VarLinearExample

Jackson Cates

9/26/2020

Libraries

```r
library(dplyr)
library(tsibble)
library(ggplot2)
library(feasts)
library(gridExtra)
library(MTS)
library(tseries)
```

Data Generation

$z_{1,t} = 0.8z_{1,t-1} + 0.6z_{2,t-1} + 0.1t + \epsilon_{1,t} + 0.12\epsilon_{1,t-1}$

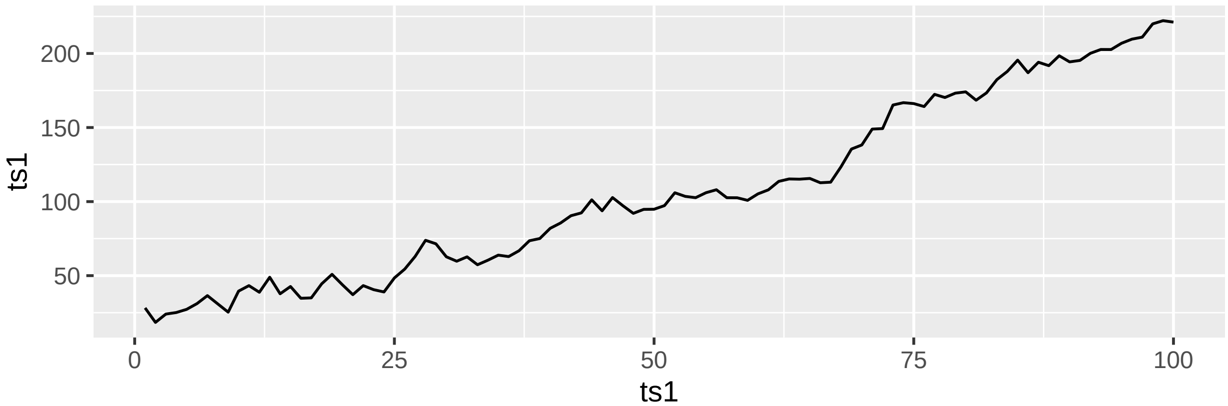
$z_{2,t} = 0.2z_{1,t-1} + 0.3z_{2,t-1} + \epsilon_{2,t} + 0.1\epsilon_{1,t-1} + 0.025\epsilon_{2,t-1}$

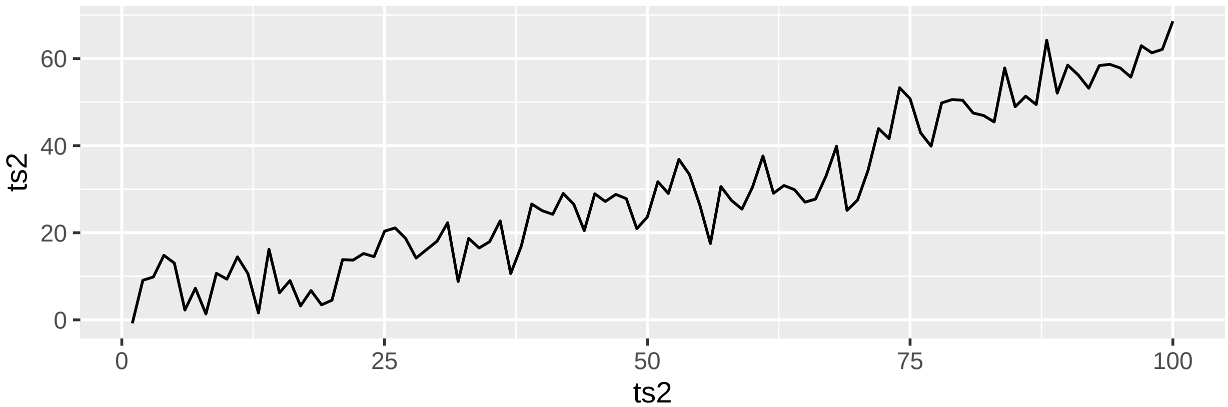
$z_t = \begin{pmatrix} 0.1 \\ 0 \end{pmatrix} t + \begin{pmatrix} 0.8 & 0.6 \\ 0.2 & 0.3 \end{pmatrix} z_{t-1} + \epsilon_t + \begin{pmatrix} 0.12 & 0 \\ 0.1 & 0.025 \end{pmatrix} \epsilon_{t-1}$

```r
set.seed(6)
skip = 10
length = 100
testLength = 20
noiseSd = 5

dataLength = skip + length + testLength

# Make some noise!
noise1 = rnorm(dataLength, 0, noiseSd)
noise2 = rnorm(dataLength, 0, noiseSd)

# Sets the first data point
ts1 = vector("numeric", length)
ts2 = vector("numeric", length)
ts1[1] = noise1[1]
ts2[1] = noise2[1]

# Loops though, makes linear data
for(t in 2:dataLength) {
  ts1[t] = 0.8*ts1[t-1] + 0.6*ts2[t-1] + 0.1*t + noise1[t] + 0.12*noise1[t-1]
  ts2[t] = 0.2*ts1[t-1] + 0.3*ts2[t-1] + noise2[t] + 0.1*noise1[t-1] + 0.025*noise2[t-1]
}
```

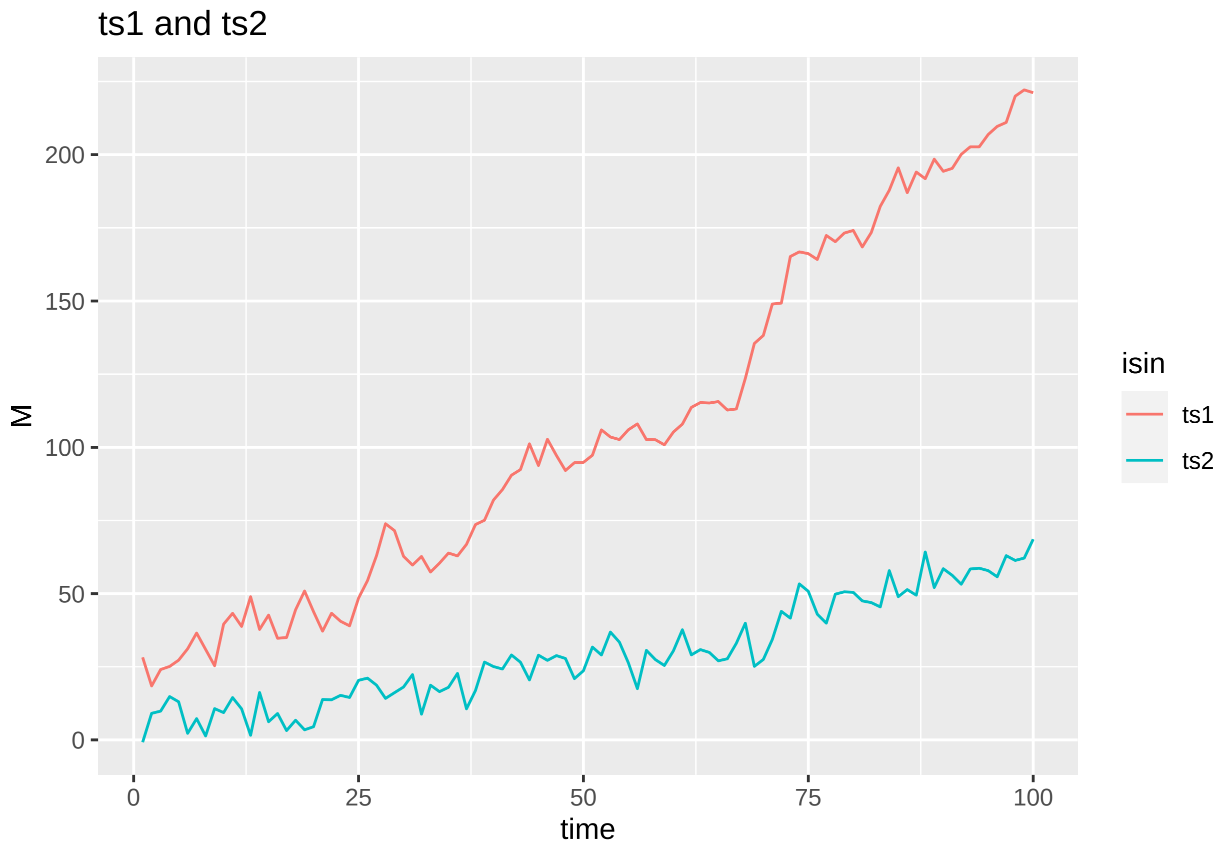
```r
# Takes out the testing data
test1 = ts1[(length + skip + 1):(dataLength)]
ts1 = ts1[(skip + 1):(length+skip)]
test2 = ts2[(length + skip + 1):(dataLength)]
ts2 = ts2[(skip + 1):(length+skip)]

# Turns them into a time series object
ts = as_tibble(ts1)
ts = rename(ts, "ts1" = "value")
ts[,2] = ts2
ts = rename(ts, "ts2" = "...2")
ts[,3] = 1:length
ts = rename(ts, "index" = "...3")

tsTest = as_tibble(test1)
tsTest = rename(tsTest, "ts1" = "value")
tsTest[,2] = test2
tsTest = rename(tsTest, "ts2" = "...2")
tsTest[,3] = (length + 1):(length + testLength)
tsTest = rename(tsTest, "index" = "...3")
tsTest = tsTest %>% as_tsibble(index = "index")

ts = ts %>% as_tsibble(index = "index")
```

```r
plot1 = ts %>% autoplot(ts1) + xlab("ts1")
plot2 = ts %>% autoplot(ts2) + xlab("ts2")
grid.arrange(plot1, plot2, nrow=2)
```






```
df1 = data.frame(time = ts$index, M = ts$ts1, isin = "ts1")
df2 = data.frame(time = ts$index, M = ts$ts2, isin = "ts2")
df = rbind(df1, df2)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("ts1 and ts2")
```

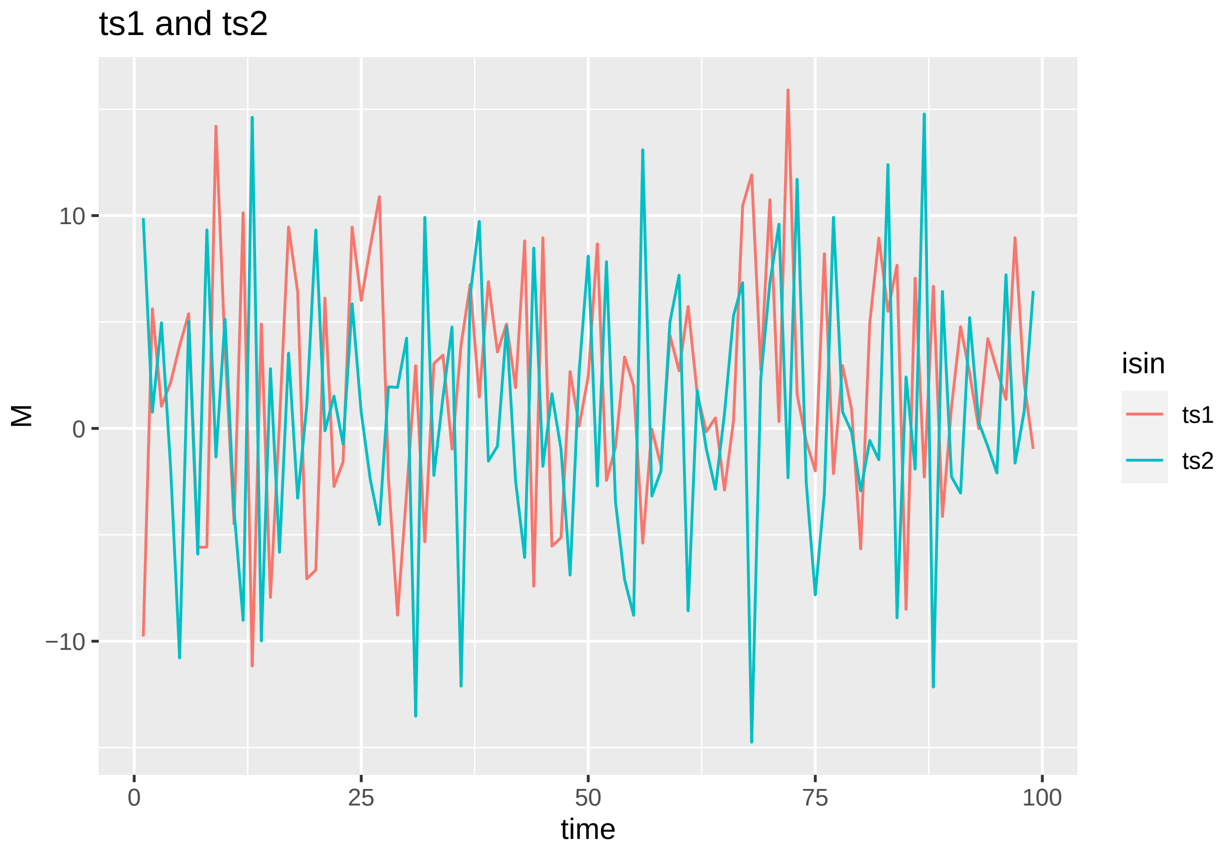


Differencing

We are going to take the change between consecutive observations

$y'_t = y_t - y_{t-1}$

```
# Takes a lagged difference of 1
tsDiff = ts %>% diffM()
tsDiff[,3] = 1:(length-1)
tsDiff = tsDiff %>% as_tibble()
tsDiff = tsDiff %>% as_tsibble(index = "index")

df1 = data.frame(time = tsDiff$index, M = tsDiff$ts1, isin = "ts1")
df2 = data.frame(time = tsDiff$index, M = tsDiff$ts2, isin = "ts2")
df = rbind(df1, df2)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("ts1 and ts2")
```




```r
kpss.test(tsDiff$ts1)
```

```
## Warning in kpss.test(tsDiff$ts1): p-value greater than printed p-value

##
##  KPSS Test for Level Stationarity
##
## data:  tsDiff$ts1
## KPSS Level = 0.18727, Truncation lag parameter = 3, p-value = 0.1
```

```r
kpss.test(tsDiff$ts2)
```

```
## Warning in kpss.test(tsDiff$ts2): p-value greater than printed p-value

##
##  KPSS Test for Level Stationarity
##
## data:  tsDiff$ts2
## KPSS Level = 0.049913, Truncation lag parameter = 3, p-value = 0.1
```

```r
VARorder(tsDiff[,-3])
```

```
## selected order: aic =  5
## selected order: bic =  1
## selected order: hq =  2
## Summary table:
##        p    AIC    BIC     HQ    M(p) p-value
##  [1,]  0 6.8313 6.8313 6.8313  0.0000  0.0000
##  [2,]  1 6.6063 6.7112 6.6488 25.2281  0.0000
```

```
##  [3,]  2 6.5534 6.7631 6.6382 10.7684  0.0293
##  [4,]  3 6.5594 6.8739 6.6867  5.8722  0.2089
##  [5,]  4 6.6225 7.0419 6.7921  1.3566  0.8517
##  [6,]  5 6.5477 7.0719 6.7598 11.5914  0.0207
##  [7,]  6 6.5624 7.1916 6.8170  4.7874  0.3098
##  [8,]  7 6.6232 7.3572 6.9202  1.4128  0.8420
##  [9,]  8 6.6631 7.5019 7.0025  2.8018  0.5915
## [10,]  9 6.6954 7.6391 7.0773  3.2241  0.5211
## [11,] 10 6.7526 7.8011 7.1768  1.5273  0.8218
## [12,] 11 6.6978 7.8512 7.1645  8.4738  0.0757
## [13,] 12 6.7360 7.9942 7.2451  2.5793  0.6305
## [14,] 13 6.7792 8.1422 7.3307  2.2016  0.6987
```

Fitting the model

$$z_t = \begin{pmatrix} 2.06 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 0.32 \\ 0.35 & -0.42 \end{pmatrix} z_{t-1} + \begin{pmatrix} -0.10 & 0 \\ 0 & -0.29 \end{pmatrix} z_{t-2} + a_t$$

```r
# Does LS estimation of the model
m1 = VAR(tsDiff[,-3], 2)
```

```
## Constant term:
## Estimates:  2.025321 0.5879369
## Std.Error:  0.6184937 0.6496756
## AR coefficient matrix
## AR( 1 )-matrix
##         [,1]   [,2]
## [1,] -0.0169  0.339
## [2,]  0.3315 -0.411
## standard error
##       [,1]   [,2]
## [1,] 0.107 0.0972
## [2,] 0.112 0.1021
## AR( 2 )-matrix
##         [,1]    [,2]
## [1,] -0.0882  0.0493
## [2,] -0.1230 -0.3291
## standard error
##       [,1]   [,2]
## [1,] 0.104 0.0979
## [2,] 0.110 0.1029
##
## Residuals cov-mtx:
##           [,1]      [,2]
## [1,] 26.084401 -7.081758
## [2,] -7.081758 28.780833
##
## det(SSE) =  700.5795
## AIC =  6.713524
## BIC =  6.923231
## HQ  =  6.798372
```

```r
m1R = refVAR(m1)
```

```
## Constant term:
## Estimates:  2.05642 0
```

```
## Std.Error:  0.5594857 0
## AR coefficient matrix
## AR( 1 )-matrix
##      [,1]   [,2]
## [1,] 0.00  0.328
## [2,] 0.35 -0.422
## standard error
##       [,1]   [,2]
## [1,] 0.000 0.0881
## [2,] 0.104 0.0973
## AR( 2 )-matrix
##        [,1]   [,2]
## [1,] -0.103  0.000
## [2,]  0.000 -0.294
## standard error
##        [,1]   [,2]
## [1,] 0.0993 0.0000
## [2,] 0.0000 0.0984
##
## Residuals cov-mtx:
##           [,1]      [,2]
## [1,] 26.156572 -7.129567
## [2,] -7.129567 29.275139
##
## det(SSE) =  714.9066
## AIC =  6.673162
## BIC =  6.804229
## HQ  =  6.726192
```

Model Checking

Multivariate Portmanteau Statistics

Let R_ℓ be the theoretical lag ℓ cross-correlation matrix of innovation a_t

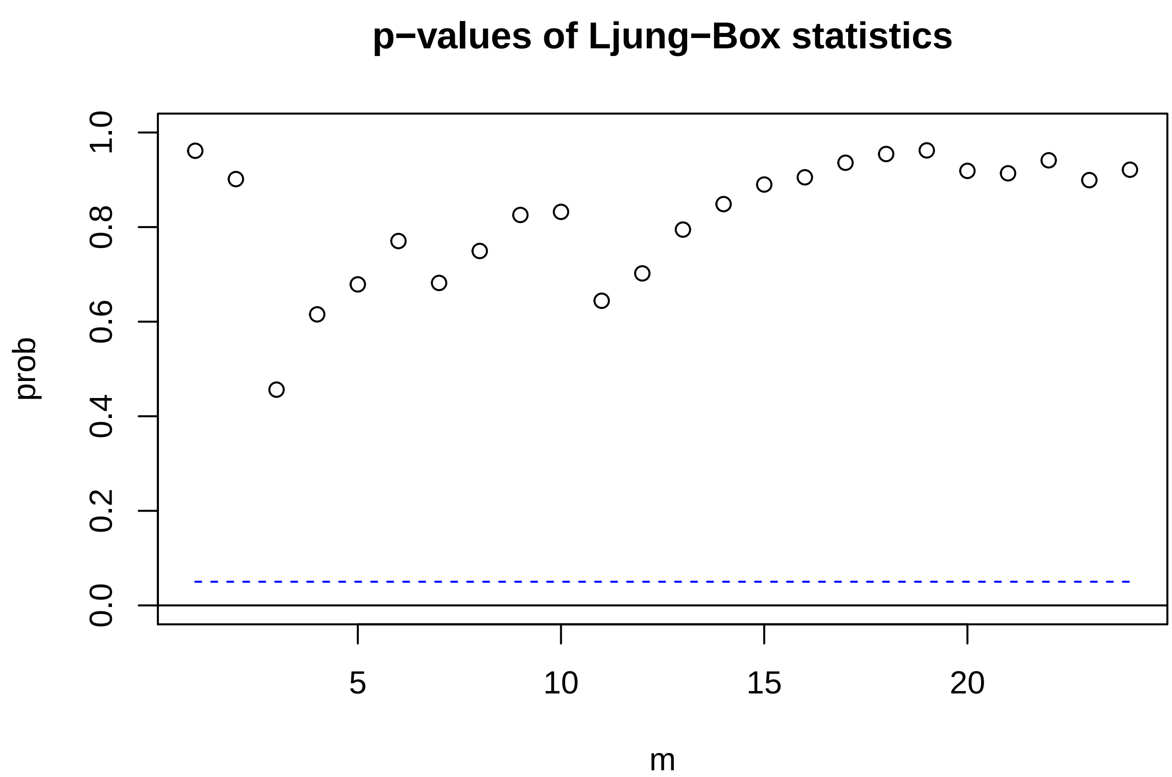
H_0: $R_1 = \cdots = R_m = 0$

H_A: $R_j \neq 0$ for some $1 \leq j \leq m$

```
mq(m1R$residuals)
```

```
## Ljung-Box Statistics:
##          m       Q(m)     df    p-value
##  [1,]  1.000     0.615   4.000     0.96
##  [2,]  2.000     3.470   8.000     0.90
##  [3,]  3.000    11.869  12.000     0.46
##  [4,]  4.000    13.775  16.000     0.62
##  [5,]  5.000    16.598  20.000     0.68
##  [6,]  6.000    18.647  24.000     0.77
##  [7,]  7.000    23.993  28.000     0.68
##  [8,]  8.000    26.316  32.000     0.75
##  [9,]  9.000    28.043  36.000     0.83
## [10,] 10.000    31.412  40.000     0.83
## [11,] 11.000    39.988  44.000     0.64
## [12,] 12.000    42.365  48.000     0.70
## [13,] 13.000    43.451  52.000     0.79
```

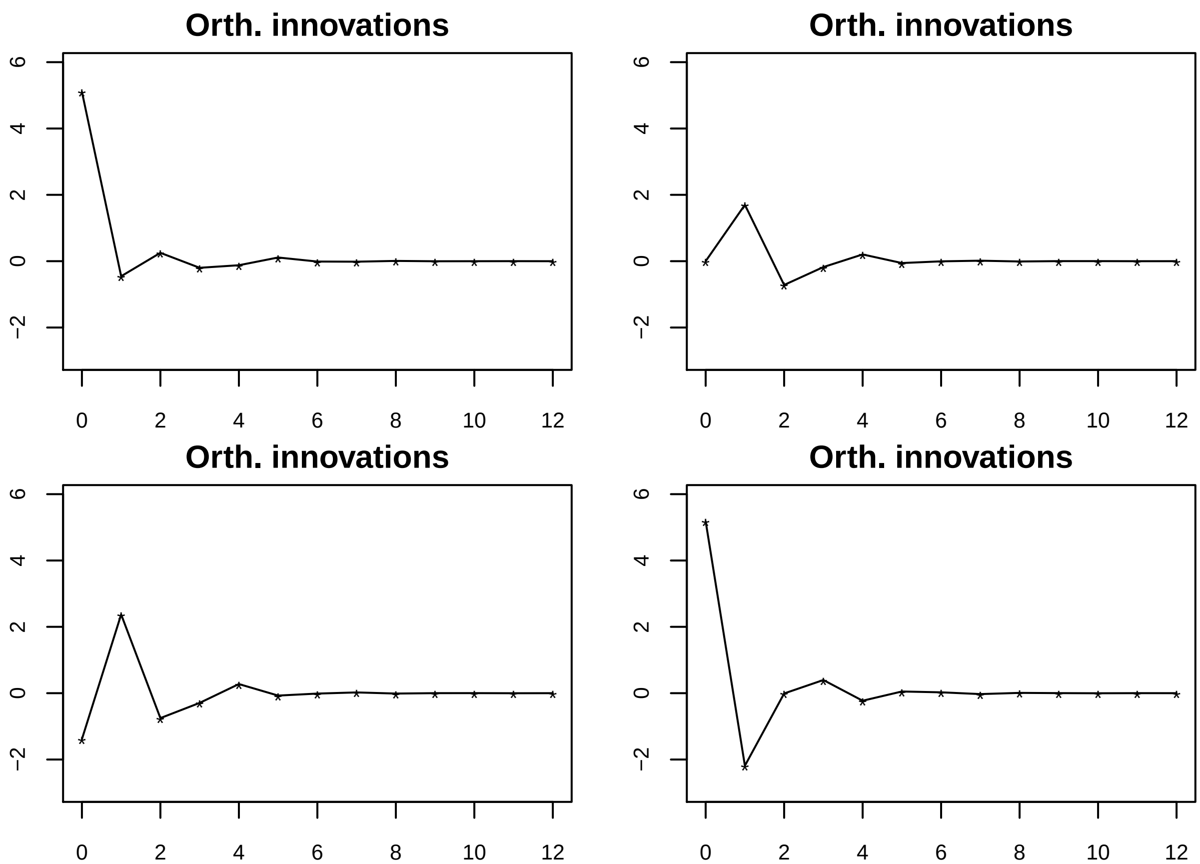
```
## [14,] 14.000    45.201  56.000     0.85
## [15,] 15.000    46.968  60.000     0.89
## [16,] 16.000    49.704  64.000     0.91
## [17,] 17.000    51.191  68.000     0.94
## [18,] 18.000    53.017  72.000     0.95
## [19,] 19.000    55.592  76.000     0.96
## [20,] 20.000    63.017  80.000     0.92
## [21,] 21.000    66.953  84.000     0.91
## [22,] 22.000    68.259  88.000     0.94
## [23,] 23.000    75.149  92.000     0.90
## [24,] 24.000    77.132  96.000     0.92
```



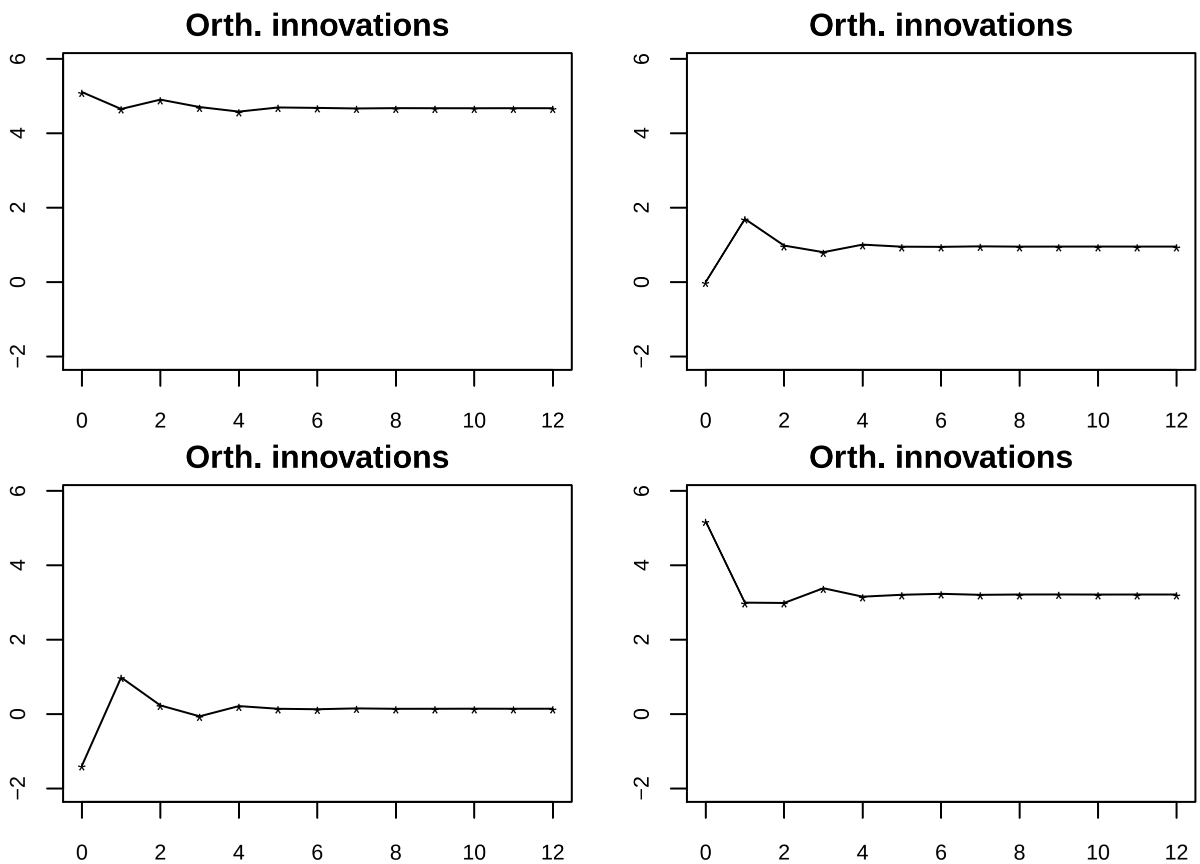
Impulse

The first graph is the impulse response, while the second is accumulated response.

```
VARirf(m1R$Phi, m1$Sigma)
```




```
## Press return to continue
```



Model Forecasting

```
pred = VARpred(m1R, h = testLength)
```

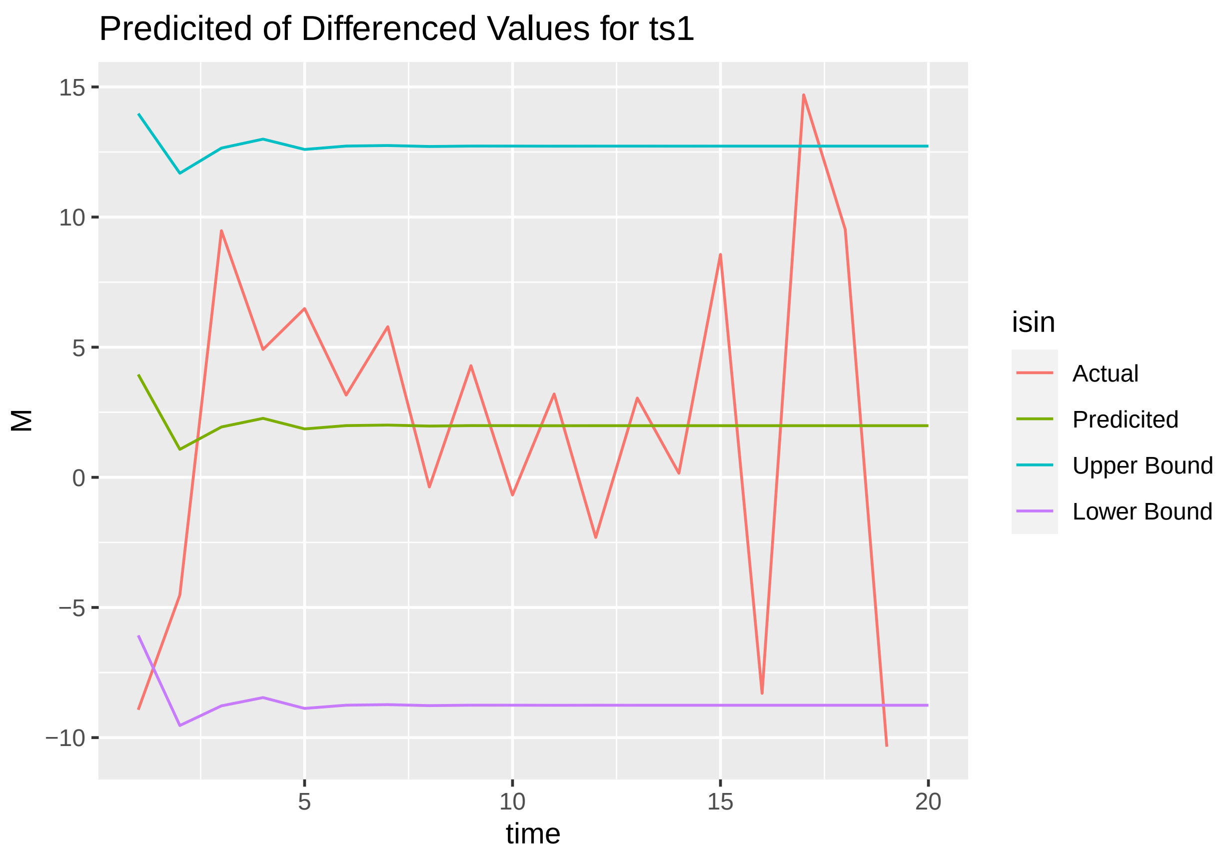
```
## orig  99
## Forecasts at origin:  99
##          ts1       ts2
##  [1,] 3.952 -3.298648
##  [2,] 1.074  0.873148
##  [3,] 1.936  0.978617
##  [4,] 2.267  0.007245
##  [5,] 1.859  0.501876
##  [6,] 1.988  0.436695
##  [7,] 2.008  0.363385
##  [8,] 1.971  0.420701
##  [9,] 1.988  0.405076
## [10,] 1.986  0.400628
## [11,] 1.983  0.406654
## [12,] 1.985  0.404310
## [13,] 1.985  0.404263
## [14,] 1.984  0.404818
## [15,] 1.985  0.404516
## [16,] 1.985  0.404560
## [17,] 1.985  0.404604
## [18,] 1.985  0.404569
## [19,] 1.985  0.404579
```

```
## [20,] 1.985  0.404581
## Standard Errors of predictions:
##        [,1]  [,2]
##  [1,] 5.114 5.411
##  [2,] 5.413 6.308
##  [3,] 5.467 6.353
##  [4,] 5.474 6.372
##  [5,] 5.479 6.383
##  [6,] 5.480 6.383
##  [7,] 5.480 6.383
##  [8,] 5.480 6.383
##  [9,] 5.480 6.383
## [10,] 5.480 6.383
## [11,] 5.480 6.383
## [12,] 5.480 6.383
## [13,] 5.480 6.383
## [14,] 5.480 6.383
## [15,] 5.480 6.383
## [16,] 5.480 6.383
## [17,] 5.480 6.383
## [18,] 5.480 6.383
## [19,] 5.480 6.383
## [20,] 5.480 6.383
## Root mean square errors of predictions:
##        [,1]  [,2]
##  [1,] 5.242 5.546
##  [2,] 5.835 7.462
##  [3,] 5.547 6.420
##  [4,] 5.484 6.402
##  [5,] 5.487 6.398
##  [6,] 5.482 6.384
##  [7,] 5.480 6.383
##  [8,] 5.480 6.383
##  [9,] 5.480 6.383
## [10,] 5.480 6.383
## [11,] 5.480 6.383
## [12,] 5.480 6.383
## [13,] 5.480 6.383
## [14,] 5.480 6.383
## [15,] 5.480 6.383
## [16,] 5.480 6.383
## [17,] 5.480 6.383
## [18,] 5.480 6.383
## [19,] 5.480 6.383
## [20,] 5.480 6.383
```

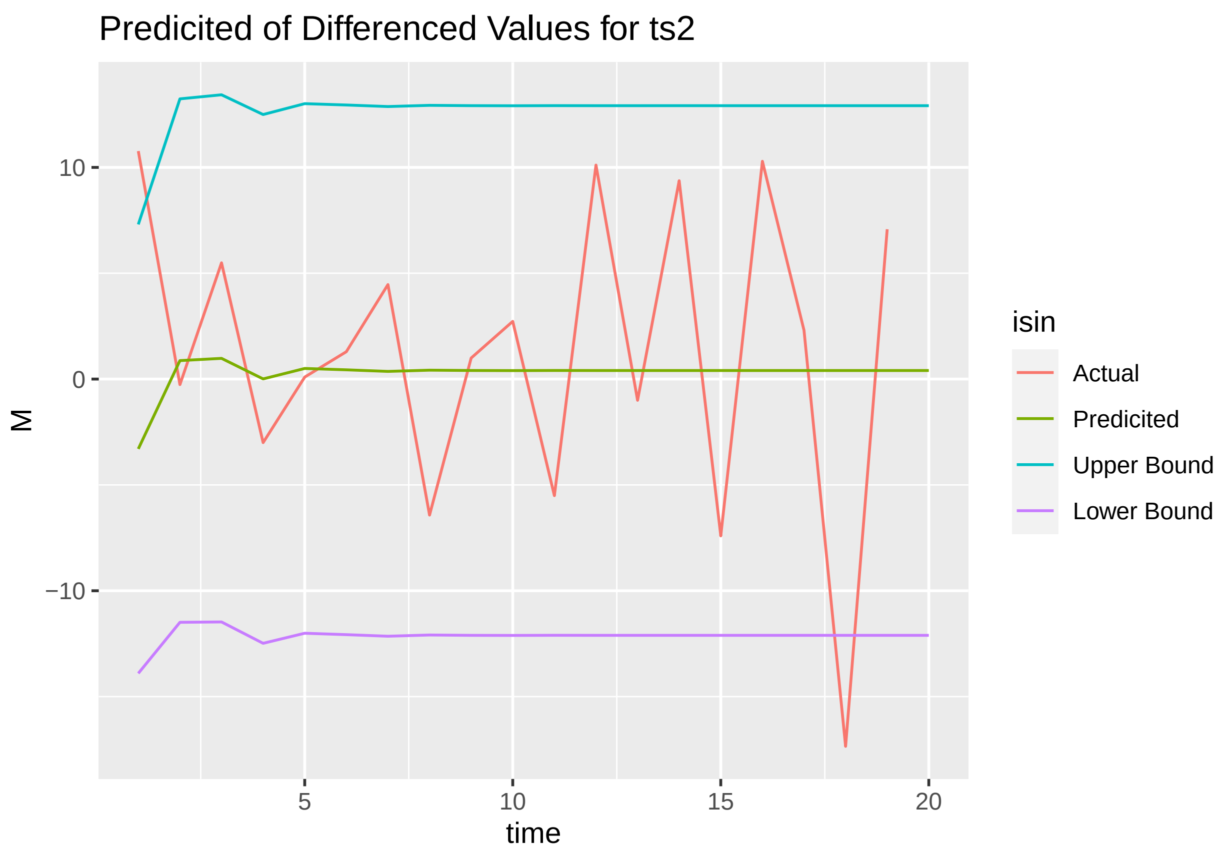
```r
# Calculates the confidence interval
upperConfDiff = pred$pred + 1.96 * pred$se.err
lowerConfDiff = pred$pred - 1.96 * pred$se.err
```

```r
testDiff = tsTest %>% diffM()
testDiff[,3] = 1:(testLength-1)
testDiff = testDiff %>% as_tibble()
testDiff = testDiff %>% as_tsibble(index = "index")
```

```r
tsPredDiff = as_tibble(pred$pred[,1])
tsPredDiff = rename(tsPredDiff, "ts1" = "value")
tsPredDiff[,2] = pred$pred[,2]
tsPredDiff = rename(tsPredDiff, "ts2" = "...2")
tsPredDiff[,3] = 1:testLength
tsPredDiff = rename(tsPredDiff, "index" = "...3")
tsPredDiff = tsPredDiff %>% as_tsibble(index = "index")

df1 = data.frame(time = testDiff$index, M = testDiff$ts1, isin = "Actual")
df2 = data.frame(time = tsPredDiff$index, M = tsPredDiff$ts1, isin = "Predicited")
df3 = data.frame(time = 1:testLength, M = upperConfDiff[,1], isin = "Upper Bound")
df4 = data.frame(time = 1:testLength, M = lowerConfDiff[,1], isin = "Lower Bound")
df = rbind(df1, df2, df3, df4)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicited of Differenced Value
```




```r
df1 = data.frame(time = testDiff$index, M = testDiff$ts2, isin = "Actual")
df2 = data.frame(time = tsPredDiff$index, M = tsPredDiff$ts2, isin = "Predicited")
df3 = data.frame(time = 1:testLength, M = upperConfDiff[,2], isin = "Upper Bound")
df4 = data.frame(time = 1:testLength, M = lowerConfDiff[,2], isin = "Lower Bound")
df = rbind(df1, df2, df3, df4)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicited of Differenced Value
```

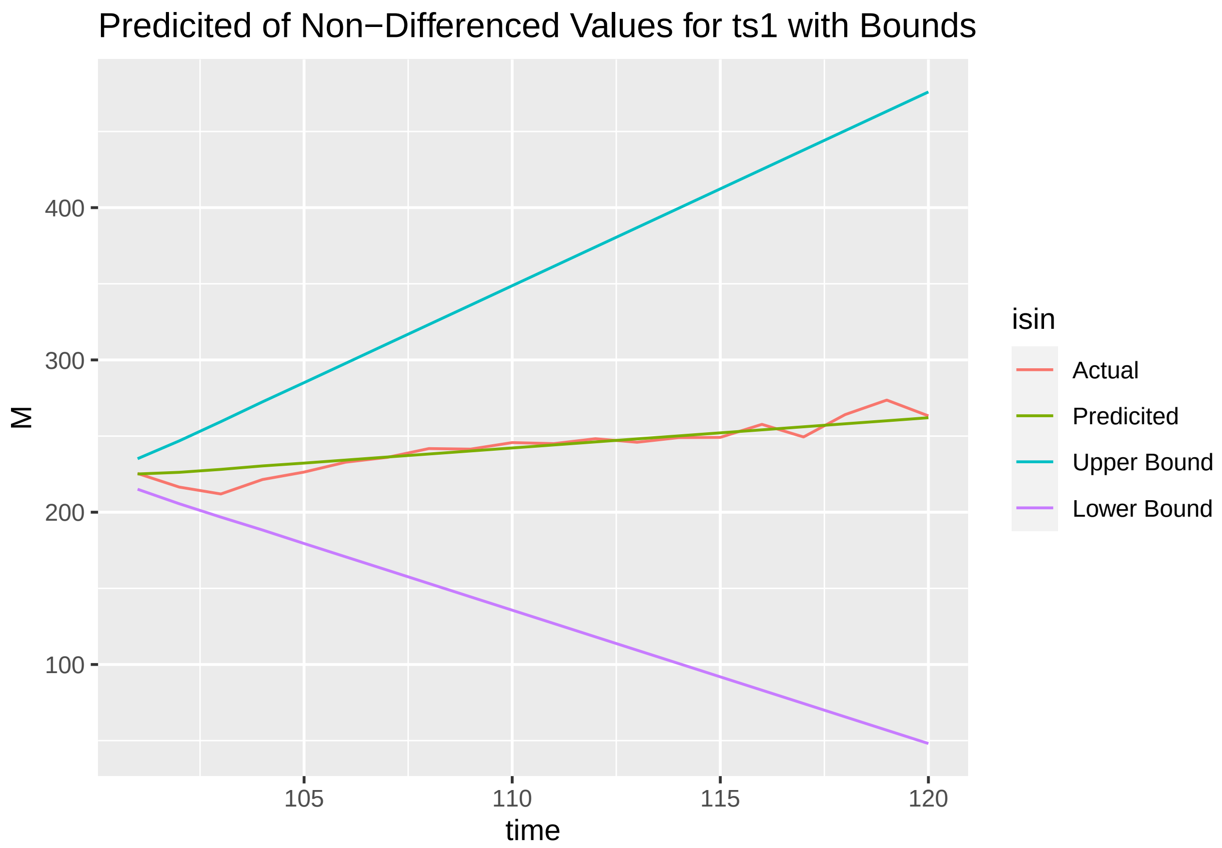


Inverting the Differencing

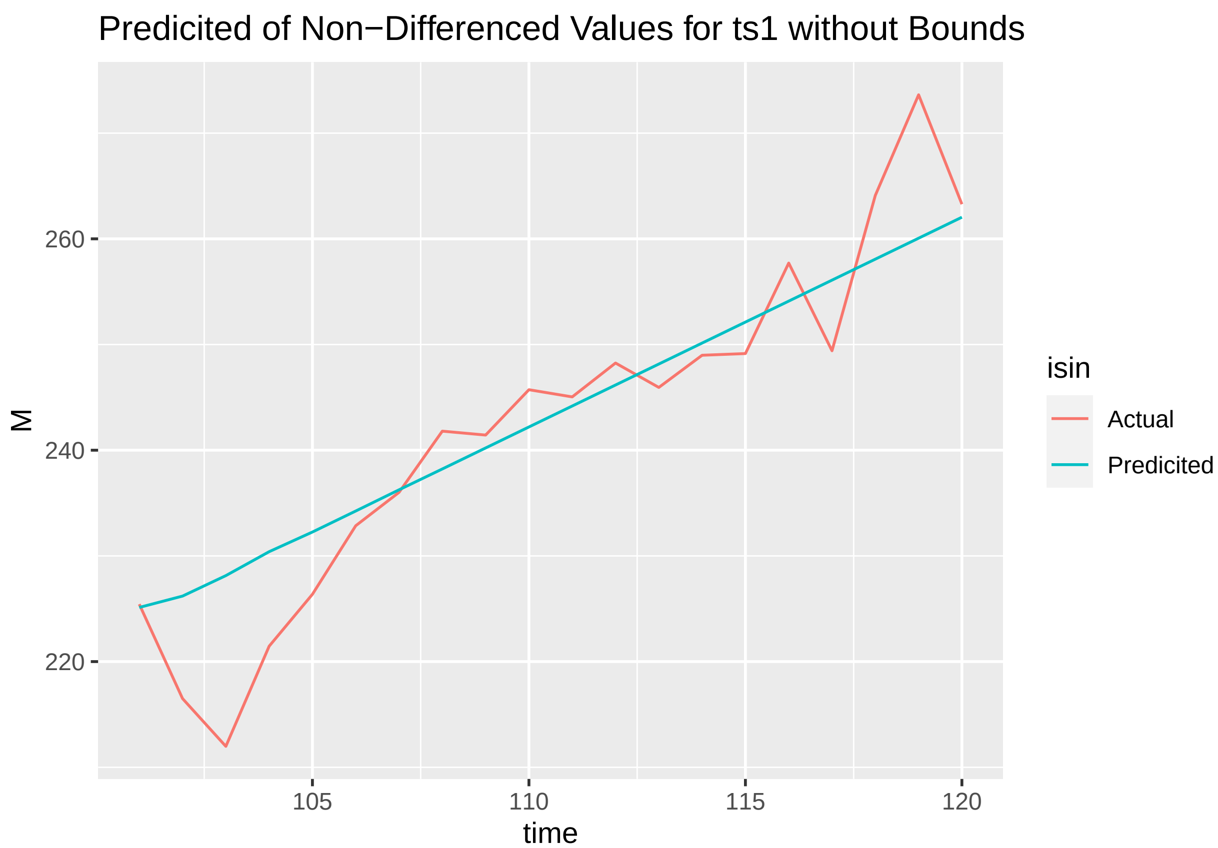
$y_t = \sum_{i=1}^{t} y_i\prime + y_0$

```
# Accumlates all of the transfomred values
tsPred = tsPredDiff[,-3] %>% cumsum() %>% as_tibble()
upperConf = matrix(nrow = testLength, ncol = 2)
upperConf[,1] = upperConfDiff[,1] %>% cumsum()
upperConf[,2] = upperConfDiff[,2] %>% cumsum()
lowerConf = matrix(nrow = testLength, ncol = 2)
lowerConf[,1] = lowerConfDiff[,1] %>% cumsum()
lowerConf[,2] = lowerConfDiff[,2] %>% cumsum()

# Adds the intercepts
tsPred$ts1 = tsPred$ts1 + ts$ts1[length]
tsPred$ts2 = tsPred$ts2 + ts$ts2[length]
tsPred[,3] = (length + 1):(length + testLength)
tsPred = rename(tsPred, "index" = "...3")
tsPred = tsPred %>% as_tsibble(index = "index")
upperConf[,1] = upperConf[,1] + ts$ts1[length]
upperConf[,2] = upperConf[,2] + ts$ts2[length]
lowerConf[,1] = lowerConf[,1] + ts$ts1[length]
lowerConf[,2] = lowerConf[,2] + ts$ts2[length]

df1 = data.frame(time = tsTest$index, M = tsTest$ts1, isin = "Actual")
df2 = data.frame(time = tsPred$index, M = tsPred$ts1, isin = "Predicited")
```

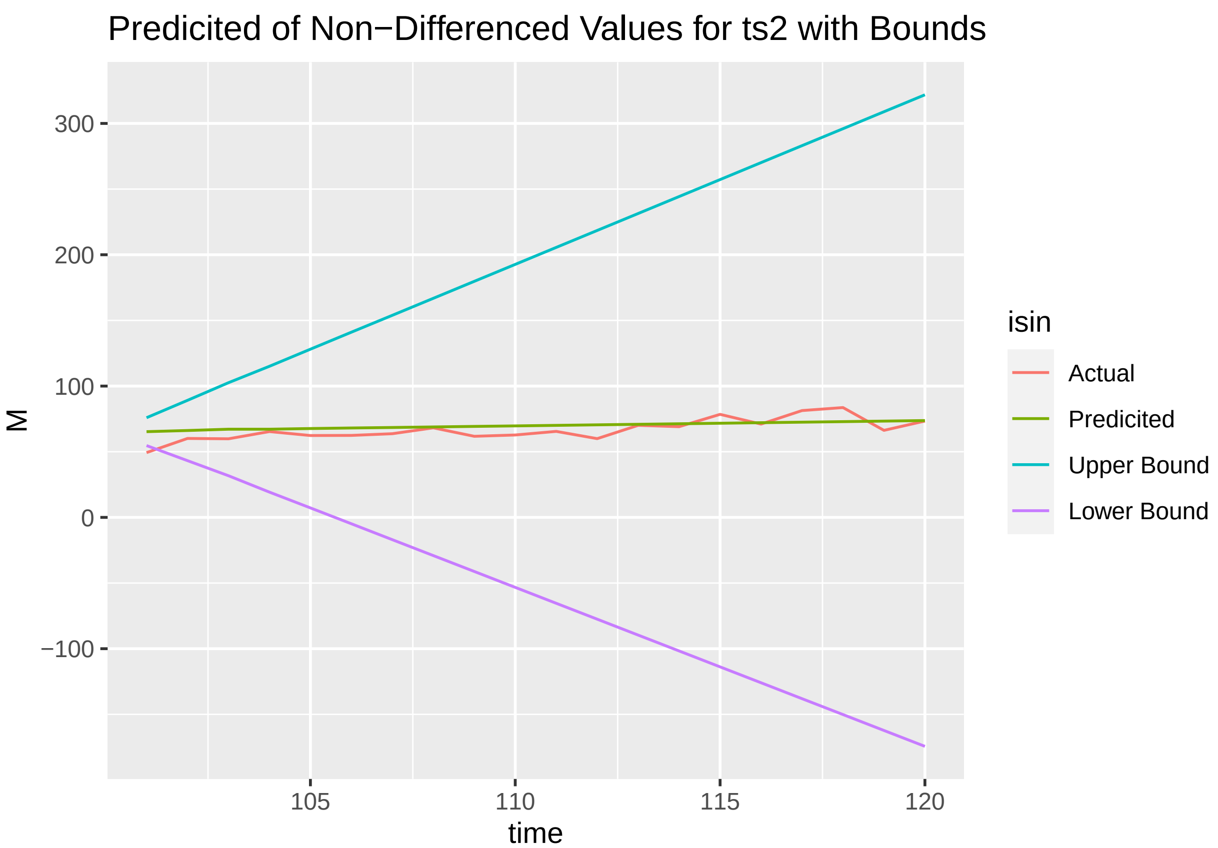
```
df3 = data.frame(time = (length+1):(length+testLength), M = upperConf[,1], isin = "Upper Bound")
df4 = data.frame(time = (length+1):(length + testLength), M = lowerConf[,1], isin = "Lower Bound")
df = rbind(df1, df2, df3, df4)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicited of Non-Differenced Va
```



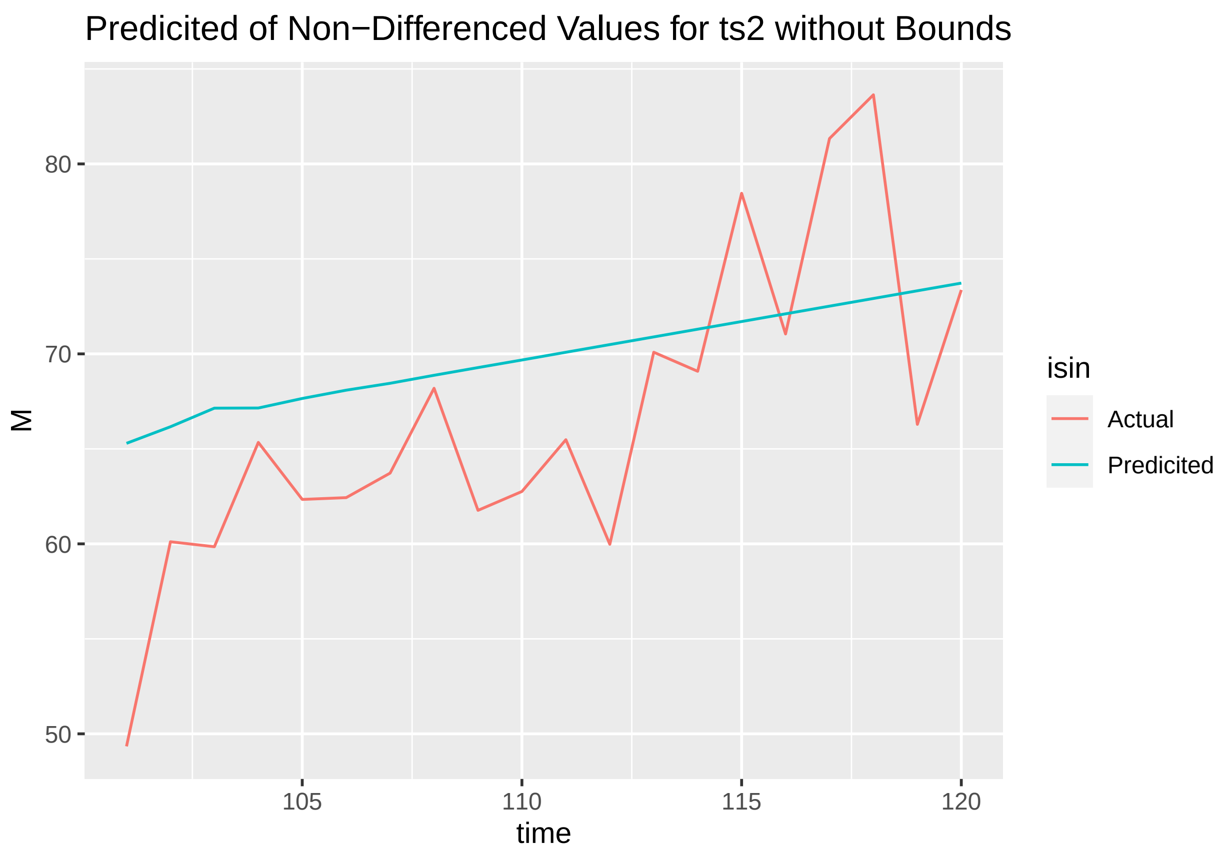

```
df = rbind(df1, df2)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicited of Non-Differenced Va
```



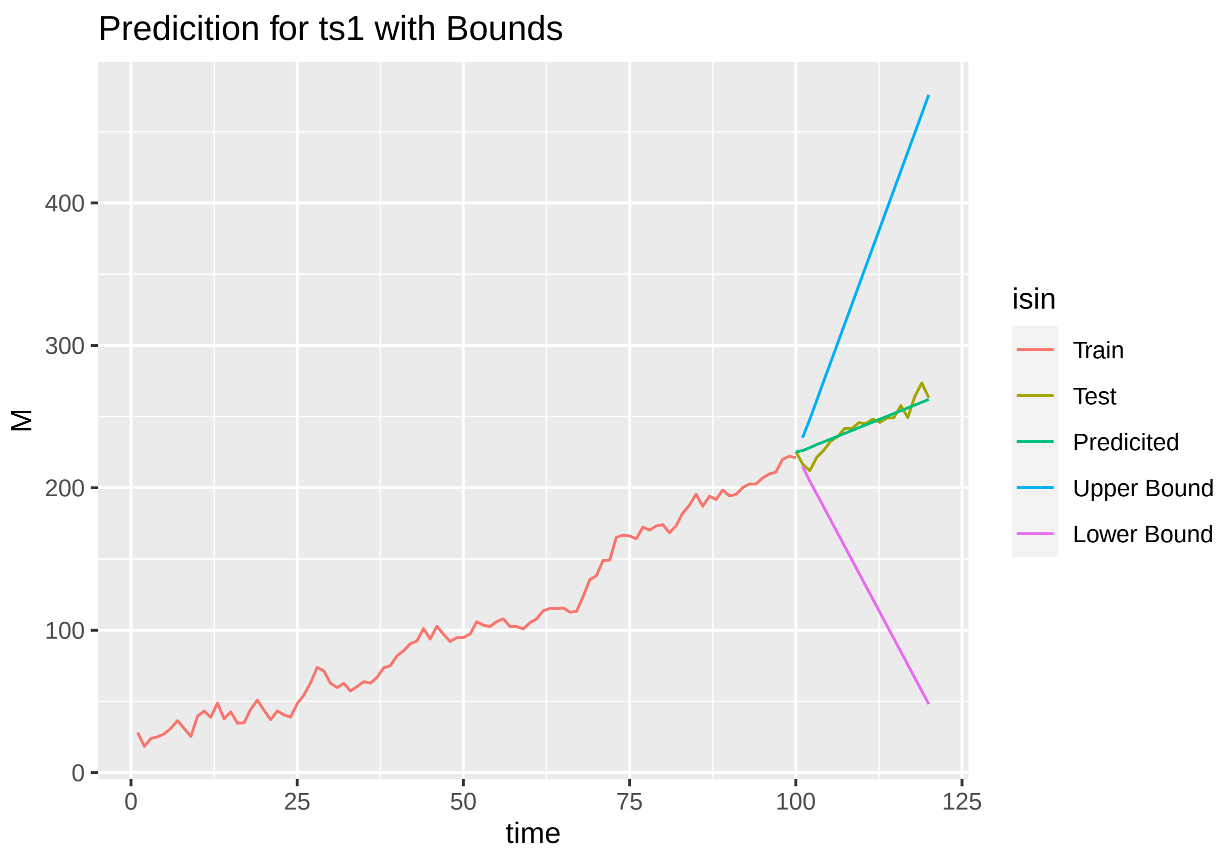

```
df1 = data.frame(time = tsTest$index, M = tsTest$ts2, isin = "Actual")
df2 = data.frame(time = tsPred$index, M = tsPred$ts2, isin = "Predicited")
df3 = data.frame(time = (length+1):(length+testLength), M = upperConf[,2], isin = "Upper Bound")
df4 = data.frame(time = (length+1):(length + testLength), M = lowerConf[,2], isin = "Lower Bound")
df = rbind(df1, df2, df3, df4)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicited of Non-Differenced Va
```



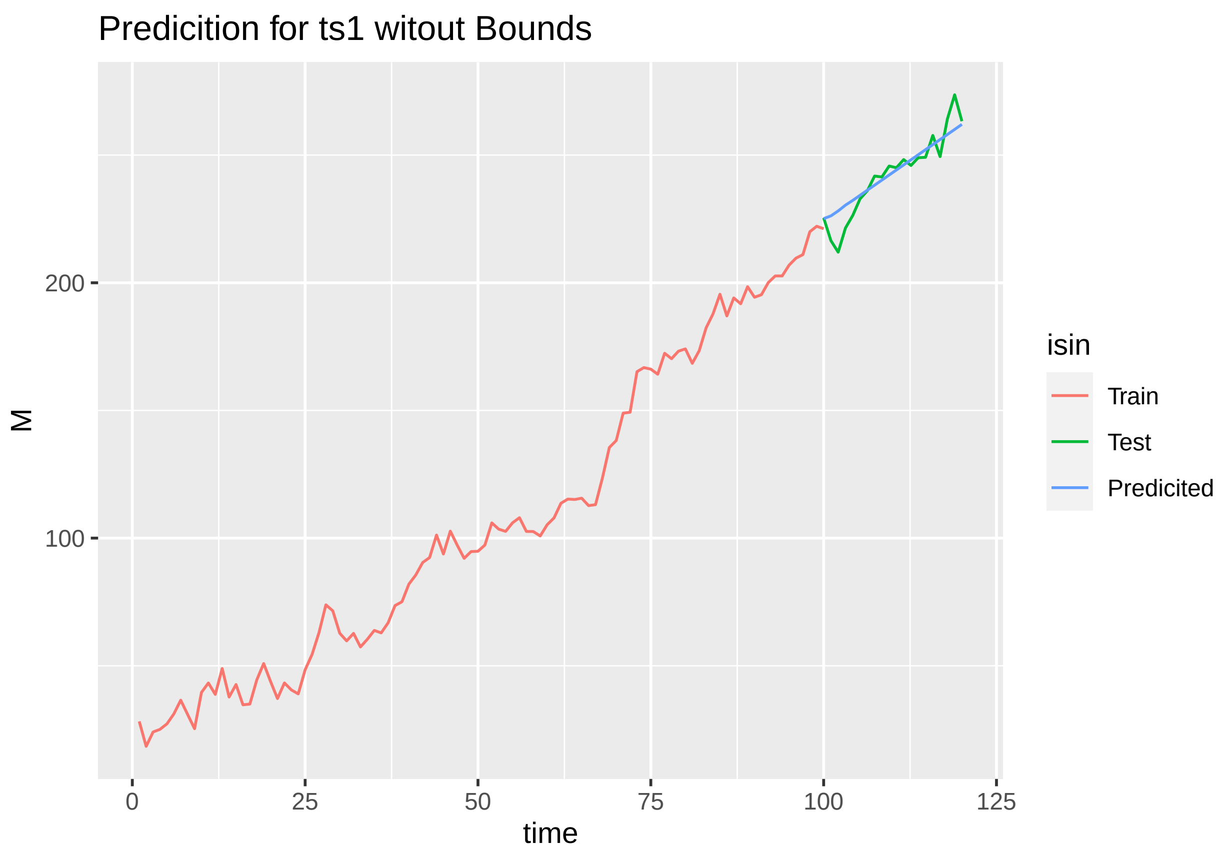

```r
df = rbind(df1, df2)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicited of Non-Differenced Va
```



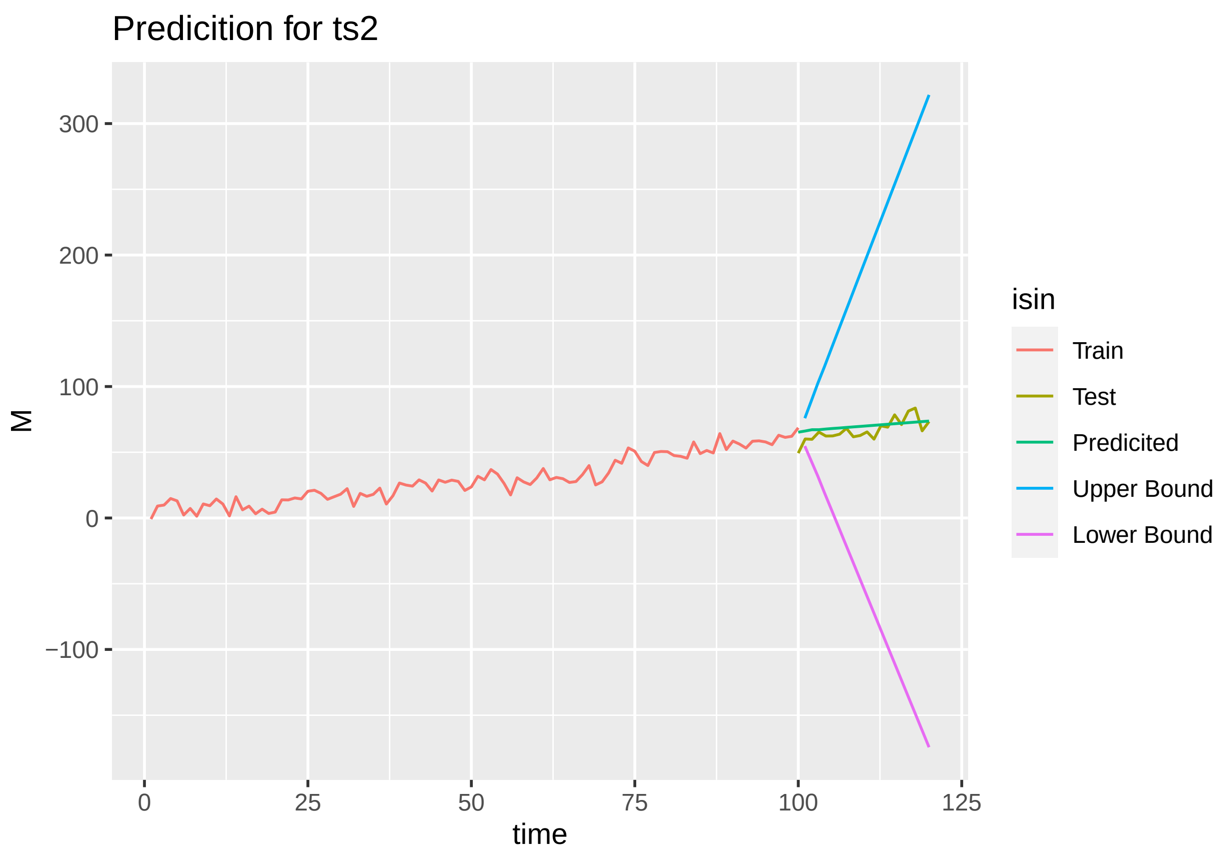

```
drawStart = 1

df1 = data.frame(time = seq(drawStart, length, length=length), M = ts$ts1, isin = "Train")
df2 = data.frame(time = seq(length, length + testLength, length=testLength), M = tsTest$ts1, isin = "Te
df3 = data.frame(time = seq(length, length + testLength, length=testLength), M = tsPred$ts1, isin = "Pr
df4 = data.frame(time = (length+1):(length+testLength), M = upperConf[,1], isin = "Upper Bound")
df5 = data.frame(time = (length+1):(length + testLength), M = lowerConf[,1], isin = "Lower Bound")
df = rbind(df1, df2, df3, df4, df5)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicition for ts1 with Bounds
```



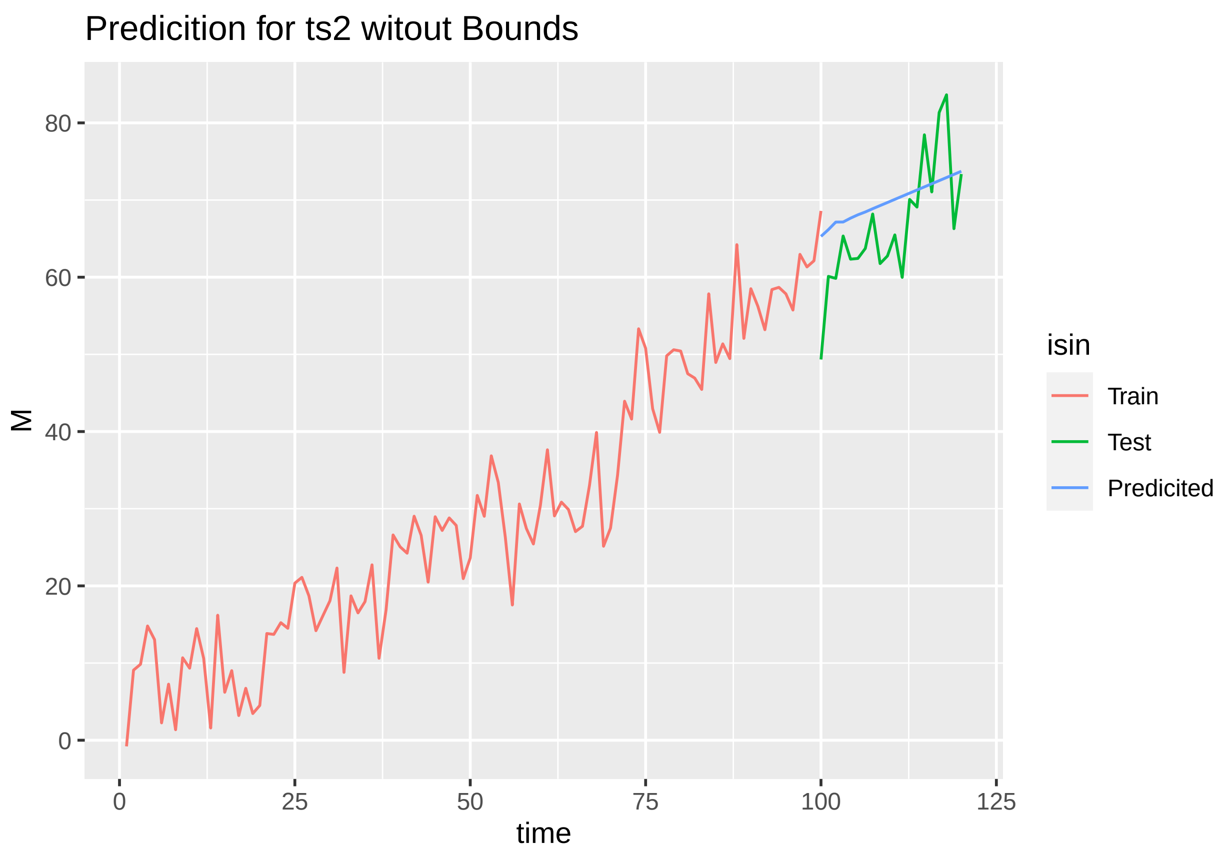

```
df = rbind(df1, df2, df3)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicition for ts1 witout Bounc
```




```
df1 = data.frame(time = seq(drawStart, length, length=length), M = ts$ts2, isin = "Train")
df2 = data.frame(time = seq(length, length + testLength, length=testLength), M = tsTest$ts2, isin = "Te
df3 = data.frame(time = seq(length, length + testLength, length=testLength), M = tsPred$ts2, isin = "Pr
df4 = data.frame(time = (length+1):(length+testLength), M = upperConf[,2], isin = "Upper Bound")
df5 = data.frame(time = (length+1):(length + testLength), M = lowerConf[,2], isin = "Lower Bound")
df = rbind(df1, df2, df3, df4, df5)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicition for ts2")
```




```
df = rbind(df1, df2, df3)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicition for ts2 witout Bound
```



MAE

```r
mean(abs(tsTest$ts1 - tsPred$ts1))
```

```
## [1] 4.567337
```

```r
mean(abs(tsTest$ts2 - tsPred$ts2))
```

```
## [1] 5.741117
```

MSE

```r
mean((tsTest$ts1 - tsPred$ts1)*(tsTest$ts1 - tsPred$ts1))
```

```
## [1] 39.9089
```

```r
mean((tsTest$ts2 - tsPred$ts2)*(tsTest$ts2 - tsPred$ts2))
```

```
## [1] 48.07554
```