

R Code for Assignment 2

Amanda Efendi & Nina Kumagai

13/10/2019

```
library(FrF2)
library(lhs)
library(ggplot2)
library(SPOT)
library(lattice)
```

Stationary Case

```
stationary = read.csv("stationary.csv", header = TRUE, row.names = 1)
stationary$phi <- as.factor(stationary$phi)
stationary$theta <- as.factor(stationary$theta)
stationary$MAPE <- 1/(stationary$MAPE)
head(stationary)
```

```
##      phi theta Method      MAPE
## 1 -0.9  -0.9    ANN  3.3048019
## 2 -0.9  -0.9    ANN  3.5511364
## 3 -0.9  -0.9    ANN  1.2526462
## 4 -0.9  -0.9    ANN  0.6029109
## 5 -0.9  -0.9    ANN  0.7708852
## 6  0.9  -0.9    ANN 100.8064516
```

```
st_anova = aov(MAPE ~ phi * theta * Method, data = stationary)
summary(st_anova)
```

```
##              Df Sum Sq Mean Sq  F value    Pr(>F)
## phi              1  85026    85026 3664.386 < 2e-16 ***
## theta            1   124      124   5.325 0.02764 *
## Method           1   300      300  12.928 0.00107 **
## phi:theta        1   256      256  11.039 0.00224 **
## phi:Method       1   231      231   9.960 0.00347 **
## theta:Method     1   256      256  11.029 0.00225 **
## phi:theta:Method 1   250      250  10.792 0.00247 **
## Residuals       32   743       23
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

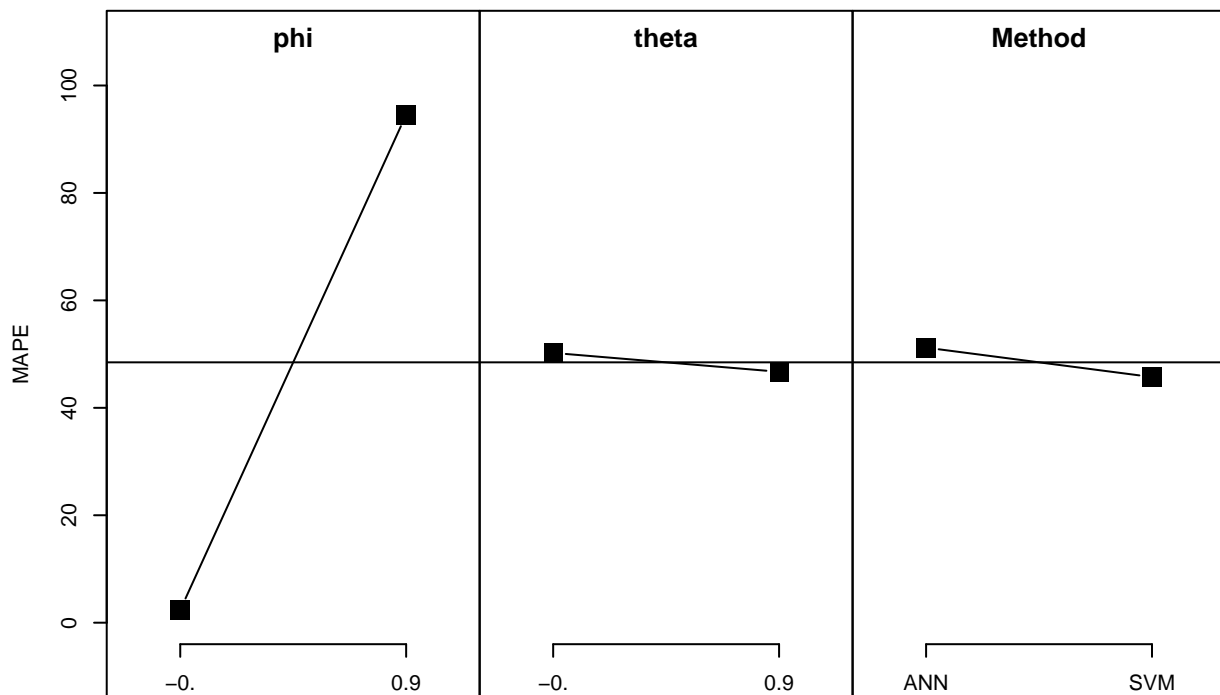
```
step(st_anova)
```

```
## Start:  AIC=132.85
## MAPE ~ phi * theta * Method
##
##              Df Sum of Sq    RSS    AIC
## <none>                742.51 132.85
## - phi:theta:Method  1    250.41  992.92 142.47
##
## Call:
## aov.default(formula = MAPE ~ phi * theta * Method, data = stationary)
```

```
##
## Terms:
##           phi      theta  Method phi:theta phi:Method
## Sum of Squares 85025.95  123.56   299.98   256.14   231.11
## Deg. of Freedom      1      1      1      1      1
##           theta:Method phi:theta:Method Residuals
## Sum of Squares      255.90      250.41   742.51
## Deg. of Freedom      1      1      32
##
## Residual standard error: 4.816983
## Estimated effects may be unbalanced
```

```
#from library FrF2
MEPlot(st_anova)
```

Main effects plot for MAPE



```
shapiro.test(st_anova$residuals)
```

```
##
## Shapiro-Wilk normality test
##
## data: st_anova$residuals
## W = 0.96078, p-value = 0.1781
```

```
library(car)
leveneTest(st_anova)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##      Df F value Pr(>F)
## group 7  1.6237 0.1645
##      32
```

Non-Stationary Case

```
nonstationary = read.csv("non-stationary.csv", header = TRUE, row.names = 1)
nonstationary
```

```
##      theta Method      MAPE
## 1   -0.9     ANN 0.094206
## 2   -0.9     ANN 0.090310
## 3   -0.9     ANN 0.081850
## 4   -0.9     ANN 0.133830
## 5   -0.9     ANN 0.117440
## 6    0.9     ANN 0.091490
## 7    0.9     ANN 0.088000
## 8    0.9     ANN 0.116300
## 9    0.9     ANN 0.089100
## 10   0.9     ANN 0.095680
## 11  -0.9     SVM 0.020776
## 12  -0.9     SVM 0.024260
## 13  -0.9     SVM 0.039130
## 14  -0.9     SVM 0.022100
## 15  -0.9     SVM 0.037200
## 16   0.9     SVM 0.066770
## 17   0.9     SVM 0.062580
## 18   0.9     SVM 0.076060
## 19   0.9     SVM 0.066620
## 20   0.9     SVM 0.064560
```

```
nonstationary$theta <- as.factor(nonstationary$theta)
nonstationary$MAPE <- (1/nonstationary$MAPE)
head(nonstationary)
```

```
##      theta Method      MAPE
## 1   -0.9     ANN 10.615035
## 2   -0.9     ANN 11.072971
## 3   -0.9     ANN 12.217471
## 4   -0.9     ANN  7.472166
## 5   -0.9     ANN  8.514986
## 6    0.9     ANN 10.930156
```

```
non_st_anova = aov(MAPE ~ theta * Method, data = nonstationary)
summary(non_st_anova)
```

```
##              Df Sum Sq Mean Sq F value    Pr(>F)
## theta          1  602.4   602.4   20.64 0.000333 ***
## Method         1 1267.0  1267.0   43.41 6.24e-06 ***
## theta:Method    1  662.5   662.5   22.70 0.000211 ***
## Residuals     16  467.0    29.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
step(non_st_anova)
```

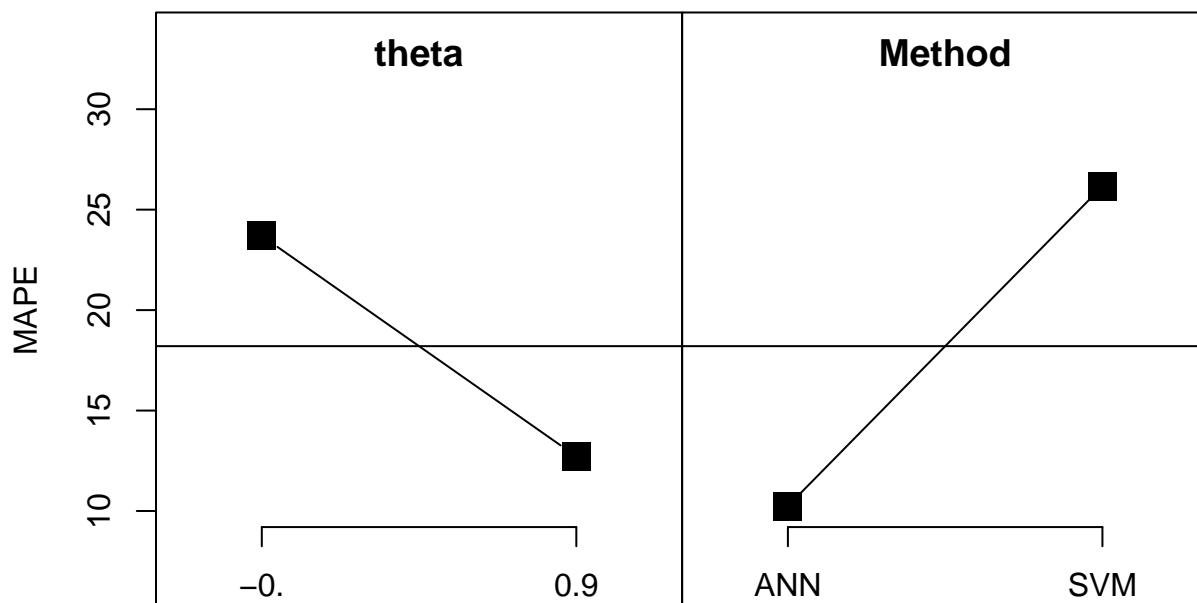
```
## Start:  AIC=71.01
## MAPE ~ theta * Method
##
##              Df Sum of Sq    RSS    AIC
```

```
## <none>                                467.01 71.012
## - theta:Method 1      662.51 1129.52 86.676

## Call:
## aov.default(formula = MAPE ~ theta * Method, data = nonstationary)
##
## Terms:
##          theta      Method theta:Method Residuals
## Sum of Squares  602.3708 1266.9630      662.5116 467.0129
## Deg. of Freedom      1          1          1      16
##
## Residual standard error: 5.40262
## Estimated effects may be unbalanced
```

```
MEPlot(non_st_anova)
```

Main effects plot for MAPE



```
shapiro.test(non_st_anova$residuals)
```

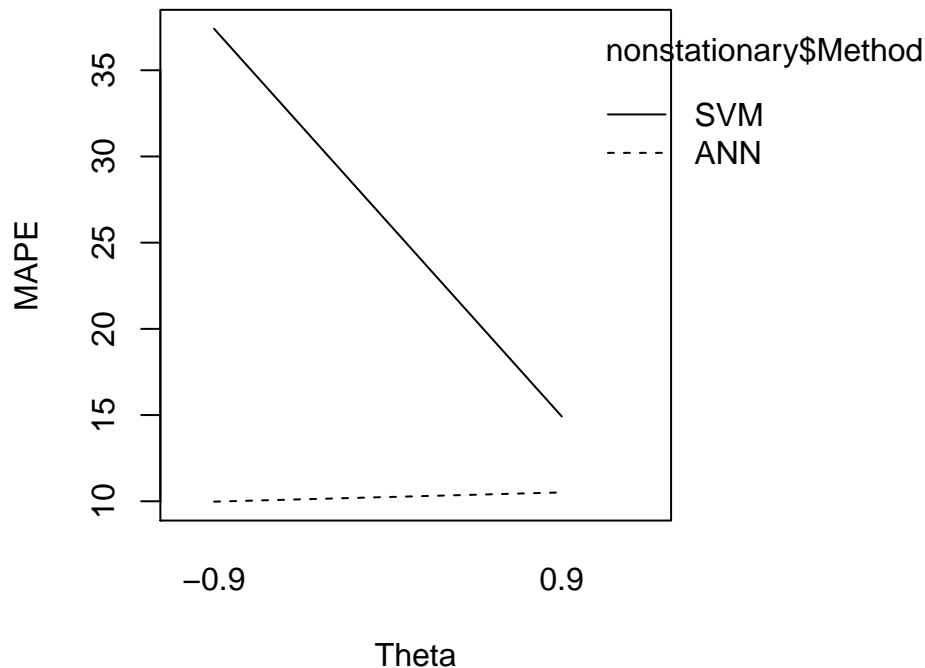
```
##
## Shapiro-Wilk normality test
##
## data: non_st_anova$residuals
## W = 0.86185, p-value = 0.008475
```

```
leveneTest(non_st_anova)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##      Df F value    Pr(>F)
## group 3  5.5222 0.008498 **
##      16
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
par(pty='s')
interaction.plot(nonstationary$theta, nonstationary$Method, nonstationary$MAPE, xlab='Theta', ylab='MAPE')
```

Non-Stationary – Interaction



Check for Normality and Homogeneity of Variance

High theta and low theta error is other way around for the STAT2003 assignment Q1 because inverting the model will make larger errors small and small errors large. This means that it is now the larger errors that show smaller error!

Make sure to use multidimensional krigging in the second question of the assignment.

Applying inverse on non-stationary actually makes it fail the Levene test of homogeneity although it makes up for the other assumption of normality. But Levene homogeneity of variance is arguably the more important assumption to hold.

Thus the article does both stationary and non-stationary cases wrong!

Question 2

```
library(mvtnorm)
library(e1071)
```

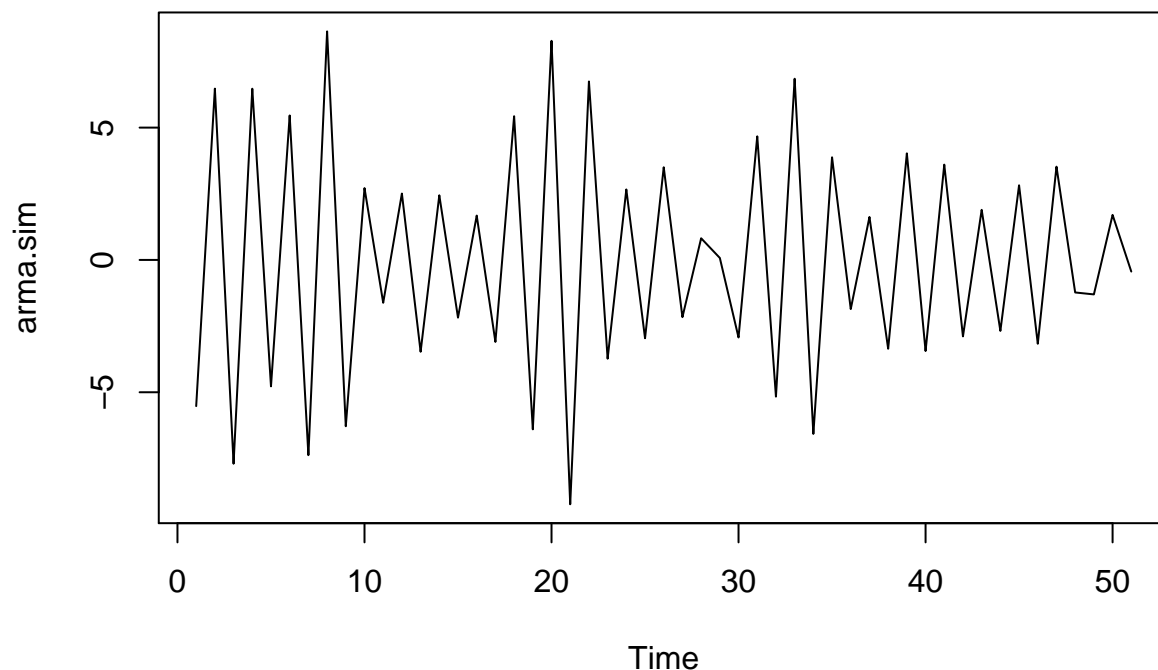
Stationary Process

```
#Stationary process - Simulation Test
set.seed(61)
arma.sim <- arima.sim(model=list(ar=-0.9,ma=-0.9),n=51)
arma.sim
```

```
## Time Series:
```

```
## Start = 1
## End = 51
## Frequency = 1
## [1] -5.52412295  6.47309097 -7.70124679  6.46978196 -4.77689181
## [6]  5.45929444 -7.37829255  8.63630155 -6.28486440  2.71437507
## [11] -1.61894120  2.50796915 -3.47061185  2.44251798 -2.18049648
## [16]  1.67534722 -3.09863685  5.43139381 -6.40205106  8.27961566
## [21] -9.23274825  6.74451014 -3.73189614  2.66370874 -2.95950364
## [26]  3.50133309 -2.16212065  0.81852349  0.07435405 -2.93009761
## [31]  4.67010294 -5.16661964  6.84739078 -6.57443077  3.87830163
## [36] -1.85745259  1.62376942 -3.35658313  4.02793098 -3.44350568
## [41]  3.60182569 -2.88885637  1.89381802 -2.68262368  2.82001595
## [46] -3.16803091  3.52396275 -1.22926979 -1.30301083  1.70207093
## [51] -0.43889334
```

```
ts.plot(arma.sim)
```



Function for simulating data

```
# Simulate ARIMA based on the phi and theta parameter as function input
# Output the result as a 51x1 matrix with each row having 51 sequences of ts data
```

```
simData <- function(phi,theta){
  all_training=NULL
  for (i in 1:50){
    armaSet = arima.sim(model=list(ar=phi, ma=theta),n=51)
    all_training = rbind(all_training, armaSet[1:51])
  }
  return(all_training)
}
```

```
#LLdata means low setting for both phi and theta
#LLdata has 50 rows and 51 columns
```

```
#set.seed(61)
#phi;theta <- runif(2,-0.9,0.9)
#data = simData(phi,theta)
```

TRIAL - start SVM prediction using first row

```
# training data is column 1 to 50
# test data is column 51st
# we only use the first row to test out.. not sure how to run all 50 rows..
#trainSVMDat = data.frame(timestamp=c(seq(1,50,1)),value=data[1,1:50])
#testSVMDat = data.frame(timestamp=1,value=data[1,51])

# svm(y~x,data=training)
# the epsilon, gamma and cost values were taken from the paper
# the syntax we used is "to predict value using timestamp"
# predict(trainingmodel, newdata=testset)
#modelSVM = svm(value~timestamp,data=trainSVMDat,epsilon=0.1,gamma=0.1,cost=10)
#predSVM = predict(LHmodelSVM,newdata=testSVMDat); predSVM

# MAPE is the prediction - actual data divided by the prediction
#MAPE = (LHpredSVM - testSVMDat[,2])/LHpredSVM; abs(MAPE)
```

Function for automating the generation of MAPE values in SVM (Nina did after Basement Cafe :))

```
#Automating the MEAN ABSOLUTE VALUE of MAPE - 1 replication
calc_mape = function(data_name){
  mape_matrix = NULL
  for (i in 1:50){
    #create the training and testing dataset from the chosen dataset (as per input)
    #each train and test is only one row of the 50 rows dataset
    trainDat = data.frame(timestamp=c(seq(1,50,1)),value=data_name[i,1:50]);trainDat
    testDat = data.frame(timestamp=1,value=data_name[i,51])

    #create a SVM model with the paramteres taken from the paper
    modelSVM = svm(value~timestamp,data=trainDat,epsilon=0.1,gamma=0.1,cost=10)
    predSVM = predict(modelSVM,newdata=testDat)

    #calculate the MAPE and "append" the value into the matrix
    MAPE = (predSVM - testDat[,2])/predSVM
    mape_matrix = rbind(mape_matrix, MAPE)
  }
  #label <- paste("MAPE", data_name, sep = "_")
  #assign(label, mape_matrix)

  #return the absolute value of the mean of the 50 rows of MAPE
  return (abs(mean(mape_matrix)))
}
```

Replicating the functions above 5 times for each setting

```
#Calculate the MAPE again but for all 20 random parameters
#This is like a wrapper function
```

```

replicatedMAPE = function(){
  #set the parameters as NULL
  phi=NULL
  theta=NULL
  #variables to contain results
  replicates = NULL
  param = NULL
  #set up the LHS design for input into simData
  design = maximinLHS(100,2,method='build')
  LHSdesign = qunif(design,-0.9,0.9)
  for (i in 1:100){
    phi <- LHSdesign[i,1]
    theta <- LHSdesign[i,2]
    #simulate stationary data using simData
    dataset = simData(phi,theta)
    #calculate MAPE
    newRep = calc_mape(dataset)
    #append results into existing variable
    replicates = rbind(replicates, newRep)
    param = rbind(param, c(phi, theta))
  }
  dataset = cbind(param,replicates)
  return(dataset)
}

```

Create the final dataset for MAPEs from Stationary

```

set.seed(88)
simStationary = replicatedMAPE()

colnames(simStationary) <- c("phi","theta","MAPE")
simStationary <- as.data.frame(simStationary)

```

Plotting MAPE

```

library(plotly)
p <- plot_ly(simStationary, x = ~phi, y = ~MAPE, z = ~theta, colors = c('#BF382A', '#0C4B8E')) %>%
  add_markers() %>%
  layout(scene = list(xaxis = list(title = 'Phi'),
                      yaxis = list(title = 'MAPE'),
                      zaxis = list(title = 'Theta')))
p

```

```

#DONT RUN THIS CHUNK AGAIN
simStationary <- simStationary[simStationary$MAPE<10,]

```

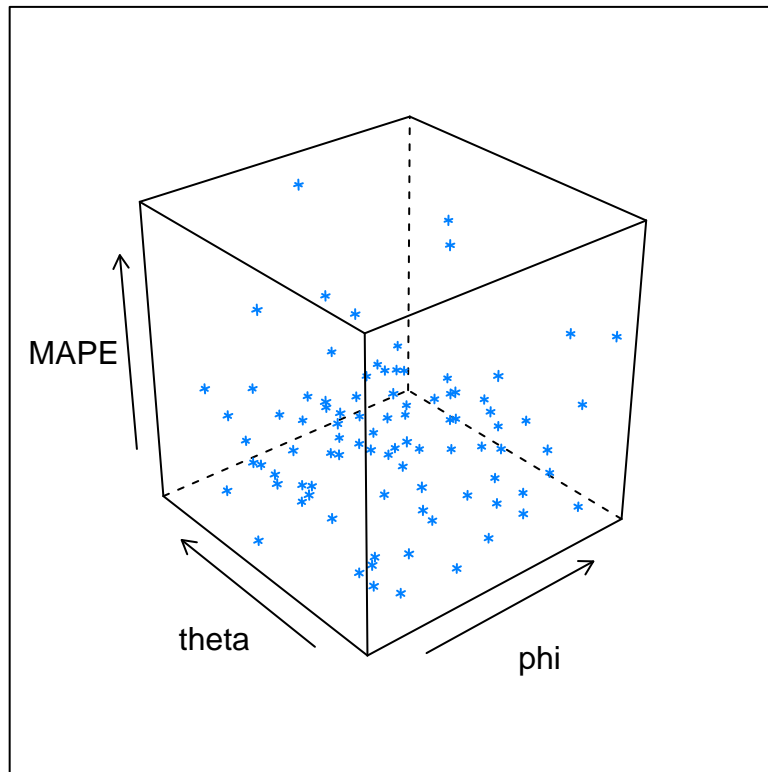
```

p <- plot_ly(as.data.frame(simStationary), x = ~phi, y = ~MAPE, z = ~theta, colors = c('#BF382A', '#0C4B8E')) %>%
  add_markers() %>%
  layout(scene = list(xaxis = list(title = 'Phi'),
                      yaxis = list(title = 'MAPE'),
                      zaxis = list(title = 'Theta')))
p

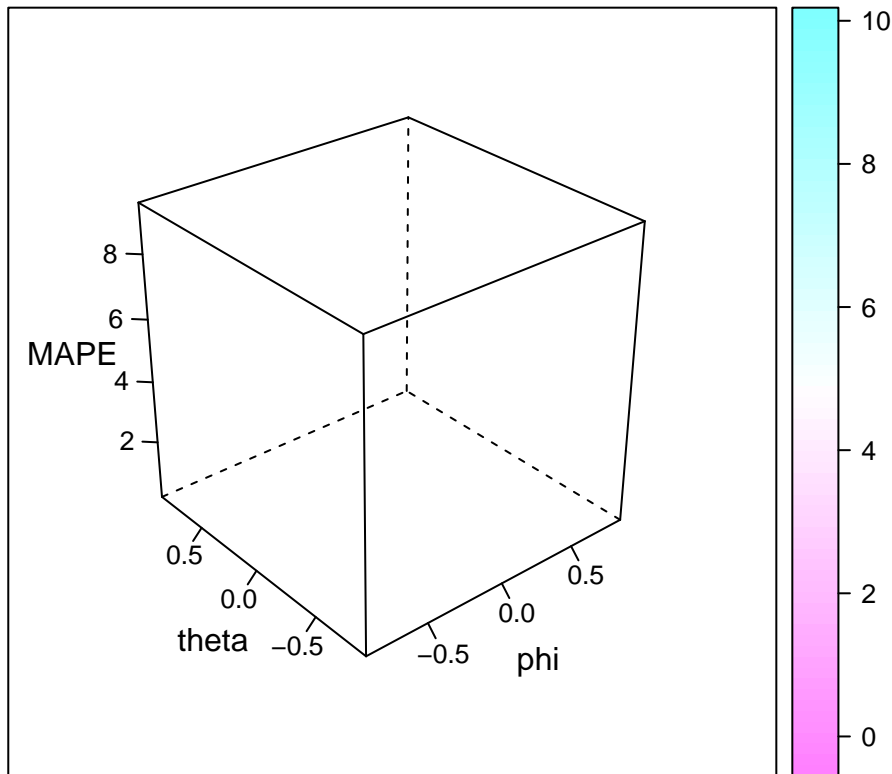
```



```
#from library(lattice)
cloud(MAPE~phi*theta, data=simStationary,
      aspect = c(1, 1),
      panel = "panel.cloud",
      scales = list(distance = rep(1, 3), arrows = TRUE),
      zoom = 0.8)
```



```
wireframe(MAPE ~ phi*theta, data = simStationary[simStationary$MAPE>0,], scales = list(arrows = FALSE),
```



Multi-dimensional Kriging

```
#Create design and response matrix of the SimulatedStationary for KrigingDace Model
# design is the phi and theta setting from LHS
# response is the MAPE
kriging_design = simStationary[1:2]
kriging_design = as.matrix(kriging_design)
kriging_response = simStationary[3]
kriging_response = as.matrix(kriging_response)
```

```
# Make the KRIGING MODEL
# MAPE ~ phi * theta, data = simStationary
fit = buildKrigingDACE(kriging_design, kriging_response)
fit
```

```
## -----
## Dace Kriging model.
## -----
## Estimated activity parameters (theta) sorted
## from most to least important variable
## x1 x2
## 2.818383 2.818383
##
## exponent(s) p:
## 1.9 1.9
##
## Estimated regularization constant (or nugget) lambda:
## 0.999
##
```

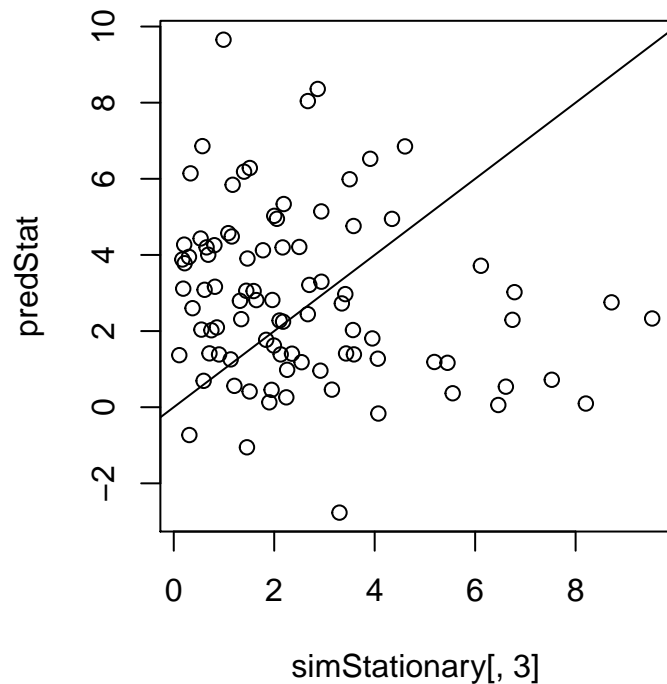
```
## Number of Likelihood evaluations during MLE:
##
## -----
```

Test the Kriging Model

```
#Create a new design
set.seed(57)
design <- maximinLHS(nrow(simStationary),2,method='build')
LHSdesign2 <- qunif(design, -0.9,0.9)
#Predict
predStat <- predict(fit,LHSdesign2)$y

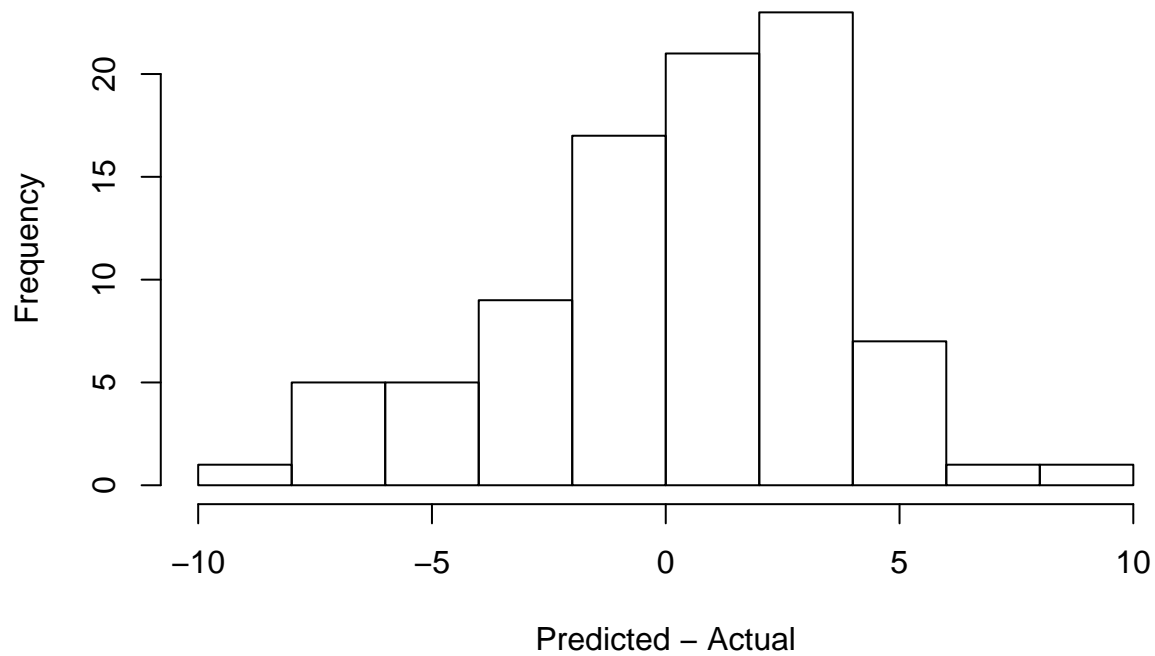
# TESTING PREDICTION ACCURACY
par(pty='s')
plot(predStat~simStationary[,3], main='Predicted MAPE vs Simulated MAPE')
abline(0,1)
```

Predicted MAPE vs Simulated MAPE



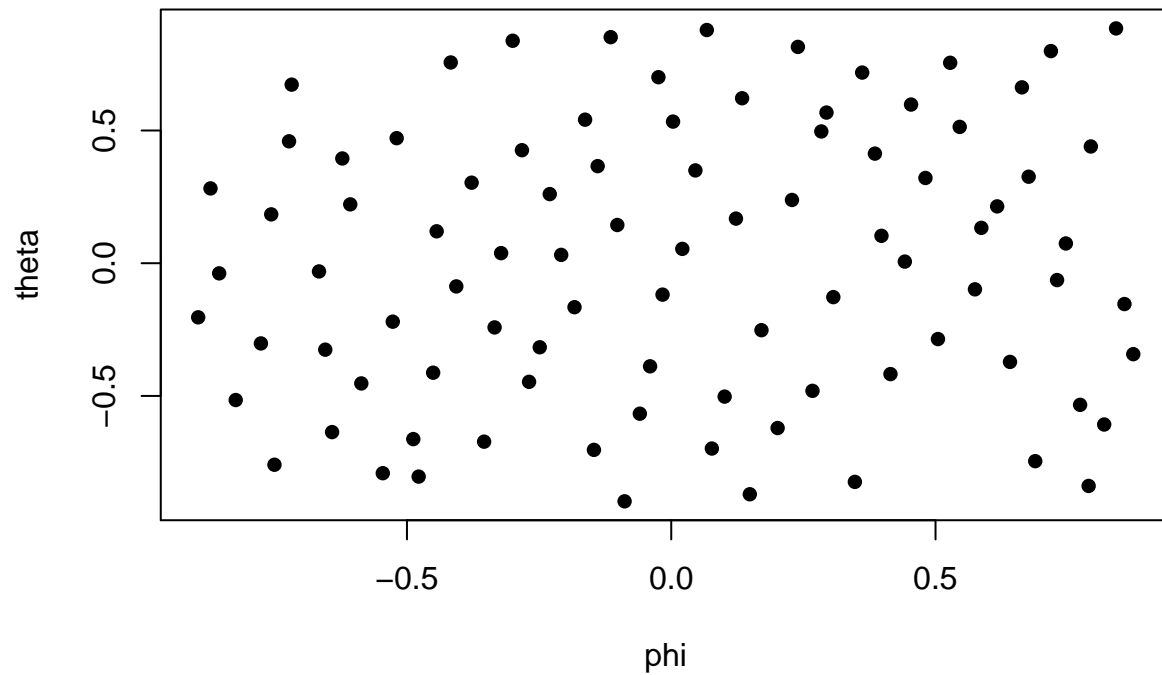
```
hist(predStat-simStationary[,3], main = "Histogram of the Differences between Predicted and Actual MAPE")
```

Histogram of the Differences between Predicted and Actual MAPE val

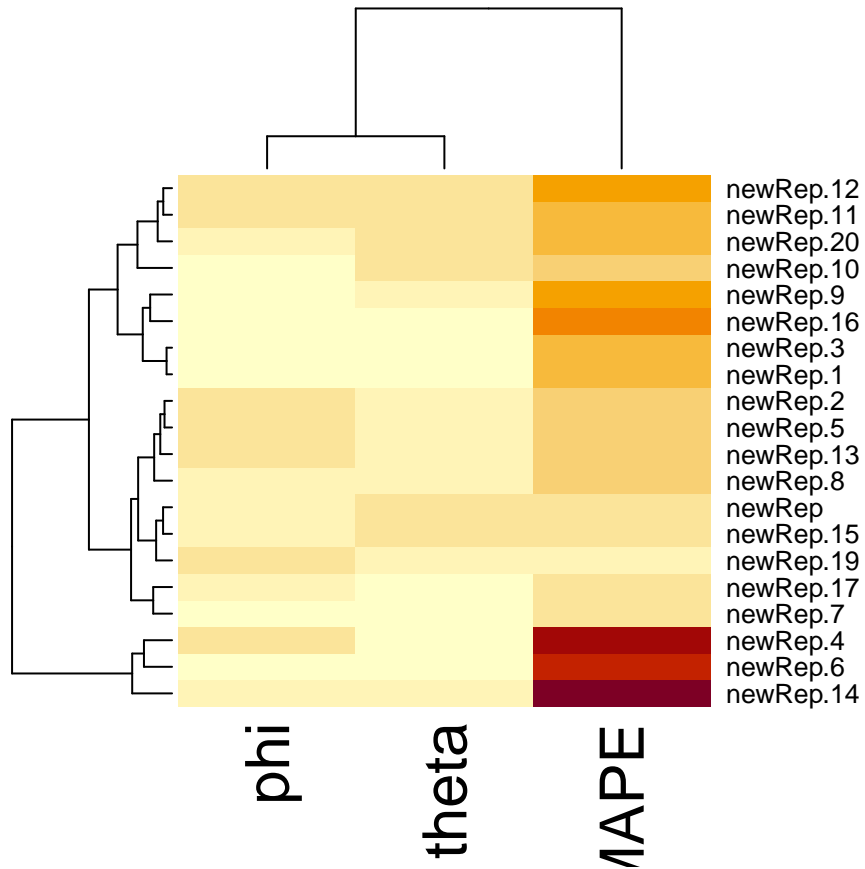


```
#(p <- ggplot(simStationary[simStationary$MAPE<10,], aes(phi, theta)) + geom_raster(aes(fill = MAPE), i  
plot(theta~phi, data=simStationary, main='Results of LHS', pch=16)
```

Results of LHS

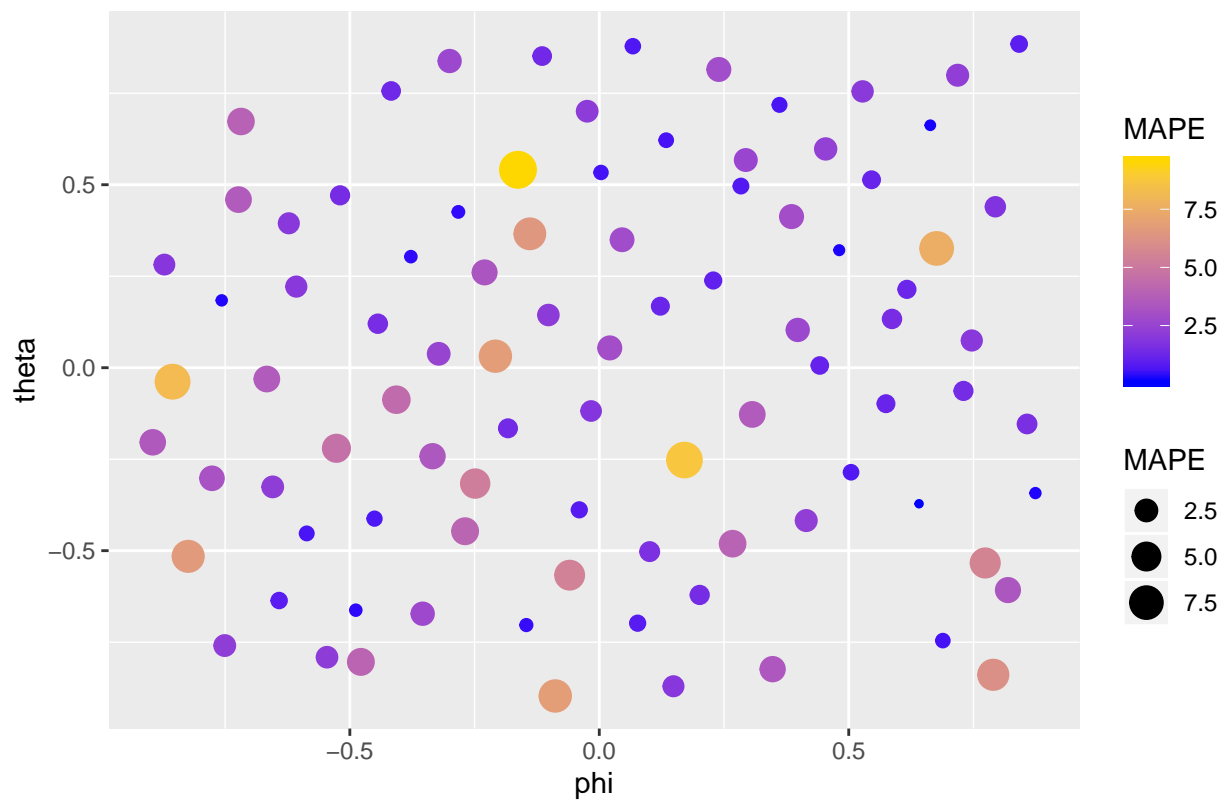


```
stationaryMatrix <- as.matrix(simStationary[1:20,])
heatmap(stationaryMatrix, scale='none')
```



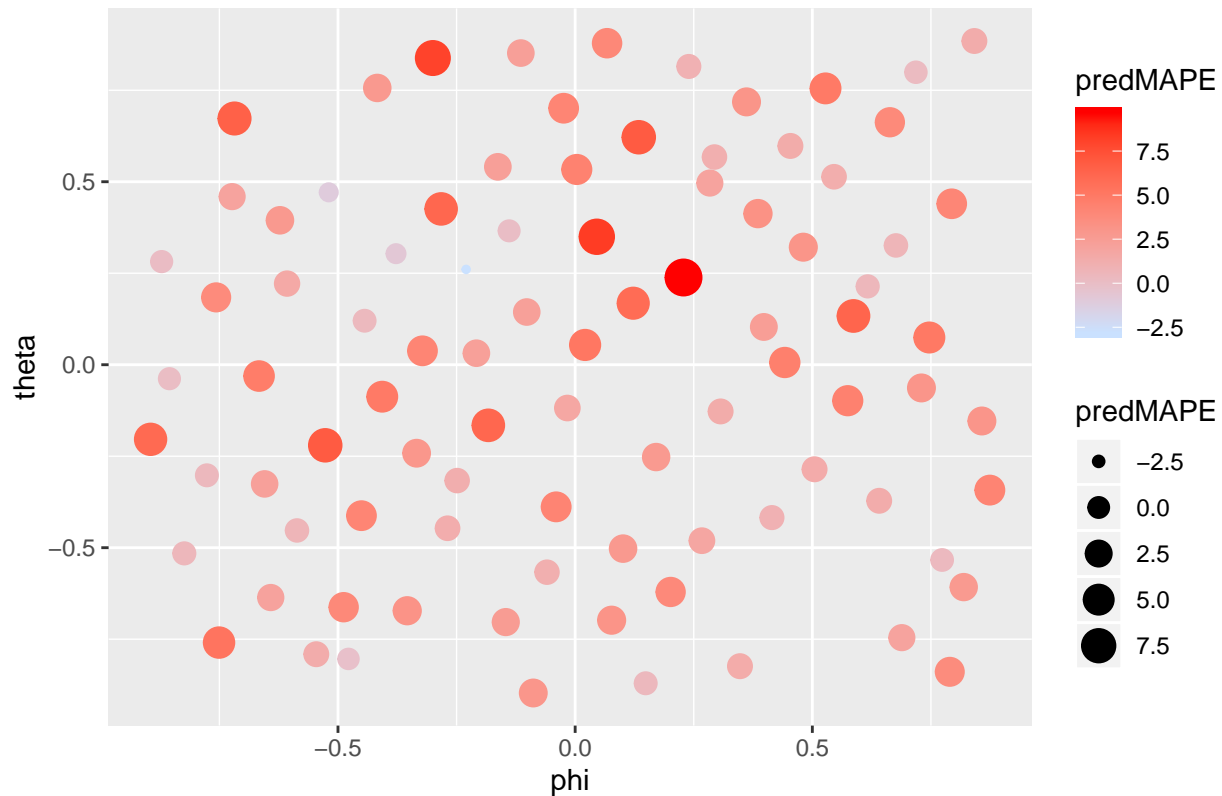
```
ggplot(simStationary, aes(x=phi, y=theta, color=MAPE, size=MAPE)) + geom_point() + scale_color_gradient
```

MAPE Values for Corresponding Phi and Theta Values – Stationary



```
predData <- data.frame(simStationary[,1:2], predMAPE = predStat)
ggplot(predData, aes(x=phi, y=theta, color=predMAPE, size=predMAPE)) + geom_point() + scale_color_gradient2()
```

Predicted MAPE Values for Corresponding Phi and Theta Values – Station



IMA MODEL

Functions (Simulate and Calc MAPE)

```
# Simulate IMA based model with theta parameter as function input
# Output the result as a 50x51 matrix with each row having 51 sequences of ts data

simData_nonst <- function(theta){
  all_training=NULL
  for (i in 1:50){
    arimaSet = arima.sim(model=list(ma=theta),n=51)
    all_training = rbind(all_training, arimaSet[1:51])
  }
  return(all_training)
}

#Automating the MEAN ABSOLUTE VALUE of MAPE - 1 replication
calc_mape_nonst = function(data_name){
  mape_matrix = NULL
  for (i in 1:50){
    #create the training and testing dataset from the chosen dataset (as per input)
    #each train and test is only one row of the 50 rows dataset
    trainDat = data.frame(timestamp=c(seq(1,50,1)),value=data_name[i,1:50]);trainDat
    testDat = data.frame(timestamp=1,value=data_name[i,51])

    #create a SVM model with the paramteres taken from the paper
    modelSVM = svm(value~timestamp,data=trainDat,epsilon=0.1,gamma=0.1,cost=10)
  }
}
```

```

predSVM = predict(modelSVM,newdata=testDat)

#calculate the MAPE and "append" the value into the matrix
MAPE = (predSVM - testDat[,2])/predSVM
mape_matrix = rbind(mape_matrix, MAPE)
}

#label <- paste("MAPE", data_name, sep = "_")
#assign(label, mape_matrix)

#return the absolute value of the mean of the 50 rows of MAPE
return (abs(mean(mape_matrix)))
}

```

Replicating the functions above 5 times for each setting

```

#Calculate the MAPE again but for all 20 random parameters
#This is like a wrapper function

replicatedMAPE_nonst = function(){
  theta=NULL
  replicates = NULL
  param = NULL
  design = maximinLHS(100,1,method='build')
  LHSdesign = qunif(design, -0.9, 0.9)
  for (i in 1:100){
    theta <- LHSdesign[i,1]
    dataset = simData_nonst(theta)
    newRep = calc_mape_nonst(dataset)
    replicates = rbind(replicates, newRep)
    param = rbind(param, theta)
  }
  dataset = cbind(param,replicates)
  return(dataset)
}

```

Create the final dataset for MAPEs from Non Stationary

```

set.seed(29)
simNonStat = replicatedMAPE_nonst()

colnames(simNonStat) <- c("theta","MAPE")
simNonStat <- as.data.frame(simNonStat)

```

Plotting MAPE

```

p <- plot_ly(simNonStat, x = ~theta, y = ~MAPE, colors = c('#BF382A', '#0C4B8E')) %>%
  add_markers() %>%
  layout(scene = list(xaxis = list(title = 'Theta'),
    yaxis = list(title = 'MAPE')))
p

simNonStat <- simNonStat[simNonStat$MAPE<20,]

```



```

p <- plot_ly(simNonStat, x = ~theta, y = ~MAPE, colors = c('#BF382A', '#0C4B8E')) %>%
  add_markers() %>%
  layout(scene = list(xaxis = list(title = 'Theta'),
                        yaxis = list(title = 'MAPE')))
p

```

Multidimensional Kriging

```

#Create design and response matrix of the SimulatedStationary for KrigingDace Model
# design is the phi and theta setting from LHS
# response is the MAPE
kriging_design = simNonStat[1]
kriging_design = as.matrix(kriging_design)
kriging_response = simNonStat[2]
kriging_response = as.matrix(kriging_response)

# Make the KRIGING MODEL
# MAPE ~ phi * theta, data = simStationary
fit2 = buildKrigingDACE(kriging_design, kriging_response)
fit2

## -----
## Dace Kriging model.
## -----
## Estimated activity parameters (theta) sorted
## from most to least important variable
## x1
## 0.03406543
##
## exponent(s) p:
## 2
##
## Estimated regularization constant (or nugget) lambda:
## 0.7406117
##
## Number of Likelihood evaluations during MLE:
##
## -----

```

Test the Kriging Model

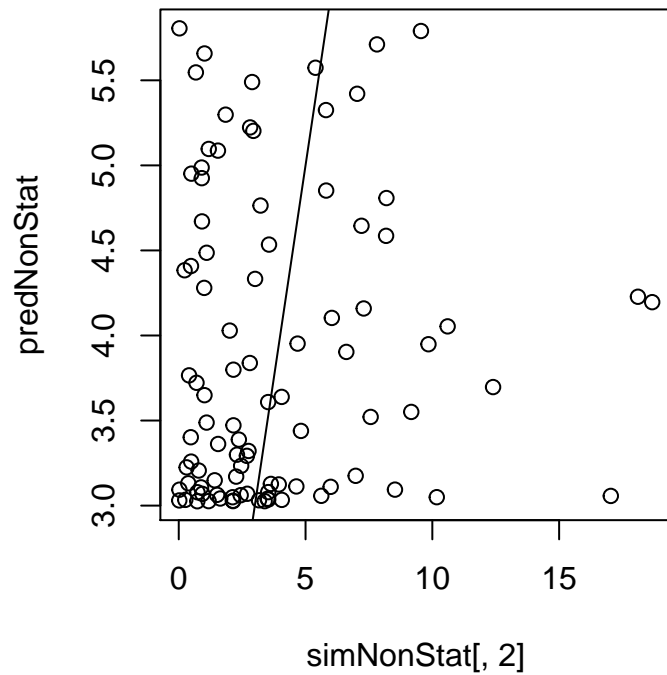
```

#Create a new design
set.seed(57)
design <- maximinLHS(nrow(simNonStat),1,method='build')
LHSdesign2 <- qunif(design, -0.9, 0.9)
#Predict
predNonStat <- predict(fit2,LHSdesign2)$y

# TESTING PREDICTION ACCURACY
par(pty='s')
plot(predNonStat~simNonStat[,2], main='Predicted MAPE vs Simulated MAPE')
abline(0,1)

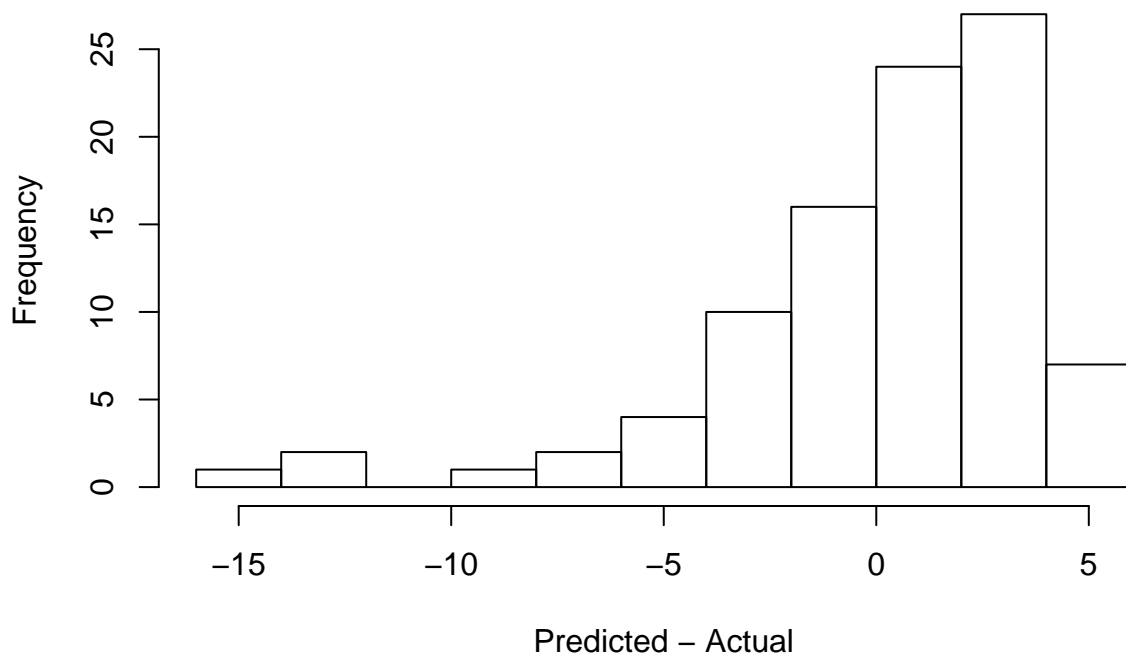
```

Predicted MAPE vs Simulated MAPE

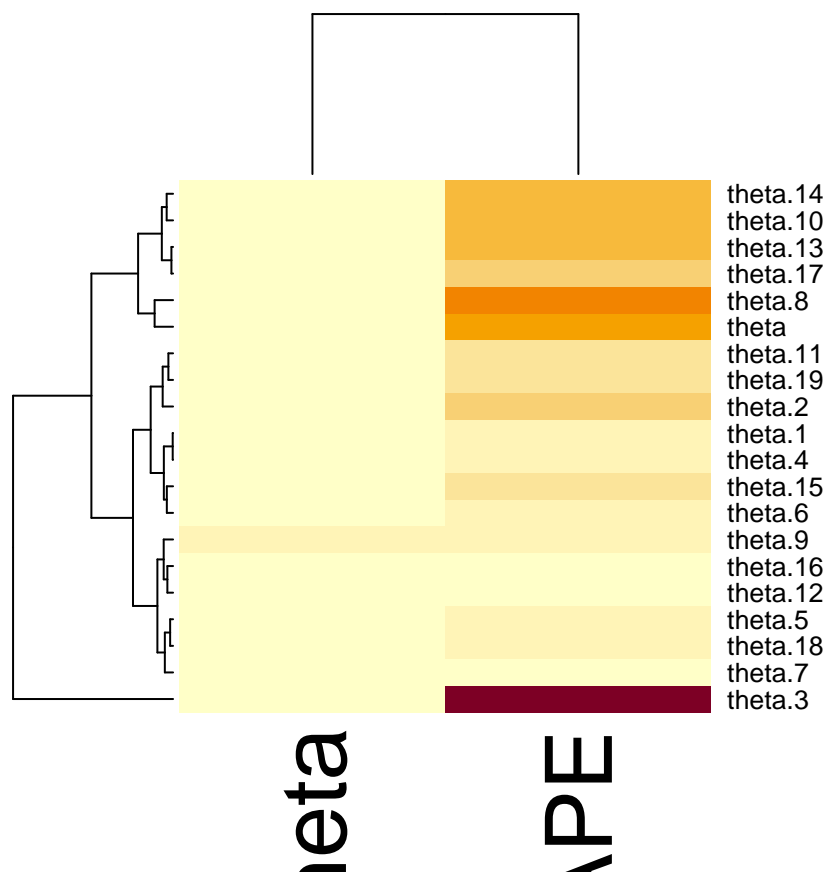


```
hist(predNonStat-simNonStat[,2], main = "Histogram of the Differences between Predicted and Actual MAPE")
```

Histogram of the Differences between Predicted and Actual MAPE values

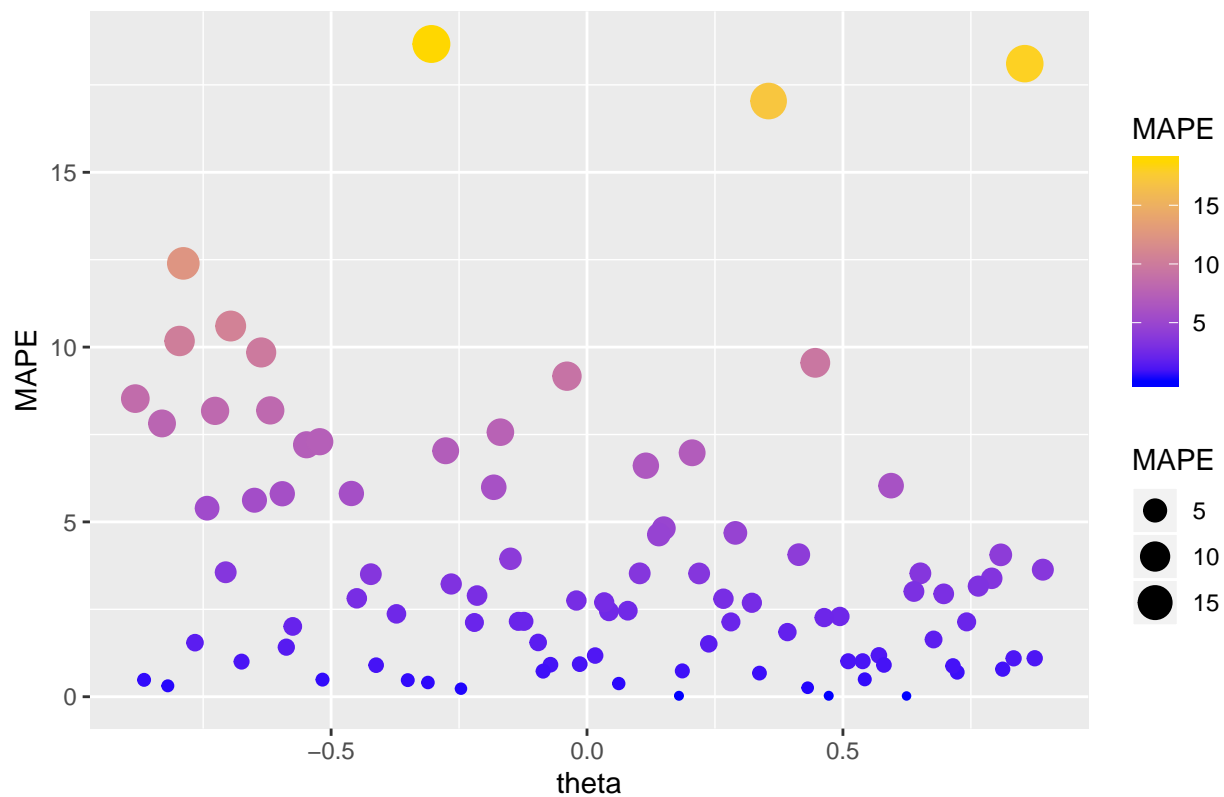


```
nonStatMatrix <- as.matrix(simNonStat[1:20,])  
heatmap(nonStatMatrix, scale='none')
```



```
ggplot(simNonStat, aes(x=theta, y=MAPE, color=MAPE, size=MAPE)) + geom_point() + scale_color_gradient(1
```

MAPE Values for Corresponding Theta Values – Non-Stationary



```
predData <- data.frame(theta=simNonStat[,1], predMAPE = predNonStat)
ggplot(predData, aes(x=theta, y=predMAPE, color=predMAPE, size=predMAPE)) + geom_point() + scale_color_
```

Predicted MAPE Values for Corresponding Theta Values – Non-Stationary

