', round(mean(hybrid$withinss), digits=3))
```

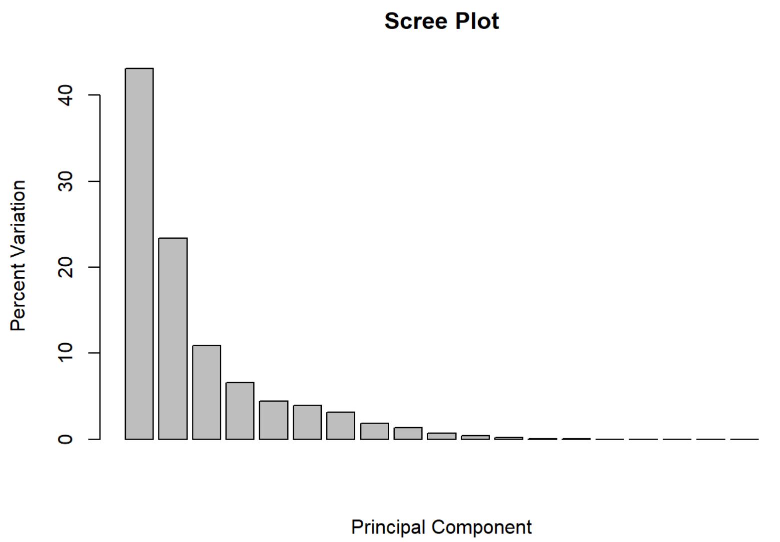
```
## [1] "hybrid_withinss_mean:  37.784"
```

```
paste('random_Kmeans_withinss_mean: ', round(mean(random_Kmeans$withinss), digits=3))
```

```
## [1] "random_Kmeans_withinss_mean:  38.378"
```

Principal Component Analysis # the first PC accounts for the most variation in the original data, and so forth. # As we want to plot a 2-D graph, we will use the first two PCs.

```
# PCA
z.pca <- prcomp(t(z))

## LET'S PLOT THE FIRST TWO PRINCIPAL COMPONENTS
#plot(z.pca$x[,1], z.pca$x[,2])


z.pca.var=z.pca$sd^2
z.pca.var.per=round(z.pca.var/sum(z.pca.var)*100, digits=3)
barplot(z.pca.var.per, main='Scree Plot', xlab='Principal Component', ylab='Percent Variation')
```

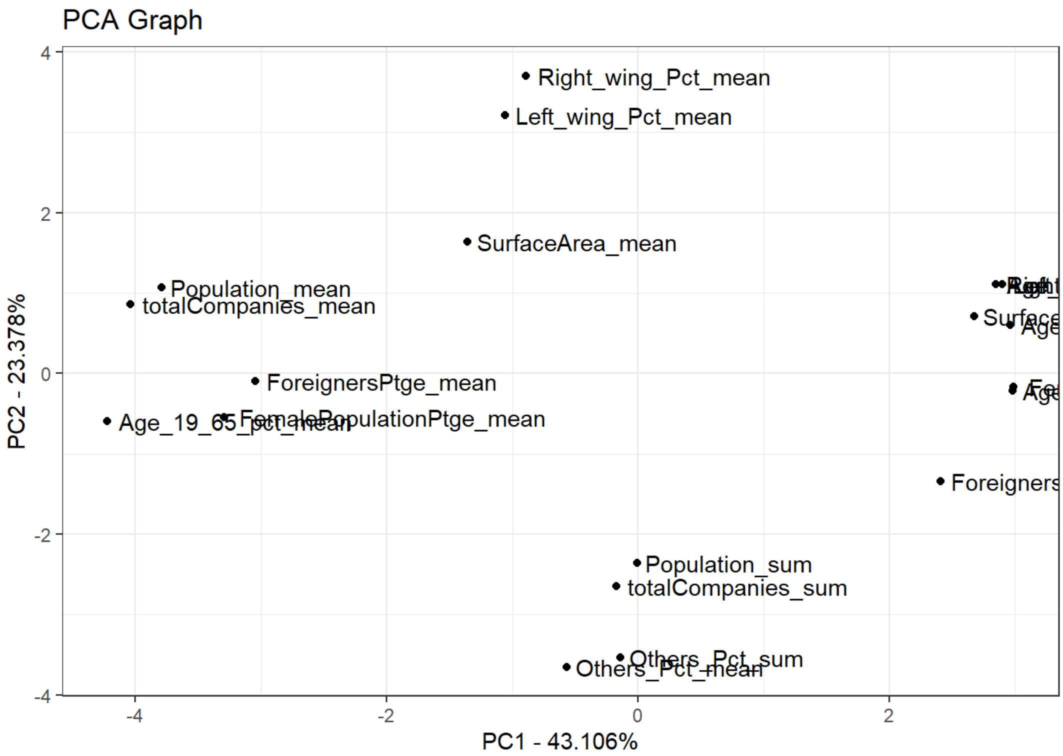


Let's check how variables are distributed under PC1 and PC2 * In the ggbiplot graph, the correlation circle has a scale from -1 to 1 and it is useful to compare the first two PCs in relation to variables

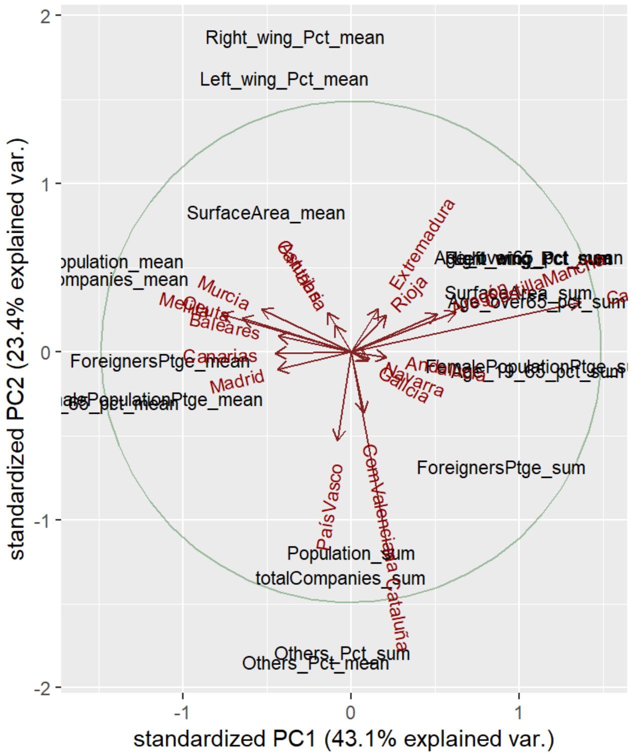
```
pca.data=data.frame(X=z.pca$x[,1], Y=z.pca$x[,2])
head(pca.data,3)
```

```
##                               X        Y
## Population_mean      -3.7879128 1.066871
## Left_wing_Pct_mean   -1.0595904 3.215874
## Right_wing_Pct_mean  -0.8916274 3.704902
```

```
library(plyr)
ggplot(pca.data, aes(x=X, y=Y, label=rownames(pca.data))) +
       geom_text(aes(label=rownames(pca.data)),hjust=-0.05, vjust=0.5) +
       xlab(paste('PC1 - ', z.pca.var.per[1], '%', sep='')) +
       ylab(paste('PC2 - ', z.pca.var.per[2], '%', sep='')) +
       theme_bw() +
       geom_point() +
       ggtitle('PCA Graph')
```



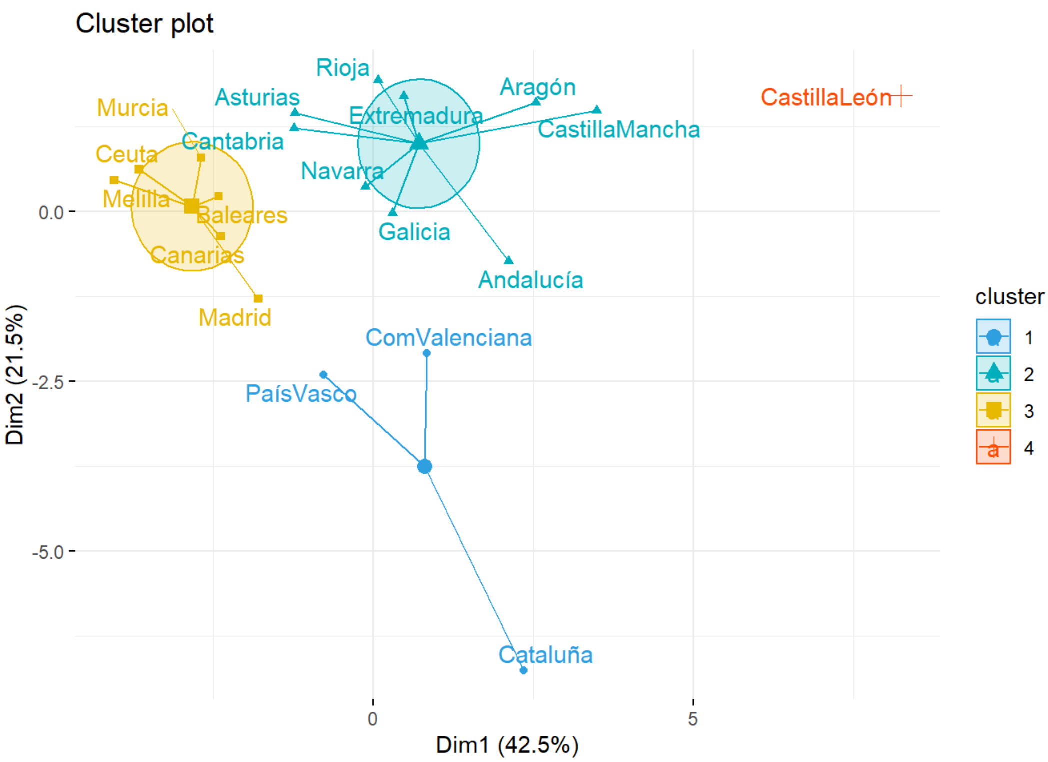

```
ggbiplot::ggbiplot(z.pca, labels = rownames(pca.data), ellipse = TRUE, circle = TRUE)
```



Cluster plot

```
hybrid=hkmeans(x=z, k=4, hc.metric = "euclidean", hc.method = "ward.D2")

fviz_cluster(hybrid, data = z,
palette = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
ellipse.type = "euclid", # Concentration ellipse
star.plot = TRUE, # Add segments from centroids to items
repel = TRUE, # Avoid label overplotting (slow)
ggtheme = theme_minimal()
)
```

```
## Too few points to calculate an ellipse
## Too few points to calculate an ellipse
```



PCA_hybrid_withinss_mean vs PCA_by_defualt_kmeans

```
PCA_z=princomp(z[,11:ncol(z)])$score[,1:2]

PCA_hybrid=hkmeans(x=PCA_z, k=4, hc.metric = "euclidean", hc.method = "ward.D2")
PCA_random_Kmeans=kmeans(PCA_z,4)

paste('PCA_hybrid_withinss_mean: ', round(mean(PCA_hybrid$withinss), digits=3))
```

```
## [1] "PCA_hybrid_withinss_mean:  5.453"
```

```
paste('PCA_random_Kmeans_withinss_mean: ', round(mean(PCA_random_Kmeans$withinss), digits=3))
```

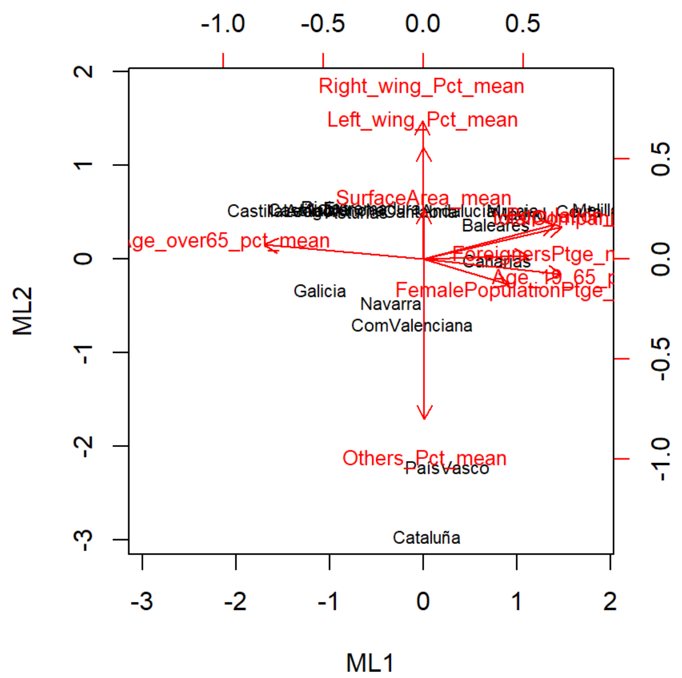
```
## [1] "PCA_random_Kmeans_withinss_mean:  5.453"
```

FACTOR ANALYSIS In Factor Analysis, variables are grouped by their correlations, this implies that all variables in a particular group will have a high correlation among themselves, but a low correlation with variables of other group(s). Here, each group is known as a factor. These factors are small in number as compared to the original dimensions of the data. However, it is important to highlight that these factors are difficult to observe and interpret.

```
z.fa=z[,1:10] #var_mean is a linear combination of var_sum. Thus, we must get rid of one
              #of them

head(z.fa[,c(1,10)],2)
```

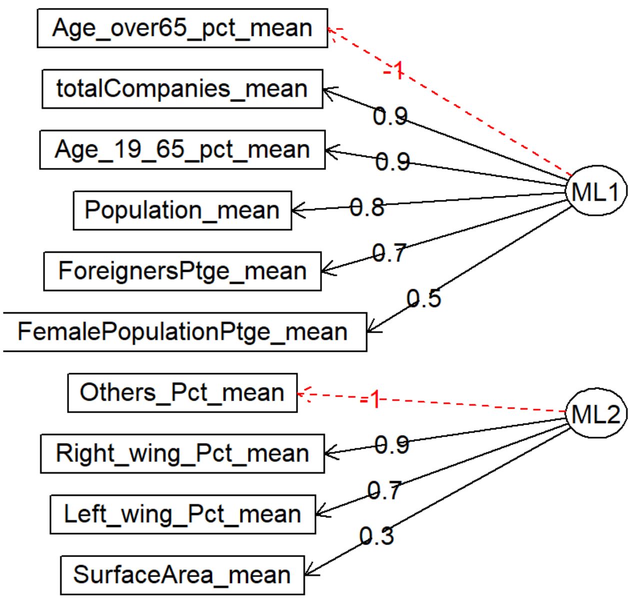
```
##            Population_mean totalCompanies_mean
## Andalucía       -0.3089806          -0.3733671
## Aragón          -0.6652714          -0.7728934
```

```
z.fa1=psych::fa(z.fa, nfactors=2, fm="ml", rotate="varimax")
FA_z=z.fa1$scores


biplot(z.fa1$scores, loadings(z.fa1), cex=c(0.7,0.8))
```




```
psych::fa.diagram(z.fa1, simple=FALSE) #
```

Factor Analysis



Let's call ML1 & ML2 with more technical names

```
head(FA_z,2)
```

```
##                  ML1      ML2
## Andalucía  0.3765881 0.523554
## Aragón    -1.1687773 0.510747
```

```
colnames(FA_z)=c('Age_Gender & right_vs_left', 'Demographics & Other_political_party')
head(FA_z,2)
```

```
##           Age_Gender & right_vs_left Demographics & Other_political_party
## Andalucía                  0.3765881                             0.523554
## Aragón                    -1.1687773                             0.510747
```

withingss_mean check

```
FA_hybrid=hkmeans(x=FA_z, k=4, hc.metric = "euclidean", hc.method = "ward.D2")
FA_random_Kmeans=kmeans(FA_z,4)

paste('FA_hybrid_withinss_mean:', round(mean(FA_hybrid$withinss), digits=3))
```

```
## [1] "FA_hybrid_withinss_mean: 1.07"
```

```
paste('FA_random_Kmeans_withinss_mean:', round(mean(FA_random_Kmeans$withinss), digits=3))
```

```
## [1] "FA_random_Kmeans_withinss_mean: 1.086"
```

cluster plot

```
fviz_cluster(FA_hybrid, data = FA_z,
palette = c("#2E9FDF", "#00AFBB", "#E7B800", "#FC4E07"),
ellipse.type = "euclid", # Concentration ellipse
star.plot = TRUE, # Add segments from centroids to items
repel = TRUE, # Avoid label overplotting (slow)
ggtheme = theme_minimal()
)
```

```
## Too few points to calculate an ellipse
## Too few points to calculate an ellipse
```