

VAR Stationary Example

Jackson Cates

9/15/2020

Libraries

```
library(dplyr)
library(tsibble)
library(ggplot2)
library(feasts)
library(gridExtra)
library(MTS)
library(dse)
```

Data Generation

The model I am going to simulate is as follows:

$$ts1_t = 0.3 * ts1_{t-1} + \epsilon_1$$

$$ts2_t = 0.4 * ts2_{t-1} + ts1_{t-1} + \epsilon_2$$

Which results in the following model

$$\begin{pmatrix} ts1_t \\ ts2_t \end{pmatrix} = \begin{pmatrix} 0.3 & 0 \\ 1 & 0.4 \end{pmatrix} \begin{pmatrix} ts1_{t-1} \\ ts2_{t-1} \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \end{pmatrix}$$

Generates the two vectors

```
set.seed(14)
length = 320
testLength = 20
ts1 = vector("numeric", length)
noise = makeTSnoise(length, 1, 0)$w
```

Simluates the model

```
ts1[1] = noise[1]
for(t in 2:length) {
  ts1[t] = 0.3*ts1[t - 1] + noise[t]
}
```

```
ts2 = vector("numeric", length)
noise = makeTSnoise(length, 1, 0)$w
ts2[1] = noise[1]
for(t in 2:length) {
  ts2[t] = 0.4*ts2[t - 1] + ts1[t - 1] + noise[t]
}
```

Takes out the testing data

```
test1 = ts1[(length-testLength):(length-1)]
```

```

ts1 = ts1[1:(length-testLength)]
test2 = ts2[(length-testLength):(length-1)]
ts2 = ts2[1:(length-testLength)]

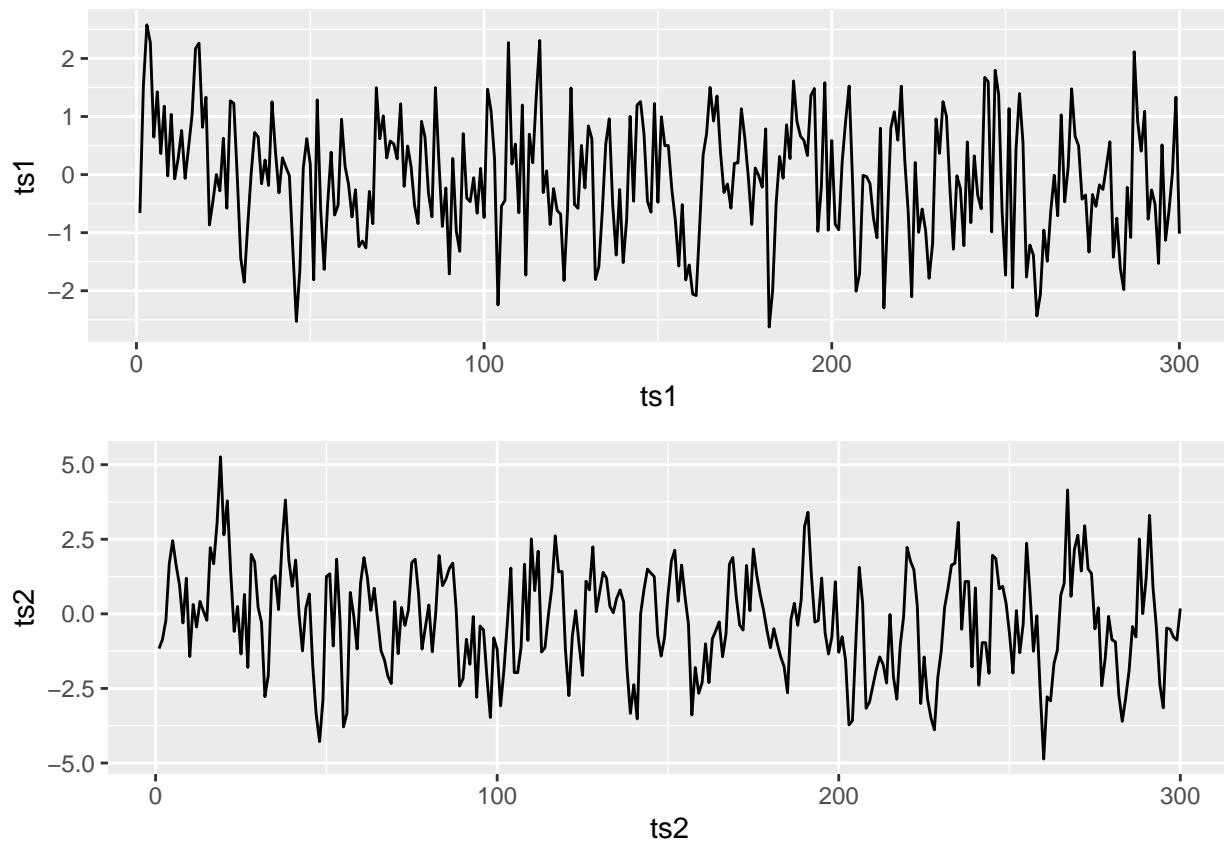
length = length - testLength

# 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")

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

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

```



Model Selection

We are going to try to select a $VAR(p)$ model. The model is:

$$z_t = \phi_0 + \sum_{i=1}^p \phi_i z_{t-i} + a_t$$

- k is the number of time series we have
- ϕ_0 is a k dimensional constant vector
- ϕ_i is a k by k matrix
- a_t is a sequence of independent and identically distributed random vectors with mean zero and covariance matrix Σ_a

Order Selection with the Sequential likelihood ratio test:

What we are going to do for our order selection is compare $VAR(\ell)$ with $VAR(\ell - 1)$

$$H_0: \phi_\ell = 0$$

$$H_A: \phi_\ell \neq 0$$

Does the order test

```
VARorder(ts[, -3])
```

```
## selected order: aic = 1
## selected order: bic = 1
## selected order: hq = 1
## Summary table:
##      p      AIC      BIC      HQ      M(p) p-value
## [1,] 0 1.1326 1.1326 1.1326 0.0000 0.0000
## [2,] 1 0.1671 0.2165 0.1869 281.2862 0.0000
## [3,] 2 0.1925 0.2913 0.2320 0.3591 0.9857
## [4,] 3 0.2113 0.3595 0.2706 2.1968 0.6996
## [5,] 4 0.1991 0.3967 0.2782 10.7857 0.0291
## [6,] 5 0.2206 0.4675 0.3194 1.4218 0.8404
## [7,] 6 0.2347 0.5310 0.3533 3.4420 0.4868
## [8,] 7 0.2399 0.5856 0.3782 5.8303 0.2122
## [9,] 8 0.2621 0.6572 0.4203 1.1907 0.8796
## [10,] 9 0.2770 0.7214 0.4549 3.1638 0.5308
## [11,] 10 0.2818 0.7757 0.4795 5.7926 0.2152
## [12,] 11 0.2945 0.8378 0.5119 3.6809 0.4509
## [13,] 12 0.2943 0.8869 0.5315 7.0347 0.1341
## [14,] 13 0.3159 0.9579 0.5728 1.3239 0.8573
```

As seen above, we should proceed with a $VAR(1)$ model

Fitting the model

```
# Does LS estimation of the model
m1 = VAR(ts[,-3], 1)

## Constant term:
## Estimates:  -0.0395207 -0.05098208
## Std.Error:  0.05796975 0.06336133
## AR coefficient matrix
## AR( 1 )-matrix
##      [,1] [,2]
## [1,] 0.296 0.0289
## [2,] 0.929 0.4456
## standard error
##      [,1] [,2]
## [1,] 0.0567 0.0338
## [2,] 0.0619 0.0370
##
## Residuals cov-mtx:
##      [,1] [,2]
## [1,] 0.97981832 -0.01086404
## [2,] -0.01086404 1.17055340
##
## det(SSE) = 1.146812
## AIC = 0.1636523
## BIC = 0.213036
## HQ  = 0.1834157
```

From the output above, we get the following model:

$$z_t = \begin{pmatrix} -0.04 \\ -0.05 \end{pmatrix} + \begin{pmatrix} 0.30 & 0.03 \\ 0.929 & 0.44 \end{pmatrix} z_{t-1} + a_t$$

Some things to note:

- Granger Causality is low from ts2 to ts1 (with value of 0.04)
- Coefficients are really similar to our simulation

Model Checking

Stationarity

It turns out that to test if a series is stationary, we can solve the following determinate:

$$|I_k - \Phi_1 B| = 0$$

and if the absolute value of the solutions are greater than 1, it is stationary!

For $VAR(1)$ models, the solutions of B is simply the reciprocal of the eigenvalues of Φ_1

```
eigen(m1$Phi)[1]
```

```
## $values  
## [1] 0.5506652 0.1904566
```

So our series is stationary!

Multivariate Portmanteau Statistics

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

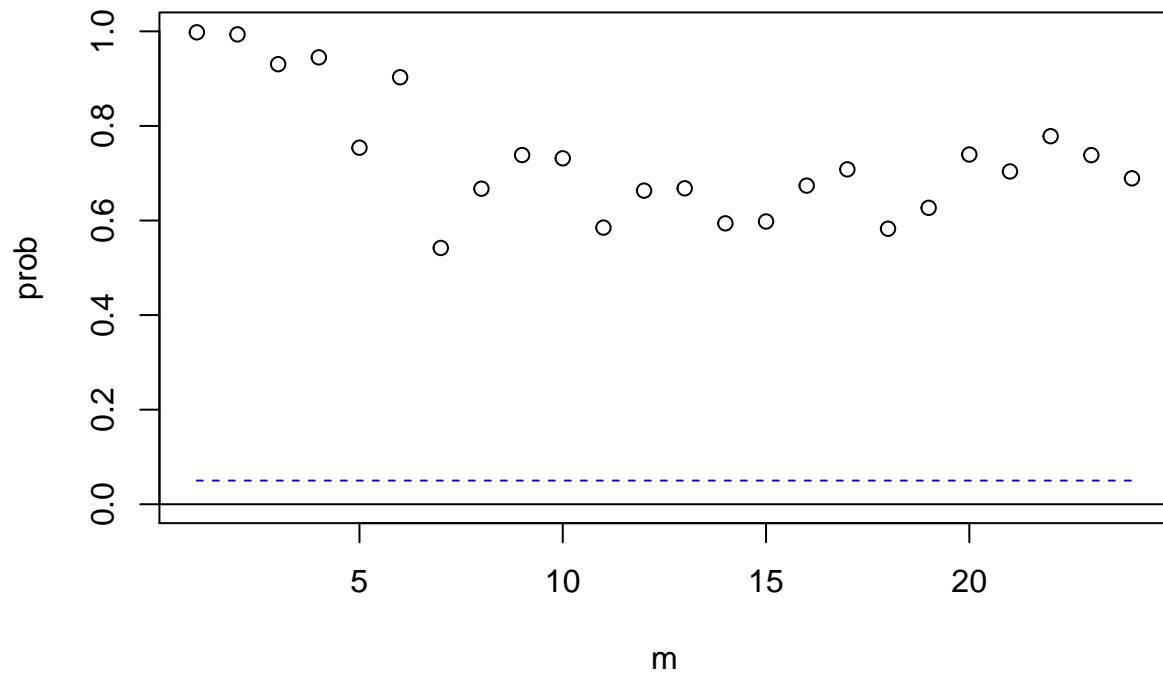
$$H_0: R_1 = \dots = R_m = 0$$

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

```
mq(m1$residuals)
```

```
## Ljung-Box Statistics:  
##           m      Q(m)    df    p-value  
## [1,]  1.000    0.131    4.000     1.00  
## [2,]  2.000    1.446    8.000     0.99  
## [3,]  3.000    5.696   12.000     0.93  
## [4,]  4.000    8.127   16.000     0.95  
## [5,]  5.000   15.384   20.000     0.75  
## [6,]  6.000   15.571   24.000     0.90  
## [7,]  7.000   26.566   28.000     0.54  
## [8,]  8.000   28.042   32.000     0.67  
## [9,]  9.000   30.244   36.000     0.74  
## [10,] 10.000   34.118   40.000     0.73  
## [11,] 11.000   41.370   44.000     0.58  
## [12,] 12.000   43.360   48.000     0.66  
## [13,] 13.000   47.061   52.000     0.67  
## [14,] 14.000   52.872   56.000     0.59  
## [15,] 15.000   56.676   60.000     0.60  
## [16,] 16.000   58.405   64.000     0.67  
## [17,] 17.000   61.180   68.000     0.71  
## [18,] 18.000   68.874   72.000     0.58  
## [19,] 19.000   71.436   76.000     0.63  
## [20,] 20.000   71.523   80.000     0.74  
## [21,] 21.000   76.618   84.000     0.70  
## [22,] 22.000   77.599   88.000     0.78  
## [23,] 23.000   82.988   92.000     0.74  
## [24,] 24.000   88.684   96.000     0.69
```

p-values of Ljung-Box statistics



Model Forecasting

Static Forecasting

```
pred = VARpred(m1, h = testLength)
```

```
## orig 300
## Forecasts at origin: 300
##          ts1      ts2
## [1,] -0.33489 -0.9148
## [2,] -0.16490 -0.7696
## [3,] -0.11047 -0.5470
## [4,] -0.08796 -0.3973
## [5,] -0.07699 -0.3097
## [6,] -0.07121 -0.2605
## [7,] -0.06809 -0.2332
## [8,] -0.06637 -0.2181
## [9,] -0.06543 -0.2098
## [10,] -0.06491 -0.2052
## [11,] -0.06463 -0.2027
## [12,] -0.06447 -0.2013
## [13,] -0.06439 -0.2006
## [14,] -0.06434 -0.2001
## [15,] -0.06431 -0.1999
## [16,] -0.06430 -0.1998
## [17,] -0.06429 -0.1997
## [18,] -0.06429 -0.1997
## [19,] -0.06428 -0.1996
## [20,] -0.06428 -0.1996
## Standard Errors of predictions:
##          [,1] [,2]
## [1,] 0.9899 1.082
## [2,] 1.0326 1.496
## [3,] 1.0390 1.661
## [4,] 1.0404 1.715
## [5,] 1.0408 1.732
## [6,] 1.0409 1.737
## [7,] 1.0409 1.739
## [8,] 1.0409 1.739
## [9,] 1.0409 1.739
## [10,] 1.0409 1.739
## [11,] 1.0409 1.739
## [12,] 1.0409 1.739
## [13,] 1.0409 1.739
## [14,] 1.0409 1.739
## [15,] 1.0409 1.739
## [16,] 1.0409 1.739
## [17,] 1.0409 1.739
## [18,] 1.0409 1.739
## [19,] 1.0409 1.739
## [20,] 1.0409 1.739
## Root mean square errors of predictions:
##          [,1] [,2]
## [1,] 0.9948 1.087
## [2,] 1.0343 1.511
```

```
## [3,] 1.0392 1.667
## [4,] 1.0405 1.717
## [5,] 1.0408 1.733
## [6,] 1.0409 1.737
## [7,] 1.0409 1.739
## [8,] 1.0409 1.739
## [9,] 1.0409 1.739
## [10,] 1.0409 1.739
## [11,] 1.0409 1.739
## [12,] 1.0409 1.739
## [13,] 1.0409 1.739
## [14,] 1.0409 1.739
## [15,] 1.0409 1.739
## [16,] 1.0409 1.739
## [17,] 1.0409 1.739
## [18,] 1.0409 1.739
## [19,] 1.0409 1.739
## [20,] 1.0409 1.739

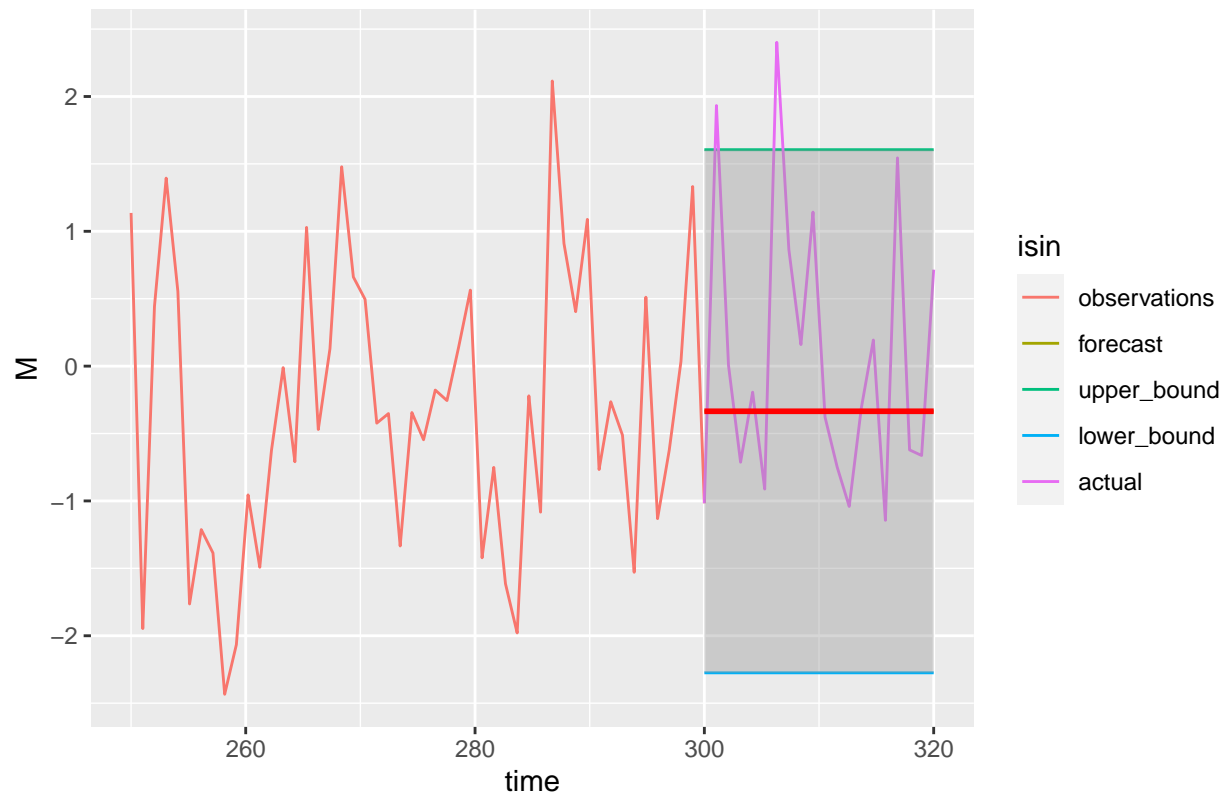
# Calculates the confidence interval
upperConf = pred$pred + 1.96 * pred$se.err
lowerConf = pred$pred - 1.96 * pred$se.err

drawLength = 50
drawStart = length - drawLength

## wrap data into a data.frame
df1 = data.frame(time = seq(drawStart,length,length=drawLength), M = ts$ts1[(drawStart+1):length], isin = "for")
df2 = data.frame(time = seq(length,length+testLength,length=testLength), M = pred$pred[1], isin = "forecast")
df3 = data.frame(time = seq(length,length+testLength,length=testLength), M = upperConf[1], isin = "upper")
df4 = data.frame(time = seq(length,length+testLength,length=testLength), M = lowerConf[1], isin = "lower")
df5 = data.frame(time = seq(length,length+testLength,length=testLength), M = test1, isin = "actual")
df = rbind(df1, df2, df3, df4, df5)

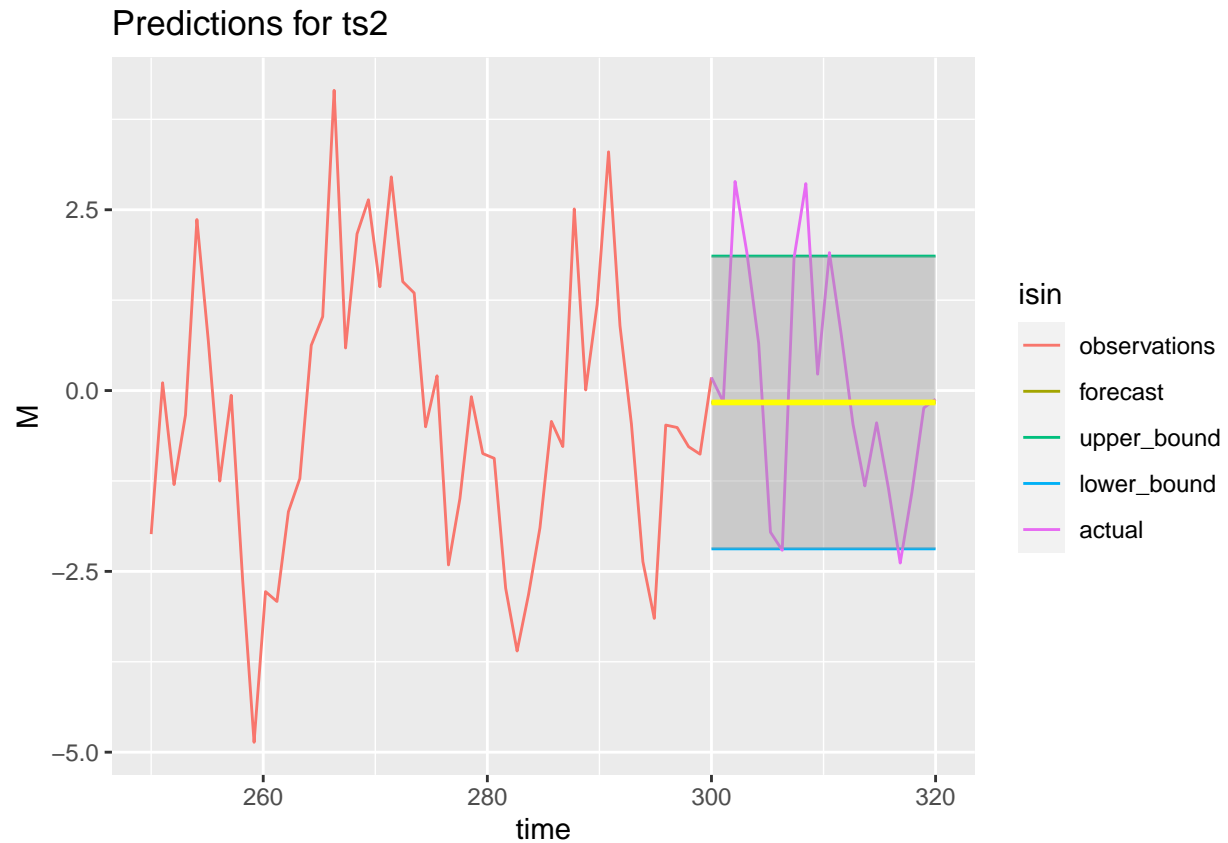
## ggplot object
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predictions for ts1") + geom_smooth()
```


Predictions for ts1



```
## wrap data into a data.frame
df1 = data.frame(time = seq(drawStart,length,length=drawLength), M = ts$ts2[(drawStart+1):length], isin = "observations")
df2 = data.frame(time = seq(length,length+testLength,length=testLength), M = pred$pred[2], isin = "forecast")
df3 = data.frame(time = seq(length,length+testLength,length=testLength), M = upperConf[2], isin = "upper_bound")
df4 = data.frame(time = seq(length,length+testLength,length=testLength), M = lowerConf[2], isin = "lower_bound")
df5 = data.frame(time = seq(length,length+testLength,length=testLength), M = test2, isin = "actual")
df = rbind(df1, df2, df3, df4, df5)

## ggplot object
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predictions for ts2") + geom_smooth()
```



Dynamic Forecasting

```
currData = ts
currPred1 = vector("numeric", testLength)
currPred2 = vector("numeric", testLength)
upperConf1 = vector("numeric", testLength)
upperConf2 = vector("numeric", testLength)
lowerConf1 = vector("numeric", testLength)
lowerConf2 = vector("numeric", testLength)
for (h in (length + 1):(length+testLength)) {
  # build the model
  m2 = VAR(currData, 1, output = F)

  i = h - length

  # predicts
  currPred = VARpred(m2, output = T)
  currPred1[i] = currPred$pred[1]
  currPred2[i] = currPred$pred[2]
  upperConf1[i] = currPred1[i] + 1.96*currPred$se.err[1]
  upperConf2[i] = currPred2[i] + 1.96*currPred$se.err[2]
  lowerConf1[i] = currPred1[i] - 1.96*currPred$se.err[1]
  lowerConf2[i] = currPred2[i] - 1.96*currPred$se.err[2]
}
```

```

oldData = as_tibble(currData)
currData = as_tibble(currData$ts1)
currData[h,1] = test1[i]
oldData = oldData[,-1]
oldData[h, 1] = test2[i]
currData[,2] = oldData$ts2
currData[,3] = 1:h
currData = rename(currData, "ts1" = "value")
currData = rename(currData, "ts2" = "...2")
currData = rename(currData, "index" = "...3")
currData = currData %>% as_tsibble(index = "index")
}

```

```

## orig 300
## Forecasts at origin: 300
##      ts1      ts2    index
## -0.5175 -0.9147 301.0000
## Standard Errors of predictions:
## [1] 9.839e-01 1.082e+00 2.470e-14
## Root mean square errors of predictions:
## [1] 9.904e-01 1.089e+00 2.486e-14
## orig 301
## Forecasts at origin: 301
##      ts1      ts2    index
## -0.5268 -0.8971 302.0000
## Standard Errors of predictions:
## [1] 9.827e-01 1.082e+00 2.282e-13
## Root mean square errors of predictions:
## [1] 9.892e-01 1.089e+00 2.297e-13
## orig 302
## Forecasts at origin: 302
##      ts1      ts2    index
##  0.3233  1.6897 303.0000
## Standard Errors of predictions:
## [1] 9.911e-01 1.081e+00 8.535e-14
## Root mean square errors of predictions:
## [1] 9.976e-01 1.088e+00 8.591e-14
## orig 303
## Forecasts at origin: 303
##      ts1      ts2    index
## -0.1456  1.2896 304.0000
## Standard Errors of predictions:
## [1] 9.896e-01 1.081e+00 1.211e-13
## Root mean square errors of predictions:
## [1] 9.961e-01 1.088e+00 1.219e-13
## orig 304
## Forecasts at origin: 304
##      ts1      ts2    index
## -0.3812  0.1961 305.0000
## Standard Errors of predictions:
## [1] 9.885e-01 1.080e+00 1.236e-13
## Root mean square errors of predictions:
## [1] 9.950e-01 1.087e+00 1.244e-13
## orig 305

```

```

## Forecasts at origin: 305
##      ts1      ts2      index
## -0.2635   0.1233 306.0000
## Standard Errors of predictions:
## [1] 9.869e-01 1.079e+00 1.221e-13
## Root mean square errors of predictions:
## [1] 9.934e-01 1.086e+00 1.229e-13
## orig 306
## Forecasts at origin: 306
##      ts1      ts2      index
## -0.5286  -1.7473 307.0000
## Standard Errors of predictions:
## [1] 9.860e-01 1.083e+00 8.223e-14
## Root mean square errors of predictions:
## [1] 9.924e-01 1.090e+00 8.276e-14
## orig 307
## Forecasts at origin: 307
##      ts1      ts2      index
##  0.4123   1.2213 308.0000
## Standard Errors of predictions:
## [1] 9.983e-01 1.082e+00 1.169e-13
## Root mean square errors of predictions:
## [1] 1.005e+00 1.089e+00 1.177e-13
## orig 308
## Forecasts at origin: 308
##      ts1      ts2      index
##  0.07193   1.61455 309.00000
## Standard Errors of predictions:
## [1] 9.970e-01 1.081e+00 6.426e-14
## Root mean square errors of predictions:
## [1] 1.003e+00 1.088e+00 6.467e-14
## orig 309
## Forecasts at origin: 309
##      ts1      ts2      index
## -0.1048   1.4408 310.0000
## Standard Errors of predictions:
## [1] 9.954e-01 1.081e+00 2.683e-14
## Root mean square errors of predictions:
## [1] 1.002e+00 1.088e+00 2.701e-14
## orig 310
## Forecasts at origin: 310
##      ts1      ts2      index
##  0.1417   1.1603 311.0000
## Standard Errors of predictions:
## [1] 9.962e-01 1.082e+00 1.184e-13
## Root mean square errors of predictions:
## [1] 1.003e+00 1.089e+00 1.191e-13
## orig 311
## Forecasts at origin: 311
##      ts1      ts2      index
## -0.2511   0.4883 312.0000
## Standard Errors of predictions:
## [1] 9.951e-01 1.081e+00 1.881e-13
## Root mean square errors of predictions:

```

```

## [1] 1.001e+00 1.088e+00 1.893e-13
## orig 312
## Forecasts at origin: 312
##      ts1      ts2      index
## -0.3849 -0.3588 313.0000
## Standard Errors of predictions:
## [1] 9.938e-01 1.079e+00 1.250e-13
## Root mean square errors of predictions:
## [1] 1.000e+00 1.086e+00 1.258e-13
## orig 313
## Forecasts at origin: 313
##      ts1      ts2      index
## -0.4999 -1.1923 314.0000
## Standard Errors of predictions:
## [1] 9.929e-01 1.077e+00 3.228e-14
## Root mean square errors of predictions:
## [1] 9.993e-01 1.084e+00 3.249e-14
## orig 314
## Forecasts at origin: 314
##      ts1      ts2      index
## -0.3137 -0.8877 315.0000
## Standard Errors of predictions:
## [1] 9.914e-01 1.076e+00 2.336e-13
## Root mean square errors of predictions:
## [1] 9.977e-01 1.082e+00 2.351e-13
## orig 315
## Forecasts at origin: 315
##      ts1      ts2      index
## -0.15242 -0.01348 316.00000
## Standard Errors of predictions:
## [1] 9.902e-01 1.074e+00 1.802e-13
## Root mean square errors of predictions:
## [1] 9.965e-01 1.081e+00 1.814e-13
## orig 316
## Forecasts at origin: 316
##      ts1      ts2      index
## -0.5496 -1.6941 317.0000
## Standard Errors of predictions:
## [1] 9.902e-01 1.075e+00 1.663e-13
## Root mean square errors of predictions:
## [1] 9.965e-01 1.082e+00 1.673e-13
## orig 317
## Forecasts at origin: 317
##      ts1      ts2      index
##  0.1876  0.3694 318.0000
## Standard Errors of predictions:
## [1] 9.955e-01 1.074e+00 8.738e-14
## Root mean square errors of predictions:
## [1] 1.002e+00 1.081e+00 8.793e-14
## orig 318
## Forecasts at origin: 318
##      ts1      ts2      index
## -0.3904 -1.2570 319.0000
## Standard Errors of predictions:

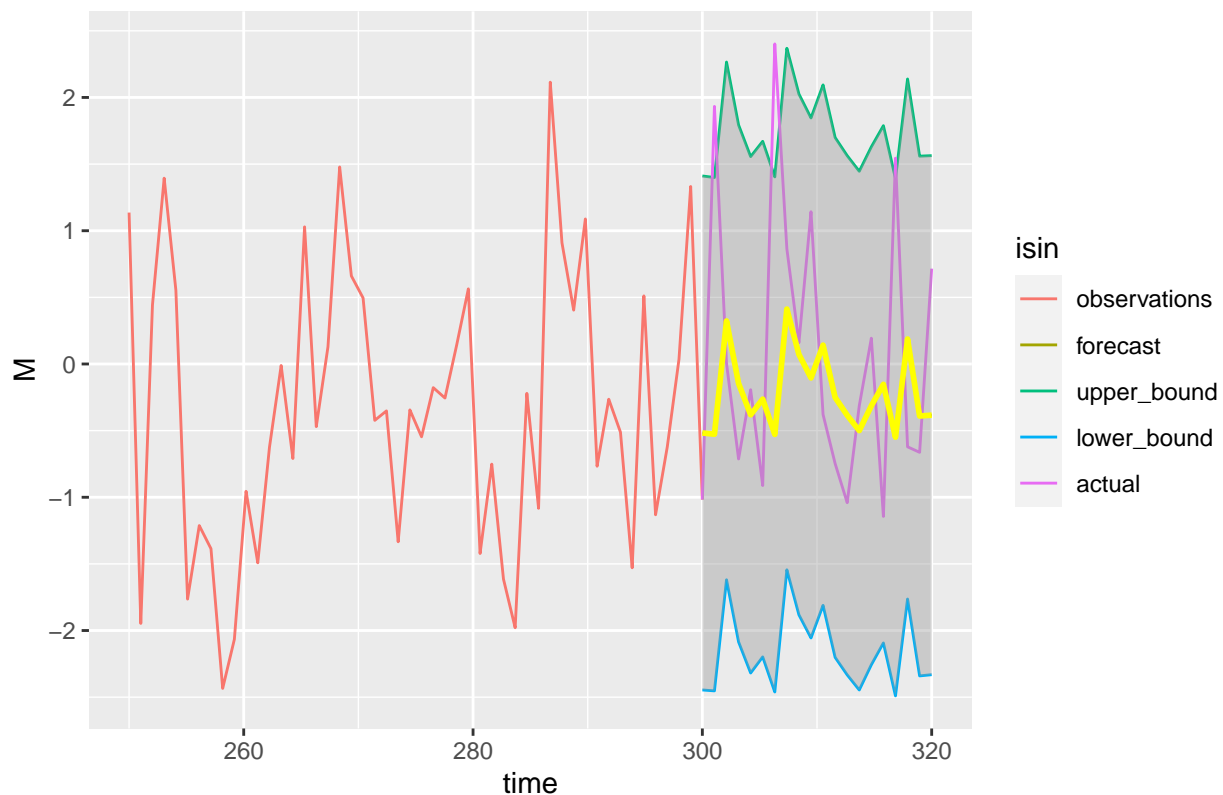
```

```
## [1] 9.950e-01 1.077e+00 3.745e-14
## Root mean square errors of predictions:
## [1] 1.001e+00 1.084e+00 3.769e-14
## orig 319
## Forecasts at origin: 319
##      ts1      ts2    index
## -0.3839 -0.7588 320.0000
## Standard Errors of predictions:
## [1] 9.935e-01 1.077e+00 1.849e-13
## Root mean square errors of predictions:
## [1] 9.997e-01 1.083e+00 1.860e-13

## wrap data into a data.frame
df1 = data.frame(time = seq(drawStart,length,length=drawLength), M = ts$ts1[(drawStart+1):length], isin = "observations")
df2 = data.frame(time = seq(length,length+testLength,length=testLength), M = currPred1, isin = "forecast")
df3 = data.frame(time = seq(length,length+testLength,length=testLength), M = upperConf1, isin = "upper_bound")
df4 = data.frame(time = seq(length,length+testLength,length=testLength), M = lowerConf1, isin = "lower_bound")
df5 = data.frame(time = seq(length,length+testLength,length=testLength), M = test1, isin = "actual")
df = rbind(df1, df2, df3, df4, df5)

## ggplot object
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predictions for ts1") + geom_smooth()
```

Predictions for ts1



```
## wrap data into a data.frame
df1 = data.frame(time = seq(drawStart,length,length=drawLength), M = ts$ts1[(drawStart+1):length], isin = "observations")
df2 = data.frame(time = seq(length,length+testLength,length=testLength), M = currPred2, isin = "forecast")
df3 = data.frame(time = seq(length,length+testLength,length=testLength), M = upperConf2, isin = "upper_bound")
```

```
df4 = data.frame(time = seq(length,length+testLength,length=testLength), M = lowerConf2, isin = "lower_")
df5 = data.frame(time = seq(length,length+testLength,length=testLength), M = test2, isin = "actual")
df = rbind(df1, df2, df3, df4, df5)
```

```
## ggplot object
```

```
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predictions for ts2") + geom_smooth()
```

