```
#packages
{
packages <- c("dplyr","caret","splitTools","ggplot2","quantmod","zoo","plyr","forecast","timetk"
,"tseries","tidyverse","furrr","reticulate","geometry","reshape","lubridate","anytime","car","ca
ret","ppcor","whitening","TSA","corrplot")

# Install packages not yet installed
installed_packages <- packages %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  install.packages(packages[!installed_packages])
}

# Packages loading
invisible(lapply(packages, library, character.only = TRUE))
}
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.6     v purrr   0.3.4
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.0     v forcats 0.5.1
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x plyr::arrange()   masks dplyr::arrange()
## x purrr::compact()  masks plyr::compact()
## x plyr::count()     masks dplyr::count()
## x plyr::failwith()  masks dplyr::failwith()
## x dplyr::filter()   masks stats::filter()
## x xts::first()      masks dplyr::first()
## x plyr::id()        masks dplyr::id()
## x dplyr::lag()      masks stats::lag()
## x xts::last()       masks dplyr::last()
## x purrr::lift()     masks caret::lift()
## x plyr::mutate()    masks dplyr::mutate()
## x plyr::rename()    masks dplyr::rename()
## x plyr::summarise() masks dplyr::summarise()
## x plyr::summarize() masks dplyr::summarize()
```

```
## Loading required package: future
```

```
##
## Attaching package: 'future'
```

```
## The following object is masked from 'package:tseries':
##
##     value
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
##
## Attaching package: 'reshape'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, smiths
```

```
## The following objects are masked from 'package:plyr':
##
##     rename, round_any
```

```
## The following object is masked from 'package:dplyr':
##
##     rename
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:reshape':
##
##     stamp
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
## Loading required package: corpcor
```

```
## Registered S3 methods overwritten by 'TSA':
##   method       from
##   fitted.Arima forecast
##   plot.Arima   forecast
```

```
##
## Attaching package: 'TSA'
```

```
## The following object is masked from 'package:readr':
##
##     spec
```

```
## The following objects are masked from 'package:stats':
##
##     acf, arima
```

```
## The following object is masked from 'package:utils':
##
##     tar
```

```
## corrplot 0.92 loaded
```

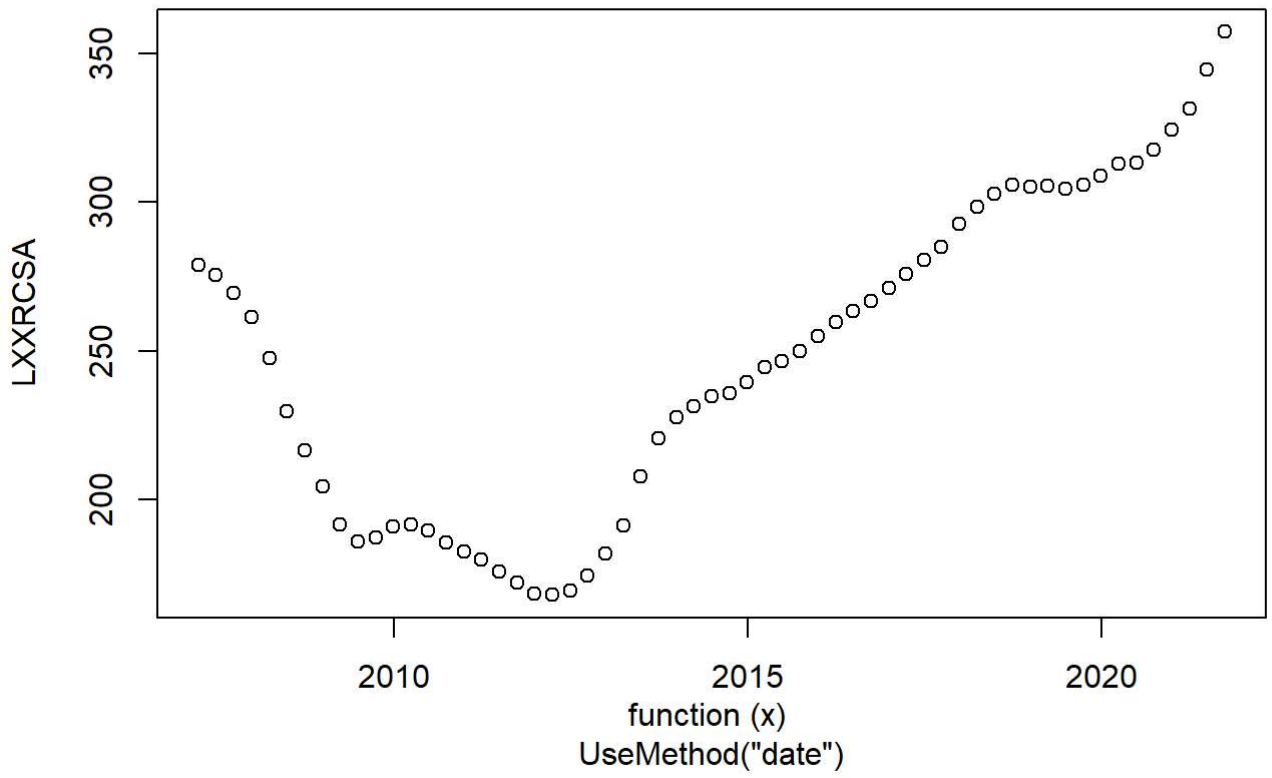
```
#data
raw <- read.csv("https://raw.githubusercontent.com/thistleknot/Python-Stock/master/data/combined
_set.csv",row.names=1,header=TRUE)
```

```
#functions
`%notin%` <- Negate(`%in%`)

PCOR <- function(x, type = c("raw", "cor")) {
          type <- match.arg(type)
          if (type == "raw") {
            x <- scale(x)
            R <- (t(x) %*% x) / (nrow(x) - 1)
          } else  {
            R <- x
          }

          ind <- unique(dim(R))
          R_inv <- ginv(R)
          ZM <- matrix(rep(0, len = (ind*ind)), nrow = ind)
          diag(ZM) <- diag(R_inv)
          D <- ginv(ZM)
          AICOV <- D %*% R_inv %*% D
          diag(ZM) <- diag(AICOV)
          D  <- ginv(sqrt(ZM))
          AICOR <- D %*% AICOV %*% D
          pcor <- AICOR
          pcor[upper.tri(pcor)] <- -pcor[upper.tri(pcor)]
          pcor[lower.tri(pcor)] <- -pcor[lower.tri(pcor)]
          dimnames(pcor) <- list(colnames(R), colnames(R))
          return(pcor)
        }

critical.r <- function( n, alpha = .05 ) {
  df <- n - 2
  critical.t <- qt(alpha/2, df, lower.tail = F)
  critical.r <- sqrt( (critical.t^2) / ( (critical.t^2) + df ) )
  return(critical.r)
}

nv_diff_sets <- function(var_of_int,dataset,f_casts)
{
  #dataset=actual_
  #f_casts = c()
  s_=sndif_[which(colnames(raw)==var_of_int)]
  d_=ndif_[which(colnames(raw)==var_of_int)]
  #-1?, based on d_?

  startRow = which(rownames(raw)==rownames(dataset[1:d_,,drop=FALSE]))

  data_ <- c(na.omit(c(dataset[,var_of_int], f_casts)))

  if(s_==0)
  {
    inv_d <- diffinv(data_,differences=d_,xi=raw[startRow,var_of_int])
  }else
  {
```

```
    inv_d <- diffinv(diffinv(data_,differences = d_, xi=raw[startRow,var_of_int]), differences =
s_,xi=raw[startRow:(startRow+season-1),var_of_int])
  }

  return(inv_d)

}
```

```
import numpy as np
import pandas as pd
from sklearn.utils import as_float_array
from sklearn.base import TransformerMixin, BaseEstimator

class ZCA(BaseEstimator, TransformerMixin):
  def __init__(self, regularization=1e-5, copy=False):
      self.regularization = regularization
      self.copy = copy
  def fit(self, X, y=None):
      X = as_float_array(X, copy=self.copy)
      self.mean_ = np.mean(X, axis=0)
      X = X - self.mean_
      sigma = np.dot(X.T, X) / (X.shape[0] - 1)
      U, S, V = np.linalg.svd(sigma)
      tmp = np.dot(U, np.diag(1 / np.sqrt(S + self.regularization)))
      self.components_ = np.dot(tmp, U.T)
      return self
  def transform(self, X):
      X_transformed = X - self.mean_
      X_transformed = np.dot(X_transformed, self.components_.T)
      return X_transformed
```

```
#vars
f = periodicity(as.Date(anytime(rownames(raw))))

options(precision=2)
#options("scipen"=100, "digits"=4)

season = switch(
  f$scale,
  "monthly"= 12,
  "daily"= 252,
  "quarterly"= 4,
  "weekly"= 52,
)

var_of_int <- sample(colnames(raw),1)
#var_of_int <- "MSPUS"
#var_of_int <- "GDPC1"
var_of_int <- "LXXRCSA"
#var_of_int <- "AWHAETP"

dm <- as.data.frame(cbind(anydate(rownames(raw)),raw[,var_of_int,drop=FALSE]))
colnames(dm) <- c(date,var_of_int)
plot(dm)
```



```
#test python functions
```

```
#differencing

sndif_ <- unlist(lapply(1:(length(colnames(raw))),function(n)
  {
    d_ <- nsdiffs(ts(raw[,n],frequency=season))
    return(d_)
  }))

combo_s <- do.call(cbind,lapply(1:length(sndif_), function(d)
  {
    if(sndif_[d]*season == 0)
    {
      temp <- raw[,d,drop=FALSE]
    }else
    {
      temp <- raw[,d,drop=FALSE]
      for(dif in 1:sndif_[d])
      {
        temp <- temp-dplyr::lag(temp,1*season)
      }
    }
  return(temp)
  }))

ndif_ <- c()

ndif_ <- unlist(lapply(1:(length(colnames(combo_s))),function(n)
{
  d_ <- ndiffs(combo_s[,n])
  #min 1 to ensure I'm being consistent (i.e. measuring rate of change between quarters). Note:
 ndiffs == na's return 0 so I handle na's in combo_d
  if(d_ == 0){d_ = 1}
  return(d_)
}))

#which(colnames(raw)==var_of_int)
combo_d <- do.call(cbind,lapply(1:length(ndif_),function(d)
{#no if check for 0 because at a minimum I want one difference
  #d=1
  #print(d)
  s=sndif_[d]
  d_=ndif_[d]

  if(s>0)
  {
    og <- as.vector(diff(zoo(raw[,colnames(raw)[d],drop=TRUE]),lag=season,differences=s,na.pad=T
RUE))
    diffset <- as.vector(diff(zoo(og),differences=d_,na.pad=TRUE))
  }else
  {
    diffset <- as.vector(diff(zoo(raw[,colnames(raw)[d],drop=TRUE]),differences=d_,na.pad=TRUE))
  }
```

```
  properSet <- as.data.frame(diffset)
  colnames(properSet) <- colnames(raw[,colnames(raw)[d],drop=FALSE])
  rownames(properSet) <- rownames(raw)
  #View(properSet)
  return(properSet)
}))

which(((ndif_==2)*(sndif_==1))==1)
```

```
## integer(0)
```

```
colnames(raw)[which(((ndif_==2)*(sndif_==0))==1)[1]]
```

```
## [1] "ASPUS"
```

```
colnames(raw)[which(((ndif_==1)*(sndif_==1))==1)[1]]
```

```
## [1] "BUSAPPWNSACA"
```

```
colnames(raw)[which(((ndif_==1)*(sndif_==0))==1)[1]]
```

```
## [1] "AWHAETP"
```

```
#multiplicative (returns) to log

if(FALSE)
#CFNAIDIFF has negative initial values which make calculating returns impossible
{

  combo_s_m <- do.call(cbind,lapply(1:length(sndif_),function(d)
    {
      if(sndif_[d]*season == 0)
      {
        temp <- raw[,d,drop=FALSE]
      }else
      {
        temp <- raw[,d,drop=FALSE]
        for (sdif in 1:sndif_[d])
        {

          temp <- (1+((temp-dplyr::lag(temp,1*season))/dplyr::lag(temp,1*season)))
        }
      }
    return(temp)
    }))

  combo_d_m <- do.call(cbind,lapply(1:length(ndif_),function(d)
  {
    #d=1
      temp <- combo_s_m[,d,drop=FALSE]
      for (dif in 1:ndif_[d])
      {
        temp <- (1+((temp-dplyr::lag(temp,1))/dplyr::lag(temp,1)))

        temp <- as.data.frame(lapply(temp,function(x){ifelse(is.nan(x),return(0),return(x))}))
        rownames(temp) <- rownames(combo_s_m[,d,drop=FALSE])
      }
      return(temp)
  }))

}
```

```
#optimal lagged correlation
# Example usage: Critical correlation coefficient at sample size of n = 100

names <- c()
lags_ <- c()
'
numZero <- colSums(combo_d == 0, na.rm = T)

numNaN <- sapply(combo_d_m, function(x) sum(is.nan(x)))

names(which(numNaN>0))

ndif_[which(colnames(raw)=="INTDSRUSM193N")]
sndif_[which(colnames(raw)=="INTDSRUSM193N")]

limit = (nrow(raw)/2)
drops = names(which(numZero>=limit))

#lapply(combo_d, function(x){ length(which(x==0))/length(x)})

combo_ <- na.omit(dplyr::select(combo_d, -c(drops)))
'
```

```
## [1] "\nnumZero <- colSums(combo_d == 0, na.rm = T)\n\nnumNaN <- sapply(combo_d_m, function(x)
sum(is.nan(x)))\n\nnames(which(numNaN>0))\n\nndif_[which(colnames(raw)==\"INTDSRUSM193N\")]\nsnd
if_[which(colnames(raw)==\"INTDSRUSM193N\")]\n\nlimit = (nrow(raw)/2)\ndrops = names(which(numZe
ro>=limit))\n\n#lapply(combo_d, function(x){ length(which(x==0))/length(x)})\n\ncombo_ <- na.omi
t(dplyr::select(combo_d, -c(drops)))\n"
```

```
combo_ <- combo_d
{
  training <- combo_[1:floor(nrow(combo_)*.7),]
  #validation <- training-round(nrow(combo_)*.7*.3,0)
  holdout <- combo_[(nrow(training)+1):nrow(combo_),]
}


training_nona <- na.omit(training)

#lags_[which(colnames(raw)==var_of_int)]
#winners

#lags_[which(colnames(raw)==winners)]

#View(newDF)
for(c in 1:(length(colnames(training_nona))))
{#c=90

  temp <- na.omit(training)

  ccf1 <- ccf(training_nona[,var_of_int,drop=FALSE],training_nona[,c], lag.max = season, correla
tion=TRUE, plot=FALSE)

  crit = critical.r(ccf1$n.used)

  #upperCI <- qnorm((1+0.95)/2)/sqrt(ccf1$n.used)
  #lowerCI <- -qnorm((1+0.95)/2)/sqrt(ccf1$n.used)

  #ind.max <- which(abs(ccf1$acf)==max(abs(ccf1$acf)))
  candidates <- which((abs(ccf1$acf)<=crit) & (ccf1$lag >= 1))
  if(length(candidates)!=0)
  {
    which_max <- which.max(abs(ccf1$acf[candidates]))
    ind.max <- max(abs(ccf1$acf[candidates][which_max]))
    max.cor <- ccf1$acf[candidates][which_max]
    lag.opt <- ccf1$lag[candidates][which_max]
    names <- c(names,(colnames(training_nona)[c]))
    #print(colnames(training_nona)[c])
    #print(lag.opt)
    #print(max.cor)
    lags_ <- c(lags_,lag.opt)
    #print(2 * (1 - pnorm(abs(max.cor), mean = 0, sd = 1/sqrt(ccf1$n.used))))
  }

}

#needed if lag.opt>=0
'
if(min(lags_)==0)
{
```

```
}
'
```

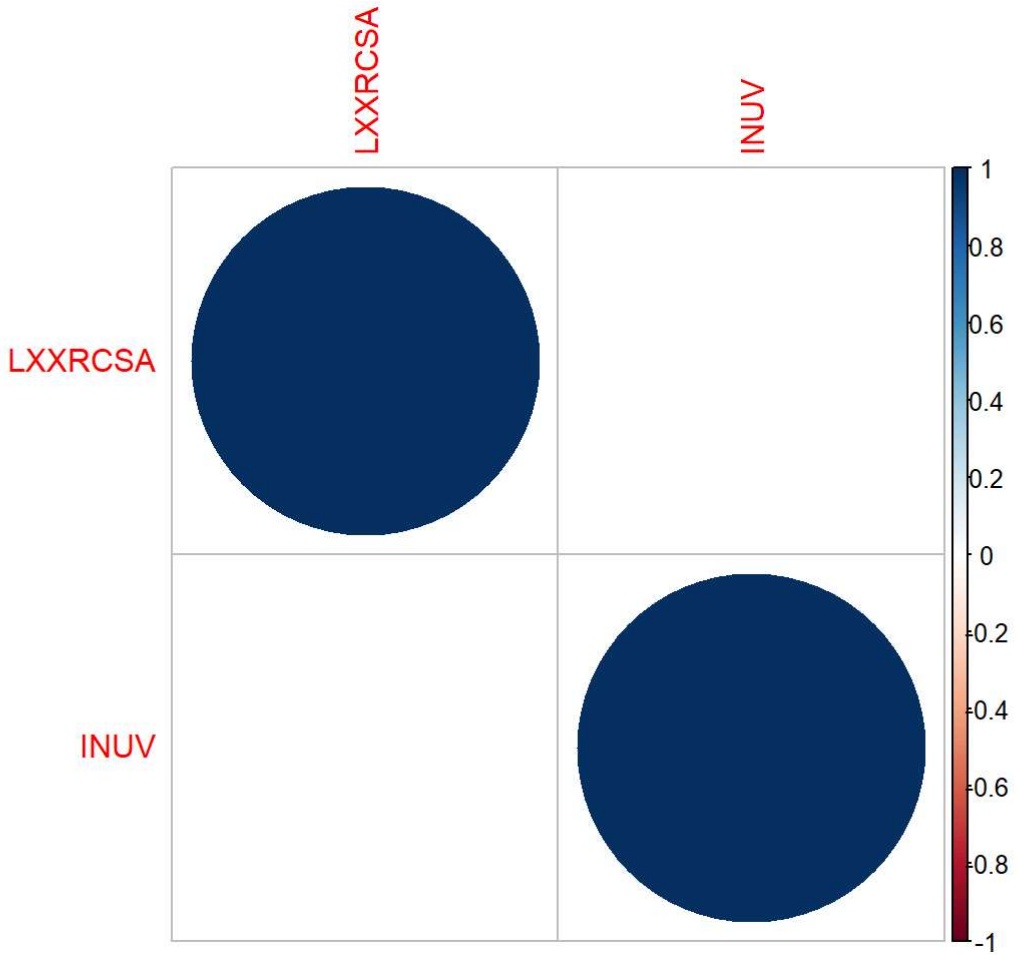
```
## [1] "\nif(min(lags_)==0)\n{\n  \n}\n"
```

```
#ZCA backstep filter

newDF_t <- do.call(cbind,lapply(1:length(names), function(n)
{#ensure var_of_int isn't differences
  if(names[n]==var_of_int)
  {
    temp <- training[,var_of_int,drop=FALSE]
  }else
  {
  ts_ <- stats::lag(x=ts(training[,names[n],drop=FALSE],frequency=season),k=lags_[n])
  temp <- as.data.frame(matrix(ts_))

  rownames(temp) <- anydate(rownames(training)[1]) %m+% months(matrix(time(ts_))*12)
  colnames(temp) <- names[n]
  }
  return(temp)
  #newDF_t <- cbind(newDF_t,temp)

}))

sig_table = matrix(0, ncol=ncol(newDF_t))
colnames(sig_table) <- colnames(newDF_t)
signs_table = matrix(0, ncol=ncol(newDF_t))
colnames(signs_table) <- colnames(newDF_t)

p_threshold = .05

iteration=0

dat <- 1:10
n=length(dat)

exclude <- c()

#crit <- critical.r(nrow(set_), .05)

max_pvalue = 1

subset = na.omit(newDF_t[,c(colnames(newDF_t) %notin% c(exclude))])

zca <- py$ZCA()

subset_w <- cbind(subset[,var_of_int,drop=FALSE],as.data.frame(zca$fit_transform(as.matrix(subse
t[,c(colnames(newDF_t) %notin% c(var_of_int))]))))

colnames(subset_w) <- colnames(subset)
rownames(subset_w) <- rownames(subset)

while(max_pvalue>=p_threshold)
{
  p_values  <- (2 * (1 - pnorm(abs(cor(subset_w)[,var_of_int,drop=FALSE]), mean = 0, sd = 1/sqrt
(nrow(subset_w)))))
```

```
  #p_values  <- (2 * (1 - pnorm(abs(PCOR(subset)[,var_of_int,drop=FALSE]), mean = 0, sd = 1/sqrt
(nrow(subset)))))

  #print(grep(var_of_int,rownames(p_values)))

  #pcor(subset, method = c("spearman"))$p.value[,var_of_int,drop=FALSE]

  max_pname = rownames(p_values)[which.max(p_values)]
  max_pvalue = p_values[max_pname,]

  if (max_pvalue >= p_threshold)
  {
    #print(max_pvalue)
    #print(max_pname)
    temp <- dplyr::select(subset_w,-c(max_pname))
    #temp <- dplyr::select(subset,-c(max_pname))
    if(ncol(temp)==1)
    {
      break
    }
    else
    {
      zca <- py$ZCA()
      temp_ <- cbind(subset_w[,var_of_int,drop=FALSE],as.data.frame(zca$fit_transform(as.matrix
(temp[,c(colnames(temp) %notin% c(var_of_int))])))
      #temp_ <- cbind(subset_w[,var_of_int,drop=FALSE],as.data.frame(py$white(as.matrix(temp[,c
(colnames(temp) %notin% c(var_of_int))]))))
      #temp_ <- cbind(subset[,var_of_int,drop=FALSE],temp[,c(colnames(temp) %notin% c(var_of_in
t))])

  colnames(temp_) <- colnames(temp)
  rownames(temp_) <- rownames(temp)
  subset_w <- temp_
  #subset <- temp_
    }
  }
}
```

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

```
winners = rownames(p_values)[rownames(p_values) %notin% c(var_of_int)]
sig_table = sig_table + as.integer(colnames(newDF_t) %in% winners)

corrplot(cor(subset_w[,c(var_of_int,winners)]))
```

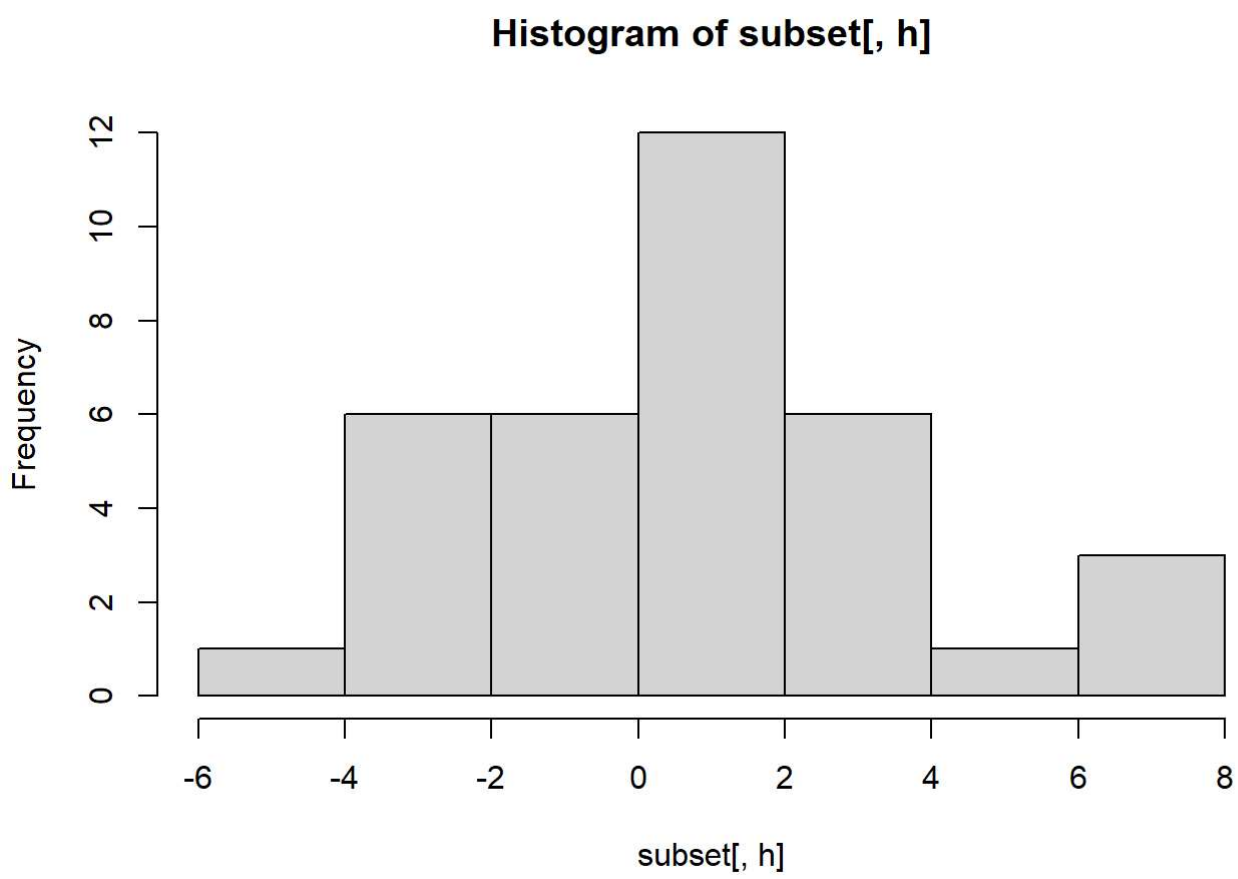


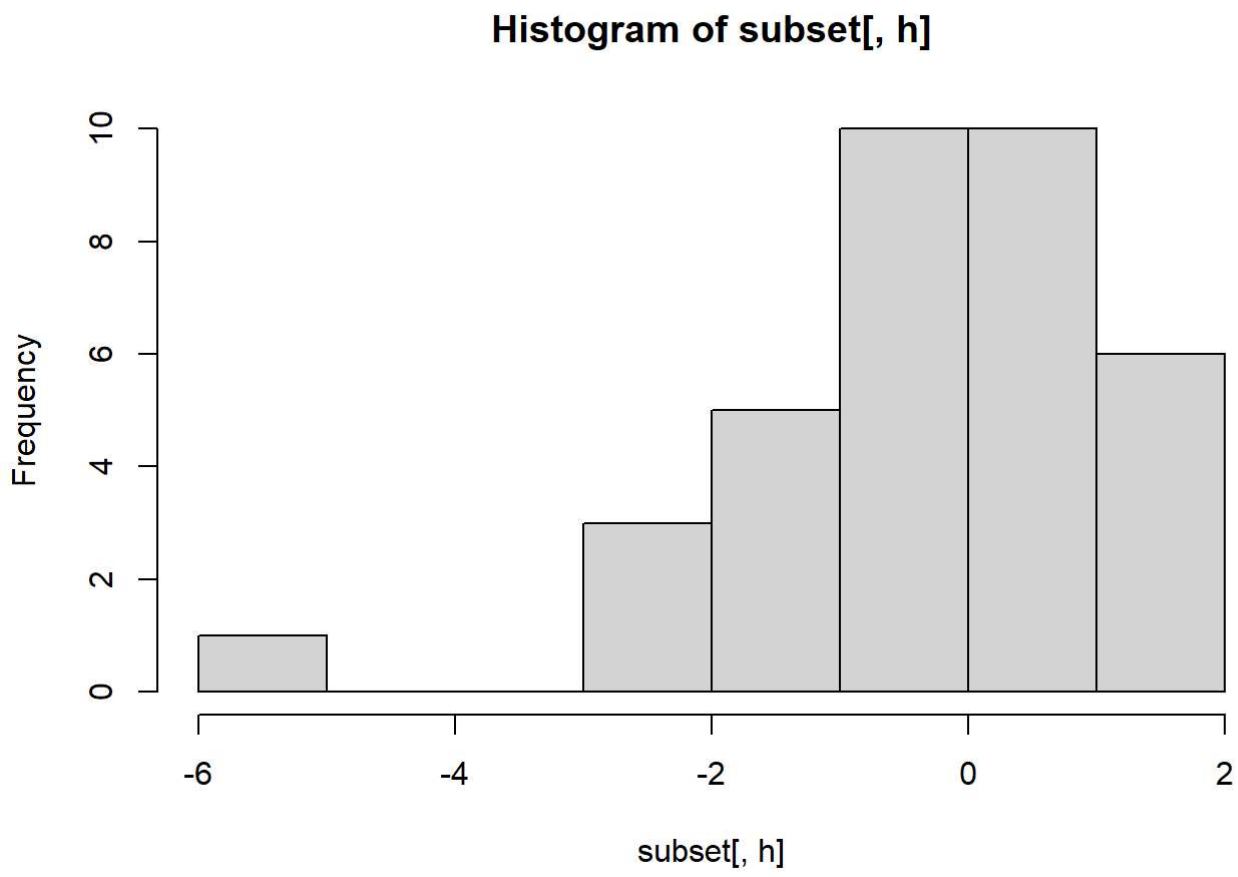

```
#remove y (for now until I figure out how to handle the name change)
winners <- winners[winners %notin% var_of_int]
```

```
#min of 1

differences <- as.data.frame(rbind(c(lags_[which(match(names,c(var_of_int,winners))>0)]),sndif_
[which(match(colnames(raw),c(var_of_int,winners))>0)],ndif_[which(match(colnames(raw),c(var_of_i
nt,winners))>0)]))

colnames(differences) <- c(var_of_int,winners)

rownames(differences) <- c("lags","season","nonseason")
print(differences)
```

```
##           LXXRCSA INUV
## lags            2    3
## season          0    0
## nonseason       2    1
```

```
differences["lags",var_of_int] <- 0

for(h in c(var_of_int,winners))
{
  hist(subset[,h])
}
```


Histogram of subset[, h]
Frequency
0
2
4
6
8
10
12
-6
-4
-2
0
2
4
6
8
subset[, h]


Histogram of subset[, h]
Frequency
0
2
4
6
8
10
-6
-4
-2
0
2
subset[, h]

```
#min of 1

#edge <- max(differences["lags",2:ncol(differences),])
edge <- season-1

#ts_ <- ts(raw[,var_of_int,drop=FALSE],frequency=season, start = c(year(rownames(combo_)[1]),(mo
nth(rownames(combo_)[1])/12)*4))

ts_ <- stats::lag(x=ts(combo_[,var_of_int,drop=FALSE],frequency=season, start = c(year(rownames
(combo_)[1]),(month(rownames(combo_)[1])/12)*4), ),k=-edge)

d_s <- c(anydate(rownames(raw)[1:edge]),anydate(rownames(raw)[1]) %m+% months((index(ts_)-year(a
nydate(rownames(raw)[1])))*12))

df_ <- as.data.frame(matrix(NA,nrow=length(d_s)))

rownames(df_) <- anydate(d_s)
#View(df_)

#newDF <- merge(df_,combo_[,var_of_int,drop=FALSE],by=0,all.x = TRUE)
#rownames(newDF) <- newDF$Row.names

#newDF <- newDF[,var_of_int,drop=FALSE]

which(colnames(raw)==winners)
```

```
## [1] 184
```

```
#for (n in 1:length(names))
newDF <- do.call(cbind,lapply(1:length(names),function(n)
{#n=182
  if(names[n]==var_of_int)
  {
    temp <- merge(df_,combo_[,var_of_int,drop=FALSE],by=0,all.x = TRUE)
    temp = temp[,3,drop=FALSE]
    colnames(temp) <- names[n]
    rownames(temp) <- temp$Row.names
  } else
  {
    #n=1
    #print(n)
    ts_i <- stats::lag(x=ts(combo_[,names[n],drop=FALSE],frequency=season,start = c(year(rowname
s(combo_)[1]),(month(rownames(combo_)[1])/12)*4)),k=-(lags_[n]))
    temp <- as.data.frame(matrix(ts_i))
    colnames(temp) <- names[n]

    qtrs = index(ts_i)-year(as.Date(yearqtr(index(ts_i)[1])))

    #rownames(temp) <- anydate(rownames(raw)[1]) %m+% months(qtrs*12)

    rownames(temp) <- anydate(rownames(raw[1,,drop=FALSE])) %m+% months(qtrs*12)

    temp <- merge(df_,temp,by=0,all.x = TRUE)

    #rownames(temp) <- #anydate(rownames(df_i)[1]) %m+% months(matrix(time(ts_))*12)
    #colnames(temp) <- names[n]

    #temp <- merge(df_i,,by=0,all.x = TRUE)
    #rownames(temp) <- temp$Row.names

    #temp <- temp[,names[n],drop=FALSE]
  }
  return(temp[,names[n],drop=FALSE])
  #newDF <- cbind(newDF,temp)

}))
newDF[,winners,drop=FALSE]
```

```
##           INUV
## 1           NA
## 2           NA
## 3           NA
## 4           NA
## 5   0.42884087
## 6   0.40340392
## 7   1.26347435
## 8   1.09610069
## 9   1.94822935
## 10 -0.58018366
## 11 -2.95837361
## 12 -5.15270989
## 13 -1.41299517
## 14 -0.36950975
## 15  1.54767383
## 16  1.42909752
## 17  1.77323688
## 18  0.18217593
## 19  0.40227978
## 20  0.31824765
## 21 -0.04469043
## 22 -0.08000123
## 23  0.13231389
## 24 -2.12457932
## 25 -1.67610944
## 26 -0.46613255
## 27 -0.68620438
## 28  1.13330776
## 29 -0.07353334
## 30  0.66547288
## 31  0.16639137
## 32  0.69483598
## 33  1.03227871
## 34  0.12471175
## 35 -0.83296749
## 36 -1.77316712
## 37 -1.66296982
## 38 -0.40312192
## 39 -2.30297115
## 40 -1.08938997
## 41  1.05664406
## 42 -0.09600219
## 43  0.09387311
## 44  0.56331389
## 45  0.29634921
## 46 -0.05750447
## 47  0.86342609
## 48  0.91449166
## 49 -1.59897711
## 50 -0.71943136
## 51 -0.27846325
```

```
## 52 -0.12738406
## 53 -0.38823682
## 54 -1.25861010
## 55 -0.29204221
## 56 -0.10532289
## 57  0.19308521
## 58  0.03876225
## 59  0.48885349
## 60  0.67891841
## 61  1.30707148
## 62 -0.04993505
```

```
rownames(newDF) <- d_s

#View(newDF)
#newDF[,var_of_int,drop=FALSE] <- combo_[,var_of_int,drop=FALSE]
```

```
#model construction and holdout analysis

horizon = min(differences["lags",2:ncol(differences),])

n=floor(nrow(combo_)*.7)

#View(newDF[,c(var_of_int,winners)])

folds = createTimeSlices(head(rownames(newDF),-(season-1)), initialWindow = n, horizon=horizon,
 fixedWindow = T)
length(folds$train)
```

```
## [1] 16
```

```
models <- lapply(1:length(folds$train),function(f)
{
  #f=1
  train_ <- newDF[folds$train[f][[1]],,drop=FALSE]

    m <- list(
      auto.arima(train_[,var_of_int,drop=FALSE]),
      NA,#auto.arima(lm_$residuals)
      NA,#arfima(lm_$residuals)
      NA,#auto.arima(train_[,var_of_int,drop=FALSE],xreg=lm_$residuals)
      #NA/NaN argument
      #arfima(train_[,var_of_int,drop=FALSE],xreg=as.matrix(train_[,winners]),estim=c("ls")),
      ets(ts(train_[,var_of_int,drop=FALSE],frequency=season)),
      auto.arima(train_[,var_of_int,drop=FALSE],xreg=as.matrix(structure(data.frame(const=matrix
(1,nrow=nrow(train_)),train_[,winners]),names=c("constant",winners))))
    )
  return(m)
})
```

```
## Warning in ets(ts(train_[, var_of_int, drop = FALSE], frequency = season)):
## Missing values encountered. Using longest contiguous portion of time series

## Warning in ets(ts(train_[, var_of_int, drop = FALSE], frequency = season)):
## Missing values encountered. Using longest contiguous portion of time series
```

```
forecasts <- lapply(1:length(folds$train),function(f)
{#f=1
  #print(f)
  train_ <- newDF[folds$train[f][[1]],,drop=FALSE]

  test_ <- newDF[folds$test[f][[1]],,drop=FALSE]

  fore_ <- list(
  {
    f_0 <- as.data.frame(forecast(models[[f]][[1]],h=horizon))
    rownames(f_0) <- rownames(test_)
    f_0
  },
  if(FALSE)
  {
    #arima of lm residuals
    f_1 <- as.data.frame(forecast(models[[f]][[2]],h=horizon))
    f_1a <- f_1 + t(predict(lm_,newdata=test_[1:horizon,winners,drop=FALSE]))
    rownames(f_1a) <- rownames(test_)
    f_1a
  }else
  {
    NA
  }
  ,
  if(FALSE)
  {
    #arfima of residuals
    f_2 <- as.data.frame(forecast(models[[f]][[3]],h=horizon))
    f_2a <- f_2 + t(predict(lm_,newdata=test_[1:horizon,winners,drop=FALSE]))
    rownames(f_2a) <- rownames(test_)
    f_2a
  }else
  {
    NA
  },
  if(FALSE)
  {
    #arima exogenous
    f_3_lm_r <- forecast(lm_, test_[1:horizon,winners,drop=FALSE], h=horizon)
    #test_4 <- arfima(newDF_t[,var_of_int,drop=FALSE],xreg=lm_$residuals,estim=c("mle"))
    f_3a <- as.data.frame(forecast(models[[f]][[4]],h=horizon,xreg=f_3_lm_r$mean))
    #lapply(c(2,3,4),function(x){f_3_lm_r[names(f_3_lm_r)[x]]})
    #rownames(f_3a)
    rownames(f_3a) <- rownames(test_)
    f_3a
  }else
    {
      NA
    }
  ,
  {
```

```
    f_4 <- as.data.frame(forecast(models[[f]][[5]],h=horizon))
    rownames(f_4) <- rownames(test_)
    f_4
  },
  {
    #proper arimax
    f_5 <- as.data.frame(forecast(models[[f]][[6]],xreg=as.matrix(structure(data.frame(const=mat
rix(1,nrow=nrow(test_)),test_[,winners]),names=c("constant",winners)))))
    rownames(f_5) <- rownames(test_)
    f_5
  })
  return(fore_)

})

errors <- lapply(1:length(folds$train),function(f)
{#f=16
  test_ <- newDF[folds$test[f][[1]],var_of_int,drop=FALSE]
  e_ <- list(#f=1
    mean(abs(forecasts[[f]][[1]]$`Point Forecast`-t(test_))),
    NA,#mean(abs(forecasts[[2]]$`Point Forecast`-t(test_)))
    NA,#mean(abs(forecasts[[3]]$`Point Forecast`-t(test_))
    NA,#mean(abs(forecasts[[4]]$`Point Forecast`-t(test_)))
    mean(abs(forecasts[[f]][[5]]$`Point Forecast`-t(test_))),
    mean(abs(forecasts[[f]][[6]]$`Point Forecast`-t(test_)))
  )

  return(e_)
})

#raw[,var_of_int]

ave_errors <- lapply(1:length(models[[1]]), function(f)
{#f=1
  errors_ <- lapply(errors, function(e){#e=errors[[2]] [[f]]
    e[[f]]
  })
  return(mean(unlist(errors_)))
})

print(ave_errors)
```

```
## [[1]]
## [1] 2.149997
##
## [[2]]
## [1] NA
##
## [[3]]
## [1] NA
##
## [[4]]
## [1] NA
##
## [[5]]
## [1] 2.09909
##
## [[6]]
## [1] 2.024272
```

```
best_model = which.min(ave_errors)
print(best_model)
```

```
## [1] 6
```

```
#actual forecast

actual_ <- newDF[,var_of_int,drop=FALSE]

actual <- nv_diff_sets(var_of_int,actual_,c())
```

```
## Warning in rownames(raw) == rownames(dataset[1:d_, , drop = FALSE]): longer
## object length is not a multiple of shorter object length
```

```
#raw[,var_of_int]

s_=sndif_[which(colnames(raw)==var_of_int)]
d_=ndif_[which(colnames(raw)==var_of_int)]

fore_data <- newDF[,c(var_of_int,winners)]

f <- as.formula(paste(var_of_int, " ~."))
lm_a <- lm(f, data=fore_data)
summary(lm_a)
```

```
##
## Call:
## lm(formula = f, data = fore_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.5703 -1.9152  0.0373  1.4274  6.7772
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.2795     0.3955   0.707   0.4828
## INUV         -0.5706     0.3170  -1.800   0.0776 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.906 on 53 degrees of freedom
##   (7 observations deleted due to missingness)
## Multiple R-squared:  0.0576, Adjusted R-squared:  0.03981
## F-statistic: 3.239 on 1 and 53 DF,  p-value: 0.07759
```

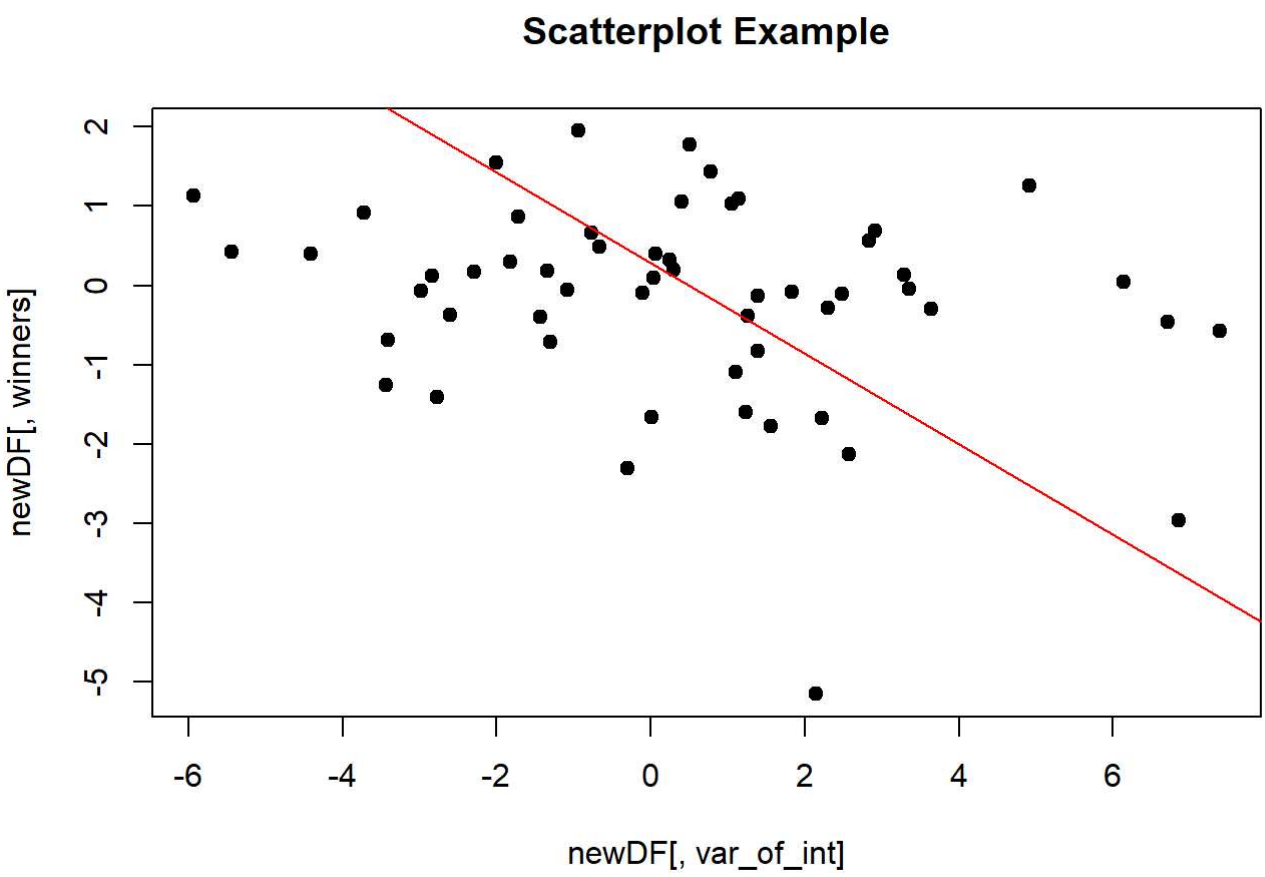
```
differences
```

```
##           LXXRCSA INUV
## lags            0    3
## season          0    0
## nonseason       2    1
```

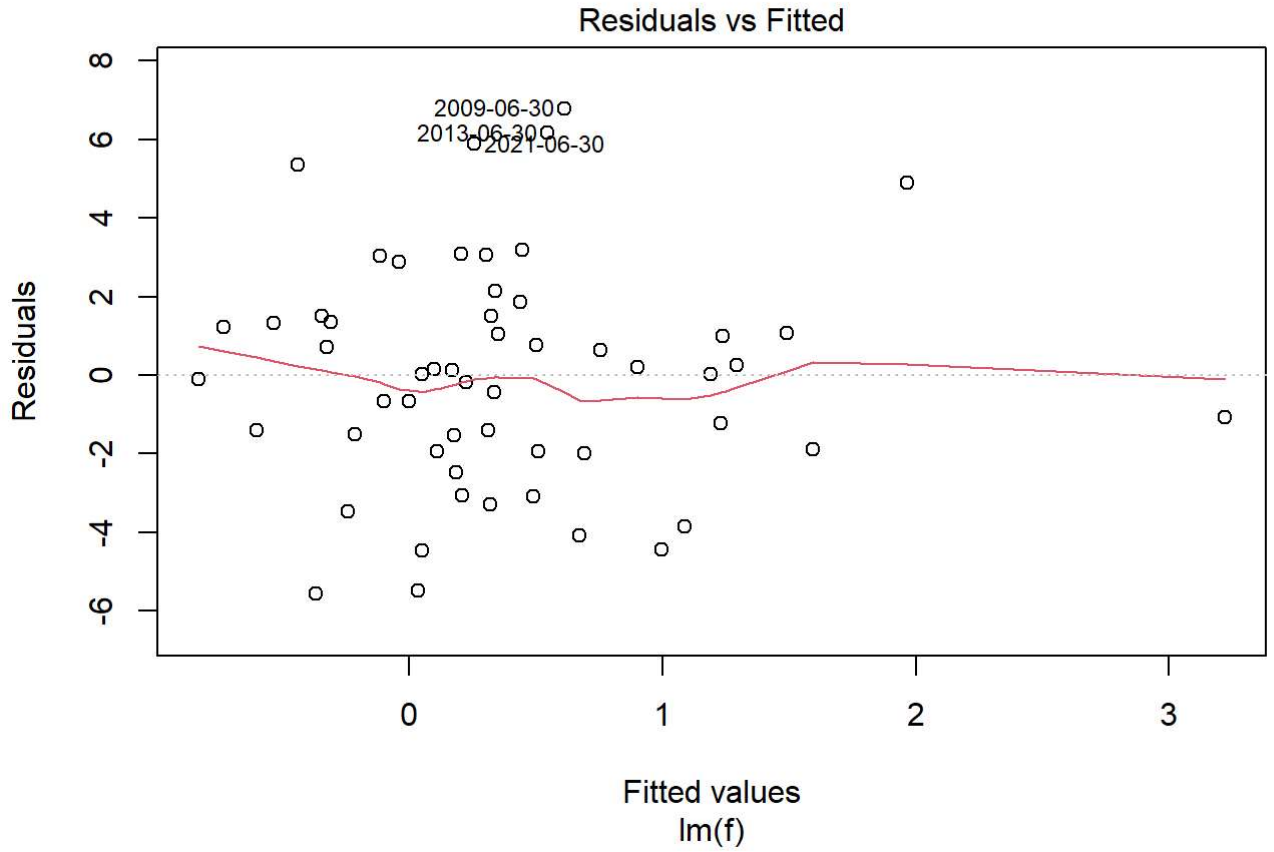
```
cor(na.omit(fore_data))
```

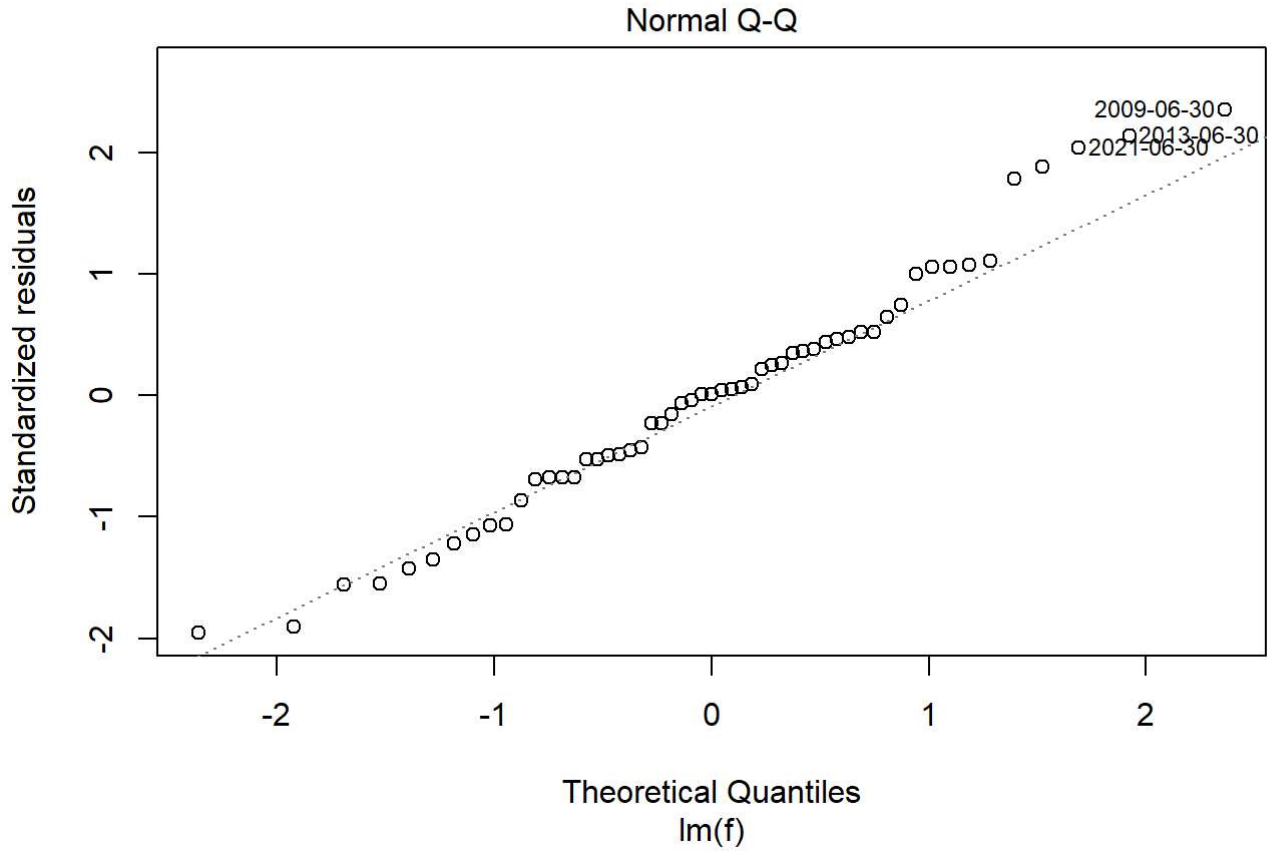
```
##            LXXRCSA       INUV
## LXXRCSA  1.0000000 -0.2399897
## INUV    -0.2399897  1.0000000
```

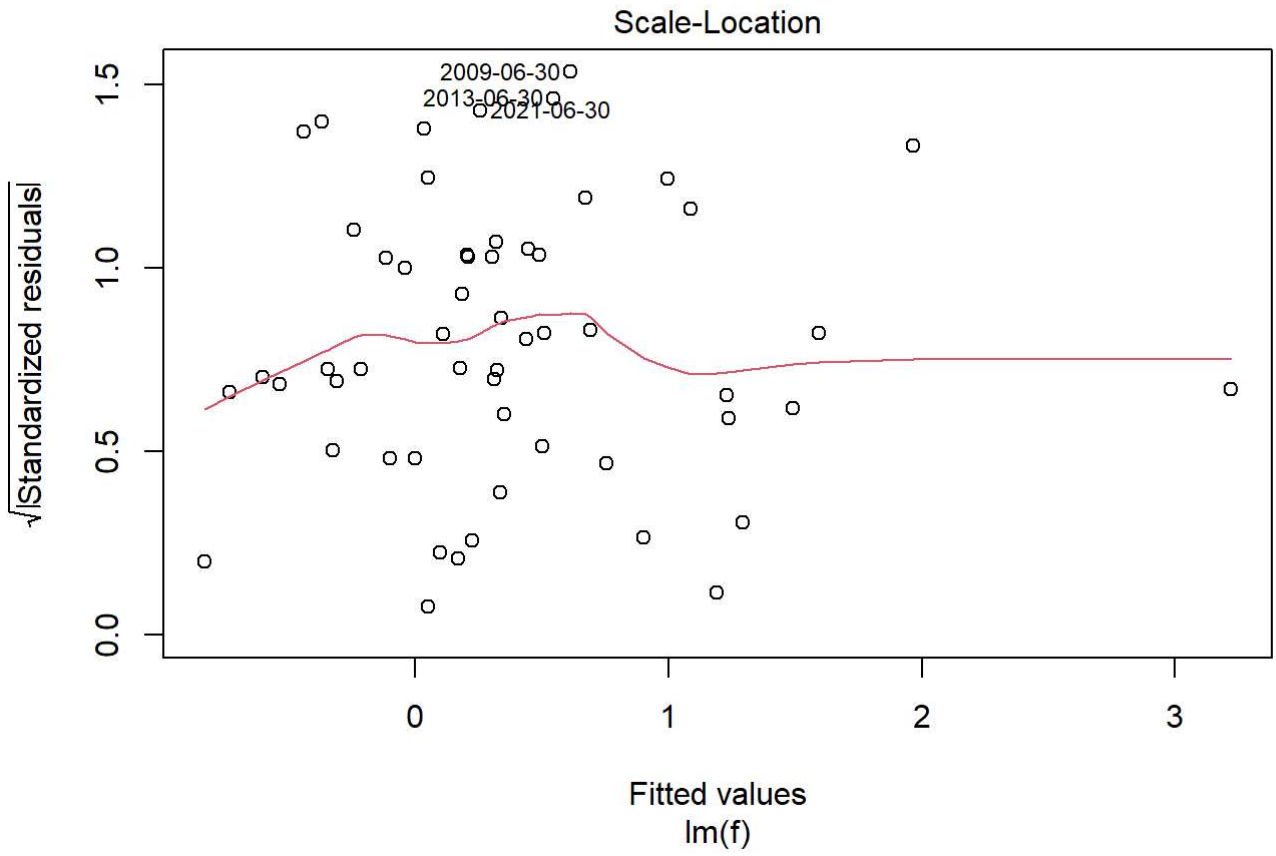
```
plot(newDF[,var_of_int], newDF[,winners], main="Scatterplot Example",pch=19)
abline(lm(newDF[,var_of_int]~ newDF[,winners]), col="red") # regression line (y~x)
```

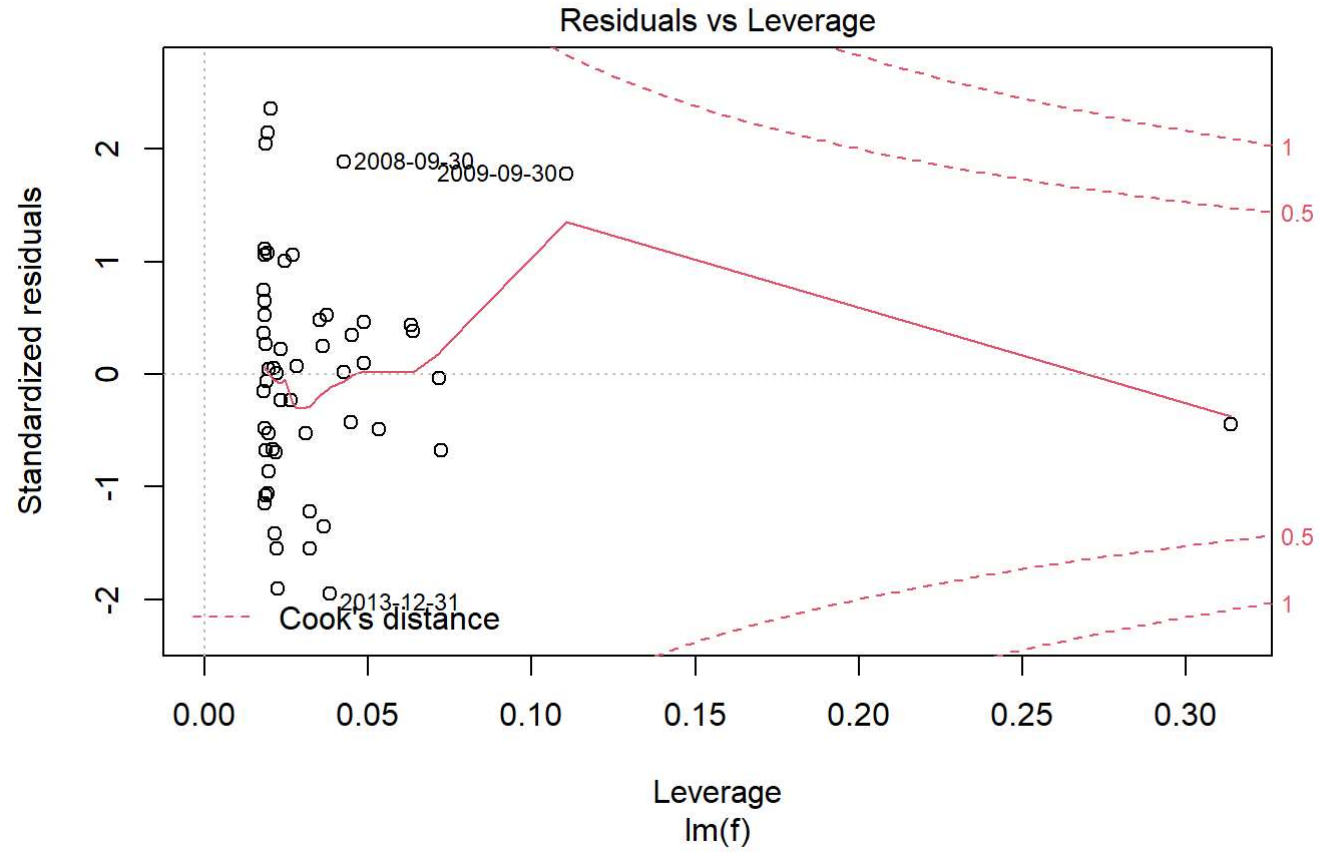



```
#lines(lowess(newDF[,winners],newDF[,var_of_int]), col="blue") # lowess line (x,y)

#summary(lm(newDF[,c(var_of_int,winners)]))

plot(lm_a)
```


Residuals vs Fitted
2009-06-30
2013-06-30
2021-06-30
Residuals
Fitted values
lm(f)

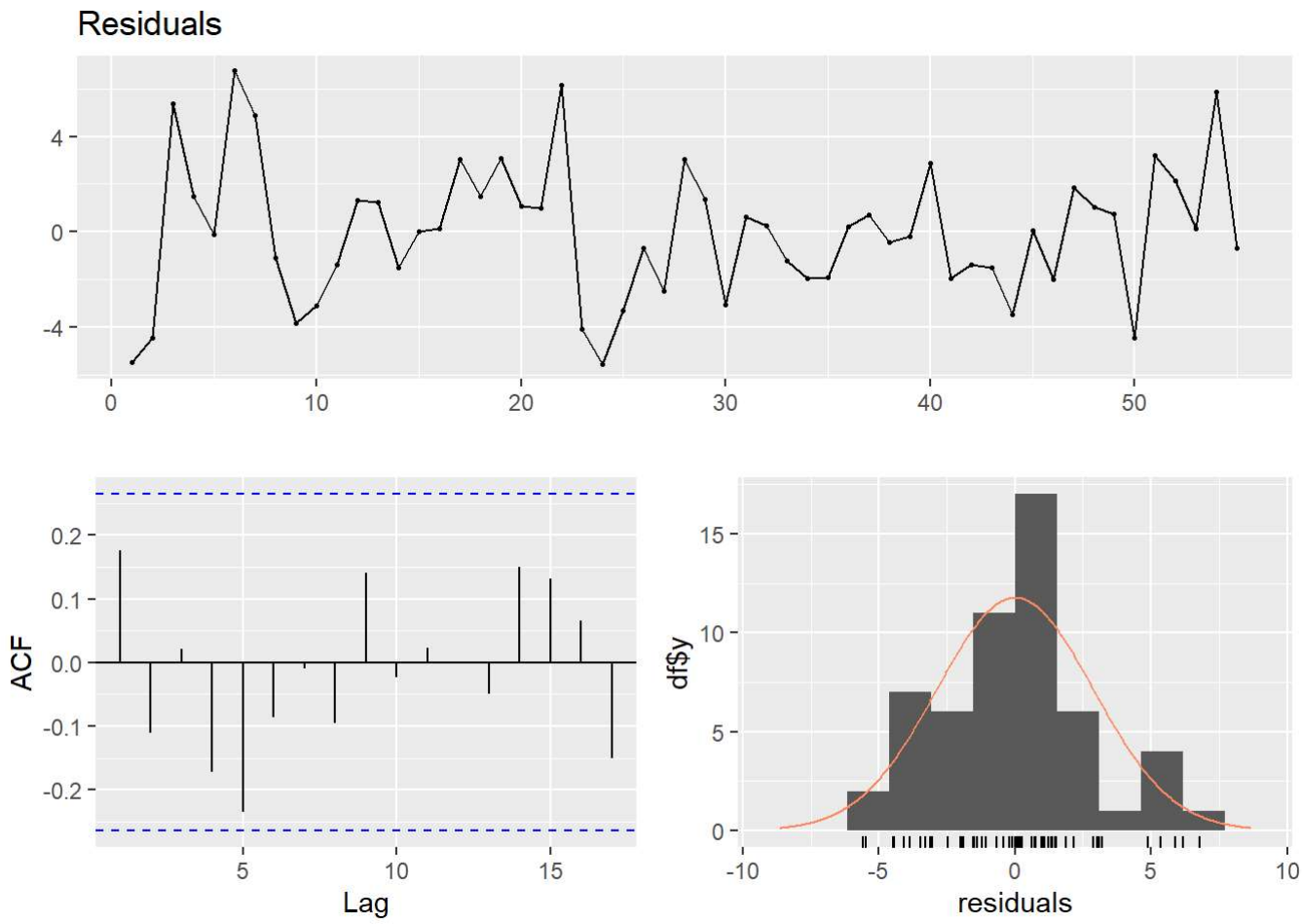

Normal Q-Q
2009-06-30
2013-06-30
2021-06-30
Standardized residuals
Theoretical Quantiles
lm(f)


Scale-Location
2009-06-30
2013-06-30
2021-06-30
√|Standardized residuals|
0.0
0.5
1.0
1.5
0
1
2
3
Fitted values
lm(f)


Residuals vs Leverage
2008-09-30
2009-09-30
2013-12-31
Cook's distance
1
0.5
0.5
1
Standardized residuals
-2
-1
0
1
2
0.00
0.05
0.10
0.15
0.20
0.25
0.30
Leverage
lm(f)

```
checkresiduals(lm_a$residuals)
```

```
## Warning in modeldf.default(object): Could not find appropriate degrees of
## freedom for this model.
```




```
fore_data
```

```
##                LXXRCSA        INUV
## 2007-03-31          NA          NA
## 2007-06-30          NA          NA
## 2007-09-30 -2.73985211          NA
## 2007-12-31 -2.21206411          NA
## 2008-03-31 -5.44089836  0.42884087
## 2008-06-30 -4.41603998  0.40340392
## 2008-09-30  4.92137965  1.26347435
## 2008-12-31  1.14605414  1.09610069
## 2009-03-31 -0.94316270  1.94822935
## 2009-06-30  7.38777991 -0.58018366
## 2009-09-30  6.84943700 -2.95837361
## 2009-12-31  2.13967668 -5.15270989
## 2010-03-31 -2.77068926 -1.41299517
## 2010-06-30 -2.60187854 -0.36950975
## 2010-09-30 -1.99985095  1.54767383
## 2010-12-31  0.78409489  1.42909752
## 2011-03-31  0.50166049  1.77323688
## 2011-06-30 -1.34457562  0.18217593
## 2011-09-30  0.06736623  0.40227978
## 2011-12-31  0.24433740  0.31824765
## 2012-03-31  3.35852486 -0.04469043
## 2012-06-30  1.82964093 -0.08000123
## 2012-09-30  3.28720757  0.13231389
## 2012-12-31  2.56919096 -2.12457932
## 2013-03-31  2.22432846 -1.67610944
## 2013-06-30  6.70688766 -0.46613255
## 2013-09-30 -3.41438311 -0.68620438
## 2013-12-31 -5.93736860  1.13330776
## 2014-03-31 -2.97679020 -0.07353334
## 2014-06-30 -0.76715305  0.66547288
## 2014-09-30 -2.29612797  0.16639137
## 2014-12-31  2.90698094  0.69483598
## 2015-03-31  1.05349940  1.03227871
## 2015-06-30 -2.84295330  0.12471175
## 2015-09-30  1.38636564 -0.83296749
## 2015-12-31  1.55669592 -1.77316712
## 2016-03-31  0.01561390 -1.66296982
## 2016-06-30 -1.43723373 -0.40312192
## 2016-09-30 -0.30460324 -2.30297115
## 2016-12-31  1.10463075 -1.08938997
## 2017-03-31  0.39538871  1.05664406
## 2017-06-30 -0.10198369 -0.09600219
## 2017-09-30  0.03410557  0.09387311
## 2017-12-31  2.82848344  0.56331389
## 2018-03-31 -1.82185530  0.29634921
## 2018-06-30 -1.08396447 -0.05750447
## 2018-09-30 -1.71675458  0.86342609
## 2018-12-31 -3.72190770  0.91449166
## 2019-03-31  1.22911707 -1.59897711
## 2019-06-30 -1.29822595 -0.71943136
## 2019-09-30  2.30644452 -0.27846325
```

```
## 2019-12-31  1.39314549 -0.12738406
## 2020-03-31  1.26347126 -0.38823682
## 2020-06-30 -3.43276277 -1.25861010
## 2020-09-30  3.64013597 -0.29204221
## 2020-12-31  2.48641115 -0.10532289
## 2021-03-31  0.29552415  0.19308521
## 2021-06-30  6.13732680  0.03876225
## 2021-09-30 -0.66549779  0.48885349
## 2021-12-31          NA  0.67891841
## 2022-03-31          NA  1.30707148
## 2022-06-30          NA -0.04993505
```

```
#straight arima of y
models_a <- list(
  auto.arima(fore_data[,var_of_int,drop=FALSE]),
  NA,#auto.arima(lm_$residuals)
  NA,#arfima(lm_$residuals)
  NA,#auto.arima(newDF_t[,var_of_int,drop=FALSE],xreg=lm_$residuals)
  ets(ts(fore_data[,var_of_int,drop=FALSE],frequency=season)),
  auto.arima(fore_data[,var_of_int,drop=FALSE],xreg=as.matrix(structure(data.frame(const=matrix(
1,nrow=nrow(fore_data)),fore_data[,winners]),names=c("constant",winners))))
)
```

```
## Warning in ets(ts(fore_data[, var_of_int, drop = FALSE], frequency = season)):
## Missing values encountered. Using longest contiguous portion of time series
```

```
models_a[best_model]
```

```
## [[1]]
## Series: fore_data[, var_of_int, drop = FALSE]
## Regression with ARIMA(0,0,0) errors
##
## Coefficients:
##       constant     INUV
##         0.2795  -0.5706
## s.e.    0.3883   0.3112
##
## sigma^2 estimated as 8.137:  log likelihood=-135.69
## AIC=277.39   AICc=277.86   BIC=283.41
```

```
forecasts_a <- list(
  {
    f_0 <- as.data.frame(forecast(models_a[[1]],h=horizon))
    f_0
  },
  if(FALSE)
  {
    #arima of lm residuals
    f_1 <- as.data.frame(forecast(models_a[[2]],h=horizon))
    #f_1
    NA
  },
  if(FALSE)
  {
    #arfima of residuals
    f_2 <- as.data.frame(forecast(models_a[[3]],h=horizon))
    #f_2
  }else
  {NA},
  if(FALSE)
  {
    #arima exogenous
    #residuals
    f_3 <- forecast(auto.arima(lm_a$residuals), h=horizon)
    #by selecting models[[4]], it's going to forecast y
    f_3a <- forecast(models_a[[4]],xreg=f_3$mean)
    #f_3a
  }else
    {NA},
  {
    f_4 <- as.data.frame(forecast(models_a[[5]],h=horizon))
    rownames(f_4) <- rownames(actual)
    f_4
  },
  {
    #proper arimax
    f_5 <- as.data.frame(forecast(models_a[[6]],xreg=as.matrix(structure(data.frame(const=matrix
(1,nrow=nrow(head(tail(fore_data[,winners,drop=FALSE],(season-1)),horizon))),head(tail(fore_data
[,winners,drop=FALSE],(season-1)),horizon)),names=c("constant",winners)))))
    rownames(f_5) <- rownames(actual)
    f_5
  }

)

s_=sndif_[which(colnames(raw)==var_of_int)]
d_=ndif_[which(colnames(raw)==var_of_int)]
data_ <- c(na.omit(c(combo_[,var_of_int],forecasts_a[best_model][[1]][,1])))

if(s_==0)
{
  inv_d <- diffinv(data_,differences = d_, xi=raw[1:d_,var_of_int])
```

```
  inv_d <- diffinv(data_,differences=d_,xi=raw[rownames(newDF[1:d_,]),var_of_int])
}else
{

  inv_d <- diffinv(diffinv(data_,differences=d_,xi=raw[rownames(newDF[1:d_,]),var_of_int]),diffe
rences = s_,xi=raw[1:(season+1+d_),var_of_int])
}

prior <- (structure(data.frame(c(combo_[,var_of_int]), row.names = rownames(combo_)),names=var_o
f_int))
#differences
#newDF[,c(var_of_int,winners)]
mean_ <- nv_diff_sets(var_of_int,prior,forecasts_a[best_model][[1]][,1])
```

```
## Warning in rownames(raw) == rownames(dataset[1:d_, , drop = FALSE]): longer
## object length is not a multiple of shorter object length
```

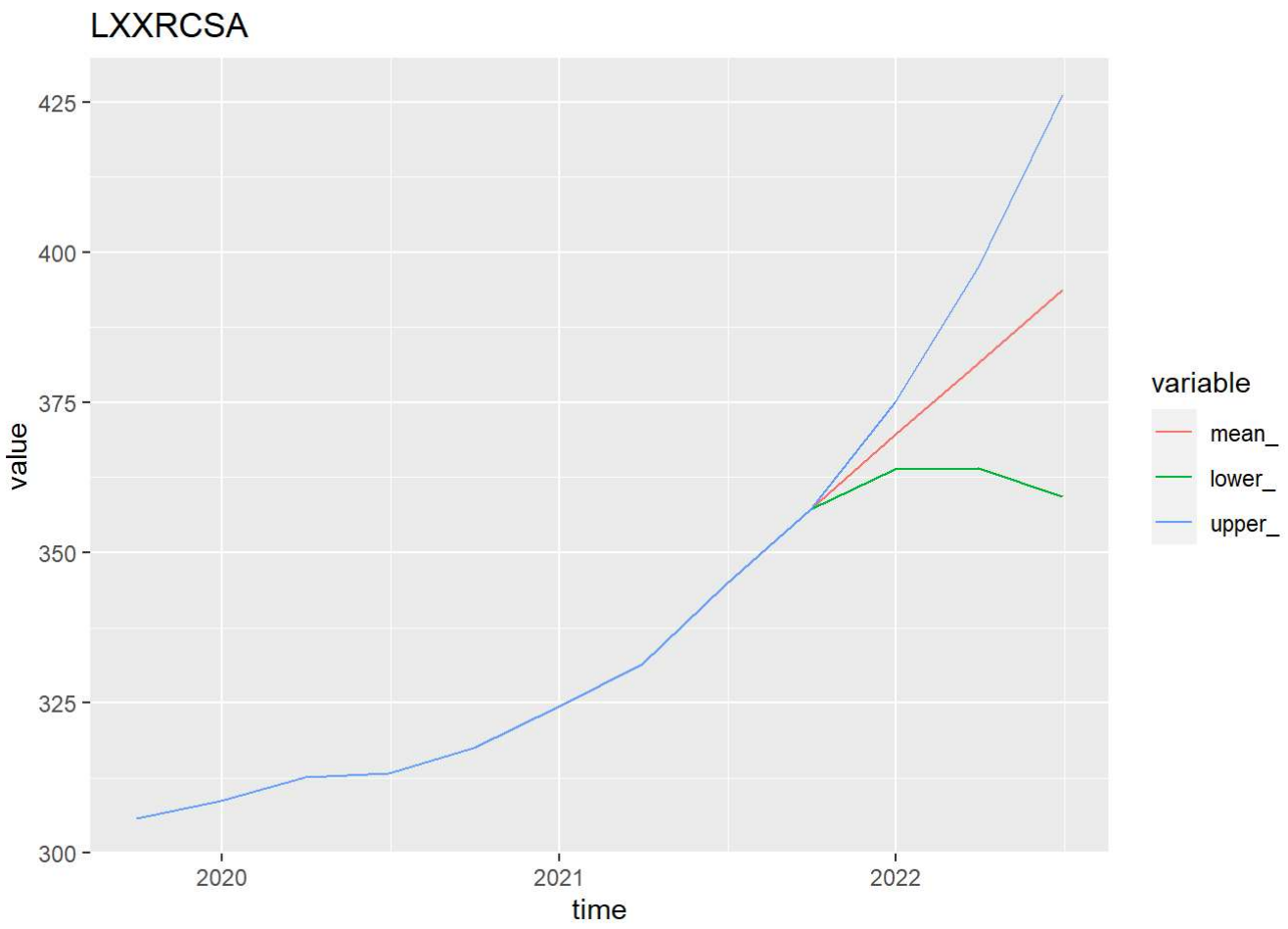
```
lower_ <- nv_diff_sets(var_of_int,prior,(forecasts_a[best_model][[1]][,4]+forecasts_a[best_mode
l][[1]][,1]))
```

```
## Warning in rownames(raw) == rownames(dataset[1:d_, , drop = FALSE]): longer
## object length is not a multiple of shorter object length
```

```
upper_ <- nv_diff_sets(var_of_int,prior,(forecasts_a[best_model][[1]][,5]+forecasts_a[best_mode
l][[1]][,1]))
```

```
## Warning in rownames(raw) == rownames(dataset[1:d_, , drop = FALSE]): longer
## object length is not a multiple of shorter object length
```

```
df_a <- data.frame(time = d_s[1:length(lower_)], mean_,lower_,upper_)

df_a <- melt(tail(df_a,horizon*4) ,  id.vars = 'time', variable.name = 'series')

ggplot(df_a, aes(time,value)) + geom_line(aes(colour = variable)) + ggtitle(var_of_int)
```


LXXRCSA
425
400
375
350
325
300
value
2020
2021
2022
time
variable
mean_
lower_
upper_