VARComplexExample

Jackson Cates

9/28/2020

Libraries

```
library(dplyr)
library(tsibble)
library(ggplot2)
library(feasts)
library(gridExtra)
library(MTS)
library(tseries)
library(forecast)
library(seastests)
```

Data Generation

$z_{1,t} = z_{2,t-1} + \epsilon_{1,t}$

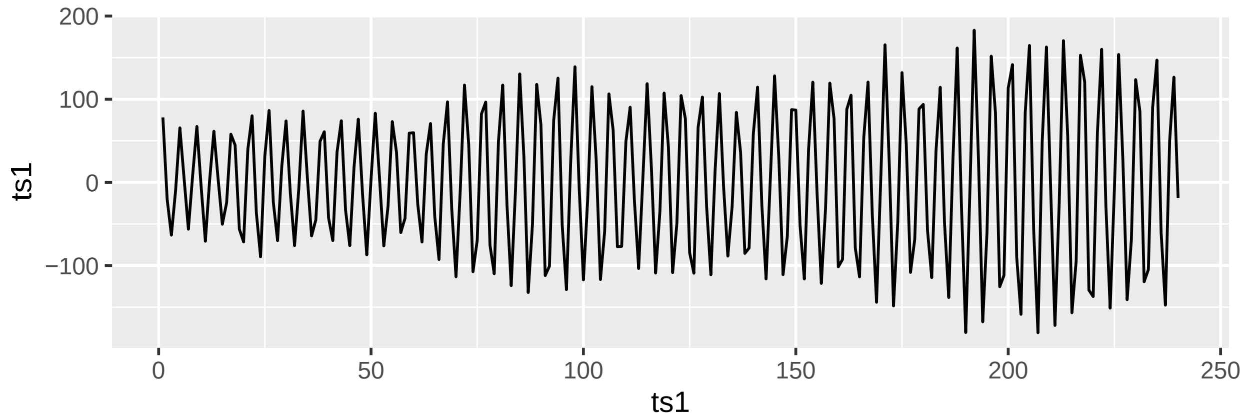
$z_{2,t} = -1.01 z_{1,t-1} + 0.2 z_{2,t-1} + \epsilon_{2,t}$

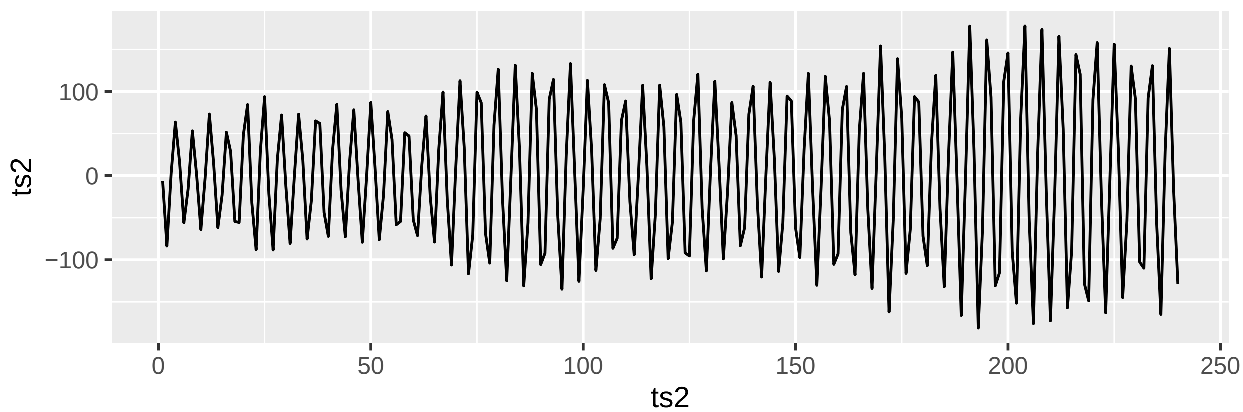
$z_t = \begin{pmatrix} 0 & 1 \\ -1.01 & 0.2 \end{pmatrix} z_{t-1} + \epsilon_t$

```
set.seed(6)
skip = 20
length = 240
testLength = 180
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] = ts2[t-1] + noise1[t] + noise1[t]
  ts2[t] = -1.01*ts1[t-1] + 0.2*ts2[t-1] + noise2[t]
```

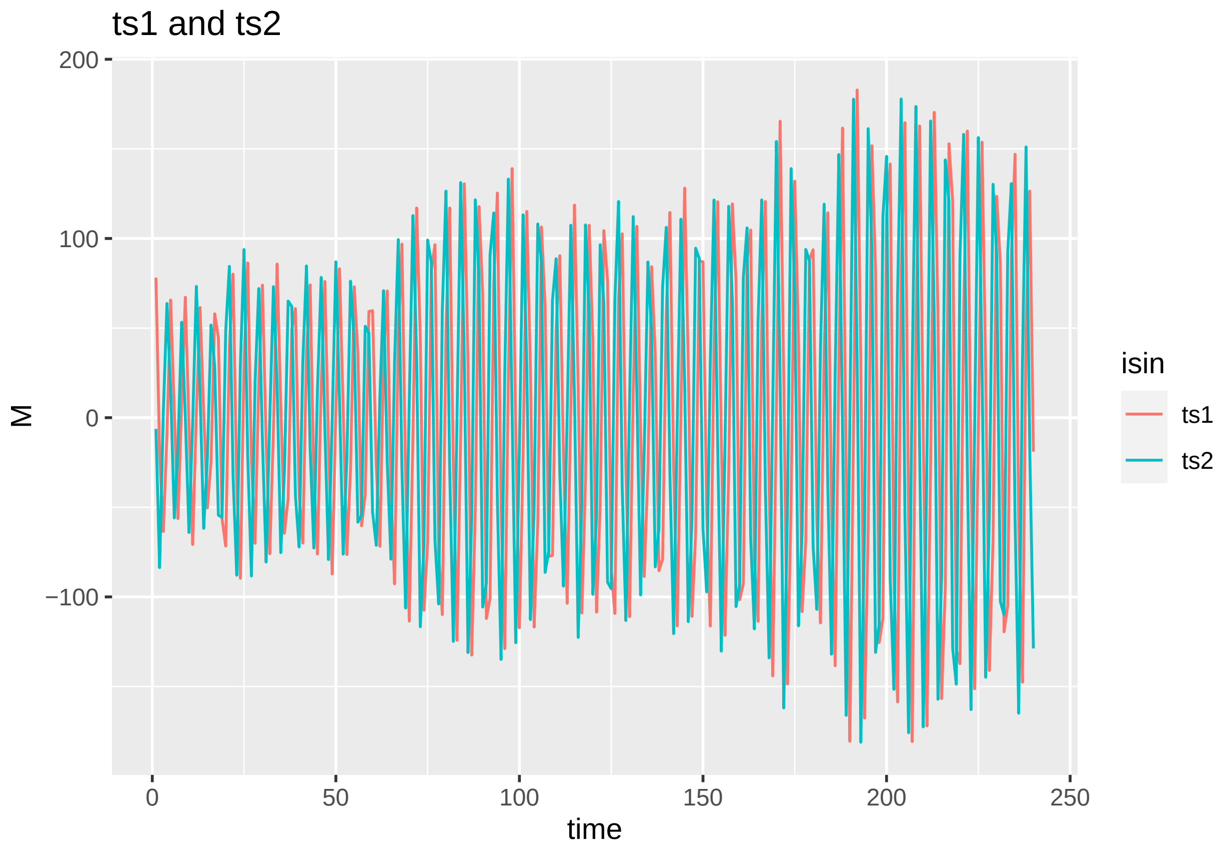
```
}

# 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")
```

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



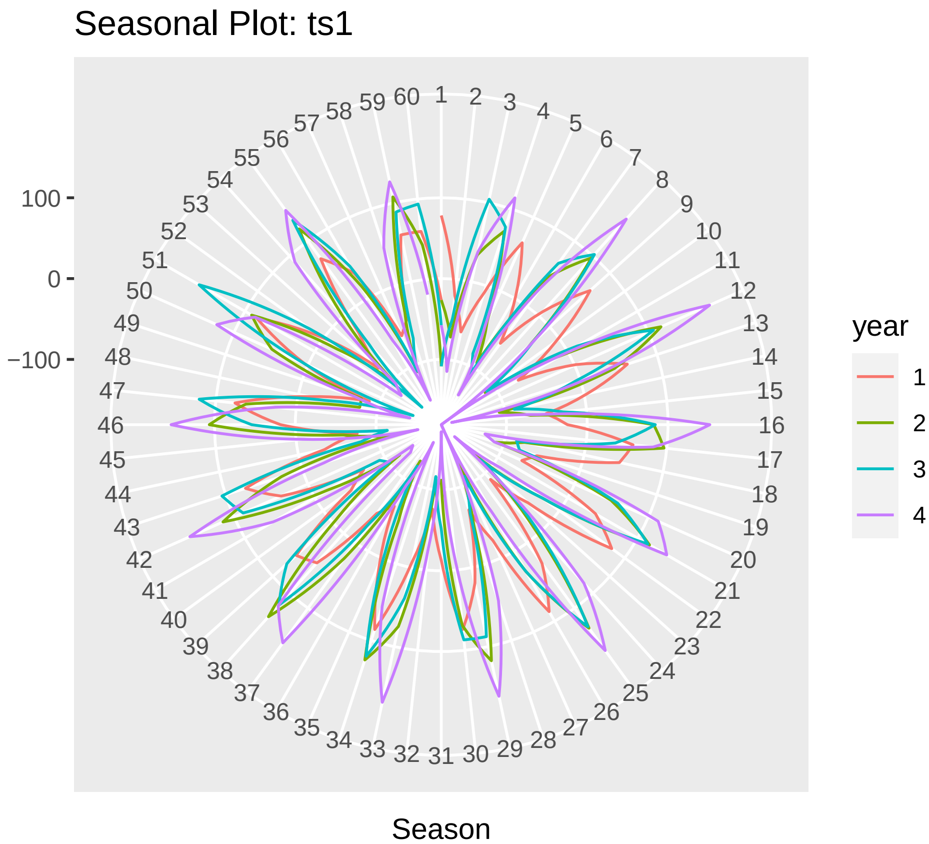



```
df1 = data.frame(time = ts$index, M = ts$ts1, isin = "ts1")
df2 = data.frame(time = ts$index, M = ts$ts2, isin = "ts2")
df = rbind(df1, df2)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("ts1 and ts2")
```

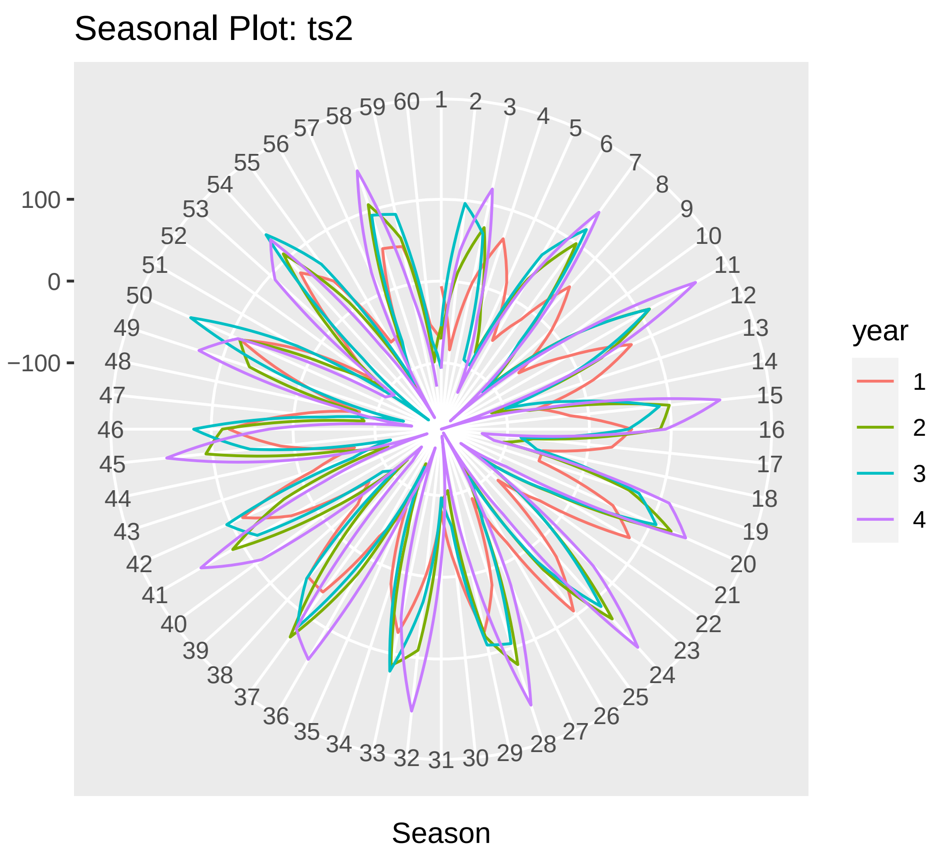


Difference

```
tsSeasonalPlot = ts %>% as.ts(start = c(1969, 2), frequency = 60)
ggseasonplot(tsSeasonalPlot[,1], polar = T ) + ggtitle("Seasonal Plot: ts1")
```



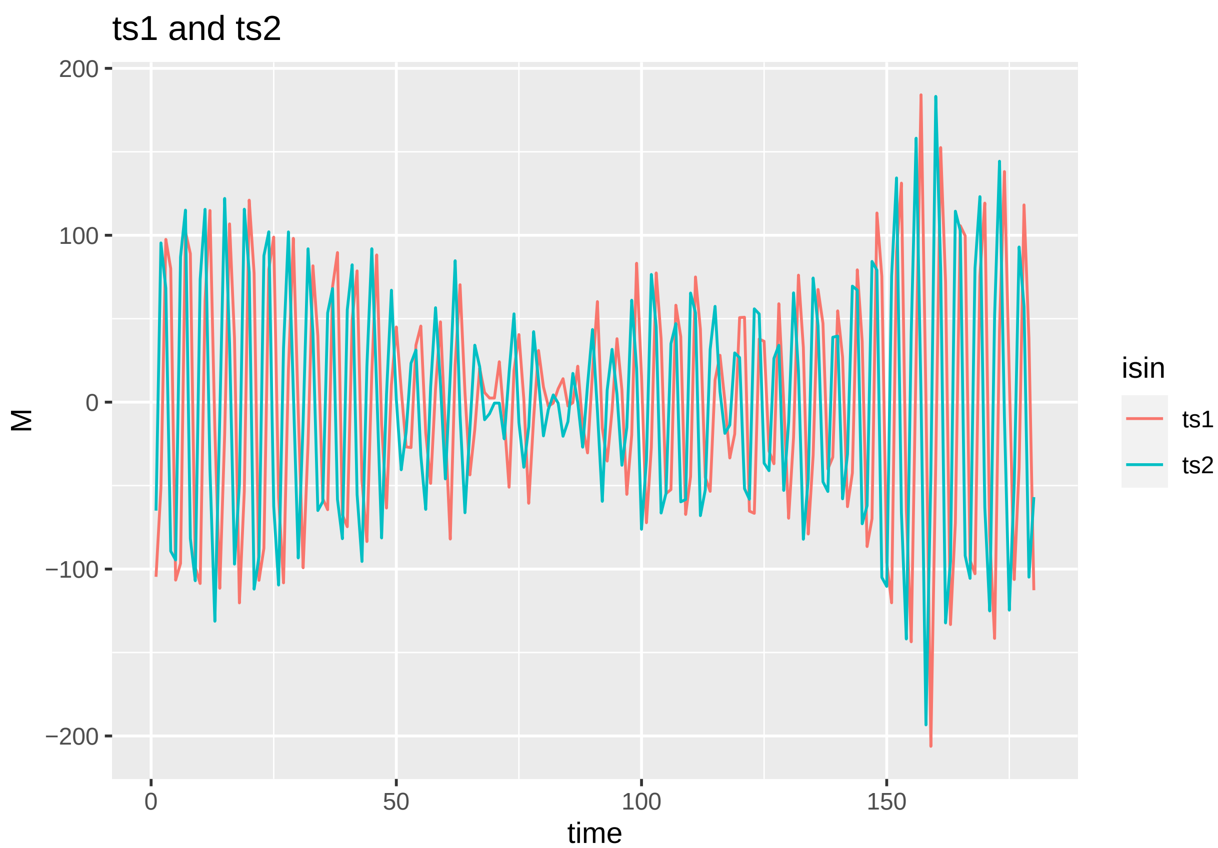

```
ggseasonplot(tsSeasonalPlot[,2], polar = T ) + ggtitle("Seasonal Plot: ts2")
```



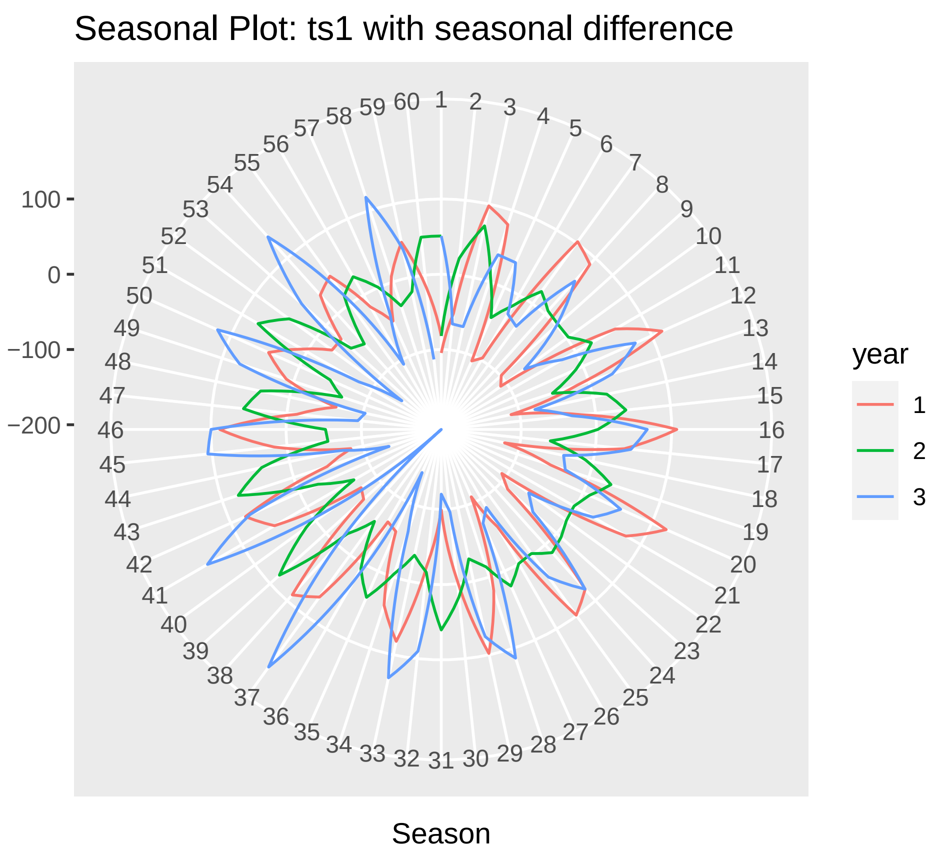
Taking the seasonal difference

$y'_t = y_t - y_{t-60}$

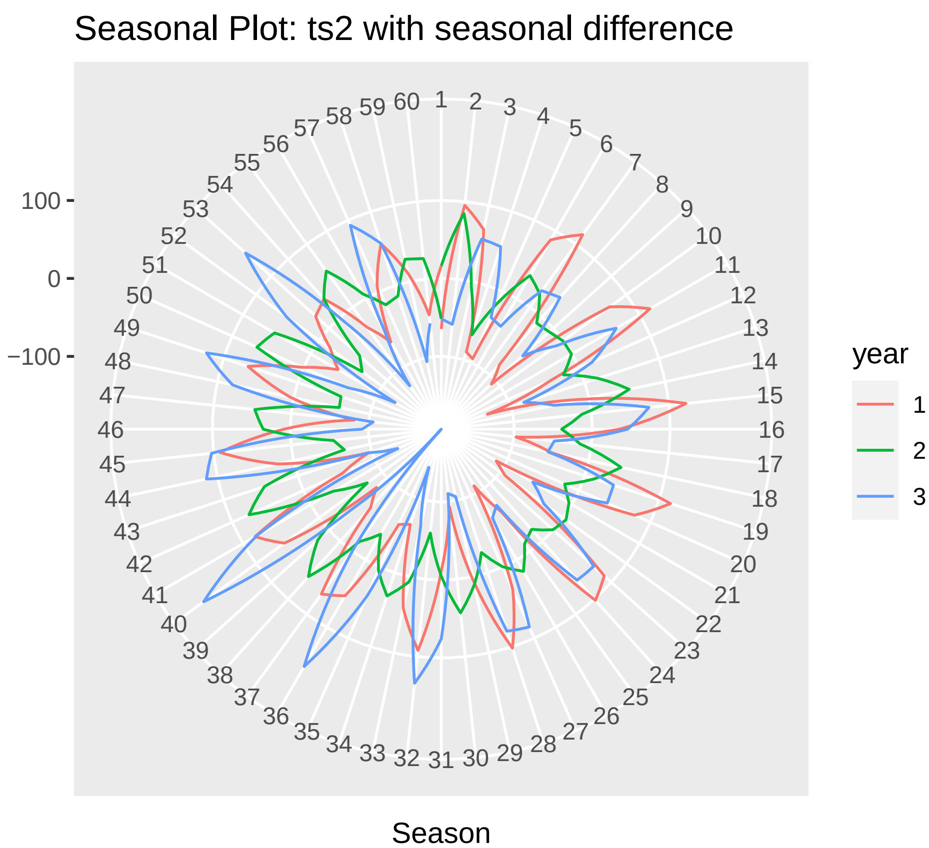
```
tsSeasonal = ts %>% diffM(d = 60)
tsSeasonal = tsSeasonal %>% as_tibble()
tsSeasonal[,3] = 1:(length - 60)
tsSeasonal = tsSeasonal %>% as_tsibble(index = "index")

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



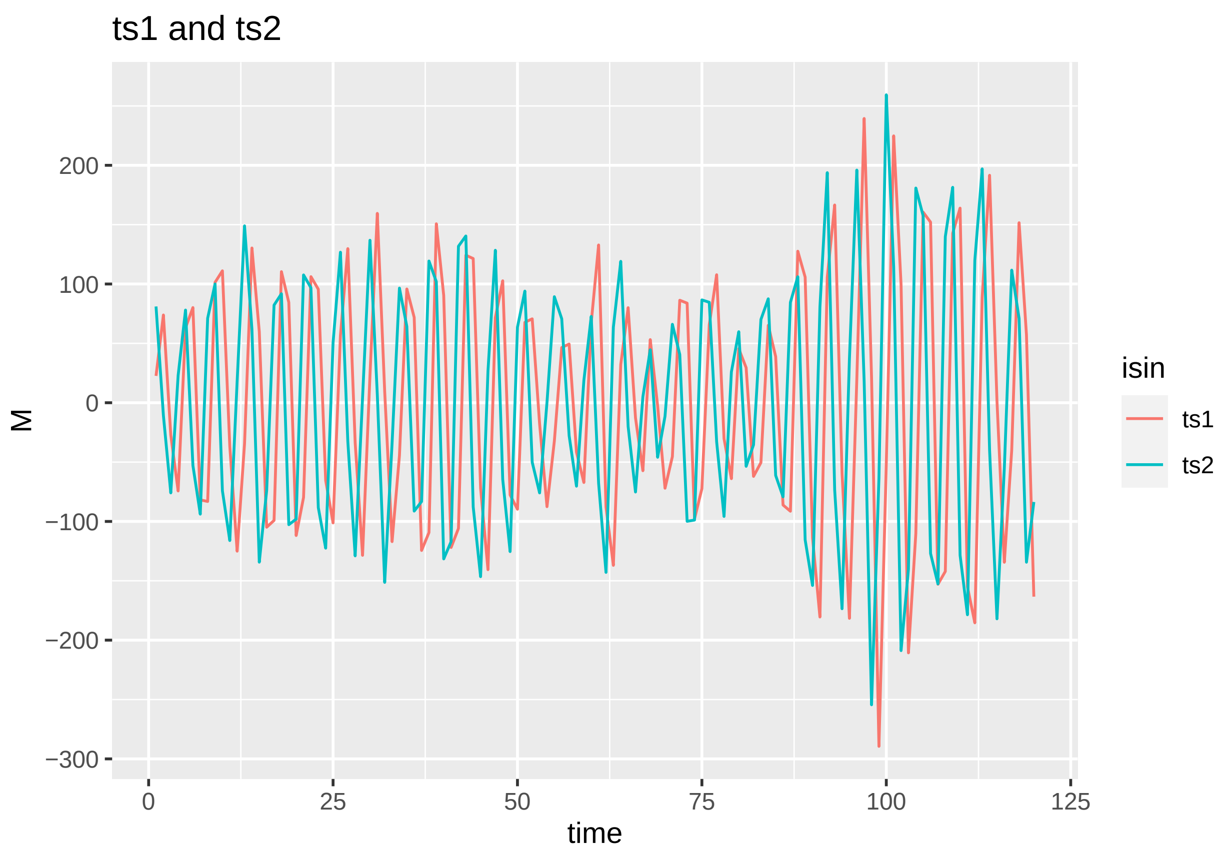

```r
tsSeasonalPlot = tsSeasonal %>% as.ts(start = c(1969, 2), frequency = 60)
ggseasonplot(tsSeasonalPlot[,1], polar = T ) + ggtitle("Seasonal Plot: ts1 with seasonal difference")
```



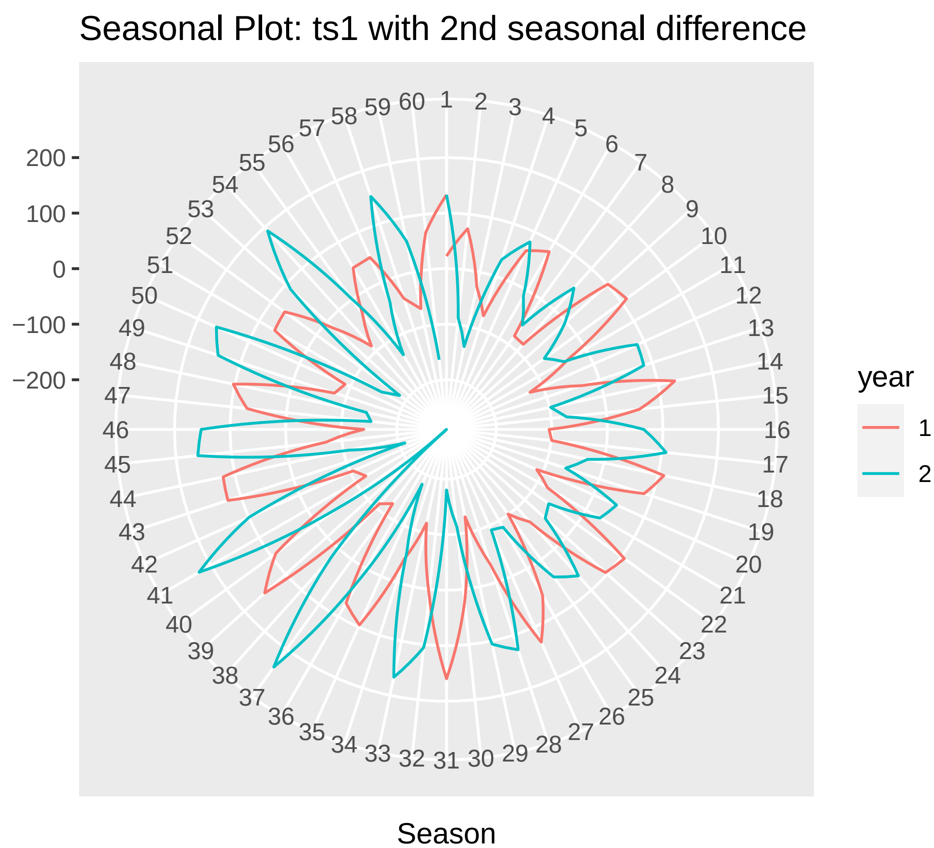

```
ggseasonplot(tsSeasonalPlot[,2], polar = T ) + ggtitle("Seasonal Plot: ts2 with seasonal difference")
```



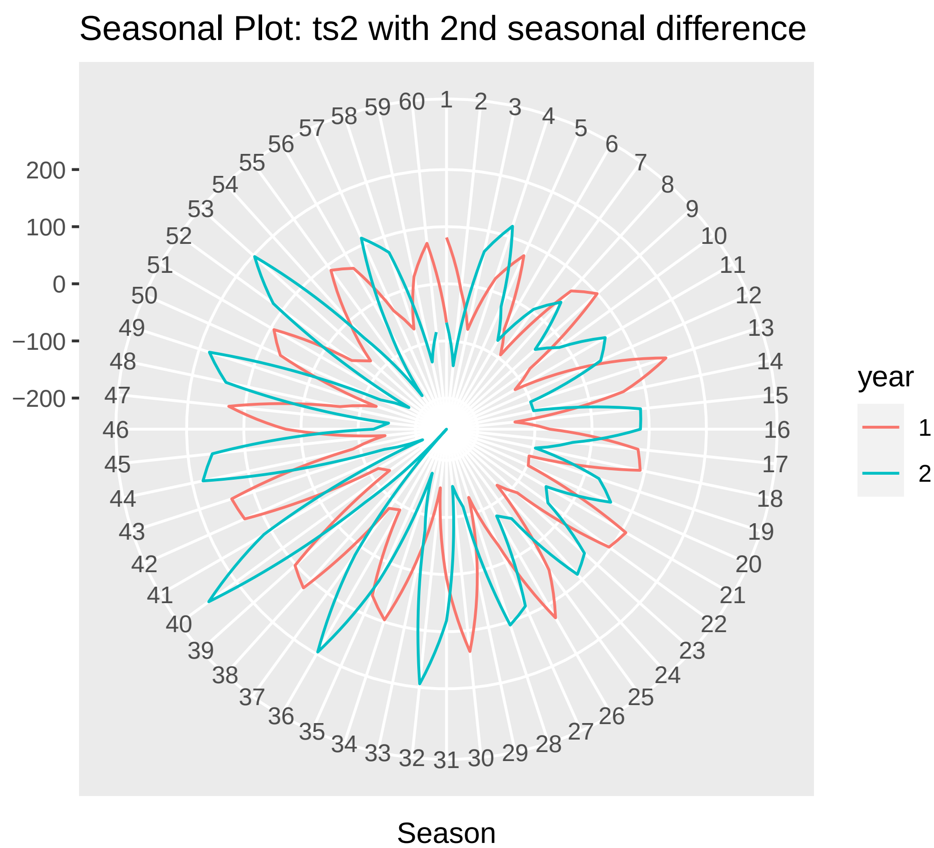

```
tsSeasonal = tsSeasonal %>% diffM(d = 60)
tsSeasonal = tsSeasonal %>% as_tibble()
tsSeasonal[,3] = 1:(length - 120)
tsSeasonal = tsSeasonal %>% as_tsibble(index = "index")

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




```
tsSeasonalPlot = tsSeasonal %>% as.ts(start = c(1969, 2), frequency = 60)
ggseasonplot(tsSeasonalPlot[,1], polar = T ) + ggtitle("Seasonal Plot: ts1 with 2nd seasonal difference
```




```
ggseasonplot(tsSeasonalPlot[,2], polar = T ) + ggtitle("Seasonal Plot: ts2 with 2nd seasonal difference
```



Webel-Ollech Test

```
kpss.test(tsSeasonal$ts1)
```

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

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsSeasonal$ts1
## KPSS Level = 0.065899, Truncation lag parameter = 4, p-value = 0.1
```

```
kpss.test(tsSeasonal$ts2)
```

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

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsSeasonal$ts2
## KPSS Level = 0.042954, Truncation lag parameter = 4, p-value = 0.1
```

Fitting the model

```
VARorder(tsSeasonal[,-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 18.7640 18.7640 18.7640   0.0000  0.0000
##  [2,]  1 11.5006 11.5935 11.5383 758.6653  0.0000
##  [3,]  2 11.5185 11.7044 11.5940   4.9453  0.2930
##  [4,]  3 11.5101 11.7888 11.6233   7.4755  0.1128
##  [5,]  4 11.5036 11.8752 11.6545   7.1352  0.1289
##  [6,]  5 11.5354 12.0000 11.7241   3.3273  0.5046
##  [7,]  6 11.5474 12.1049 11.7738   5.1099  0.2762
##  [8,]  7 11.5567 12.2071 11.8209   5.2482  0.2628
##  [9,]  8 11.6112 12.3545 11.9130   1.0927  0.8954
## [10,]  9 11.5820 12.4182 11.9216   8.3875  0.0784
## [11,] 10 11.6005 12.5296 11.9778   4.1206  0.3899
## [12,] 11 11.6294 12.6515 12.0445   3.1514  0.5328
## [13,] 12 11.6264 12.7414 12.0792   5.6769  0.2246
## [14,] 13 11.5554 12.7633 12.0459  10.9446  0.0272
```

$z_t = \begin{pmatrix} 0 & 0.90 \\ -0.95 & 0.22 \end{pmatrix} z_{t-1} + a_t$

```r
m1 = VAR(tsSeasonal[,-3], p = 1)
```

```
## Constant term:
## Estimates:  0.4089481 -0.2947539
## Std.Error:  2.282573 1.127269
## AR coefficient matrix
## AR( 1 )-matrix
##        [,1]  [,2]
## [1,] -0.023 0.977
## [2,] -1.002 0.177
## standard error
##        [,1]   [,2]
## [1,] 0.0217 0.0215
## [2,] 0.0107 0.0106
##
## Residuals cov-mtx:
##           [,1]      [,2]
## [1,] 604.23864 -17.95039
## [2,] -17.95039 147.37184
##
## det(SSE) =  88725.54
## AIC =  11.45997
## BIC =  11.55289
## HQ  =  11.4977
```

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

```
## Constant term:
## Estimates:  0 0
## Std.Error:  0 0
## AR coefficient matrix
## AR( 1 )-matrix
##        [,1]  [,2]
## [1,] -0.023 0.977
```

```
## [2,] -1.002 0.177
## standard error
##        [,1]   [,2]
## [1,] 0.0216 0.0214
## [2,] 0.0107 0.0106
##
## Residuals cov-mtx:
##          [,1]     [,2]
## [1,] 604.4058 -18.0709
## [2,] -18.0709 147.4587
##
## det(SSE) =  88798.34
## AIC =  11.46079
## BIC =  11.55371
## HQ  =  11.49852
```

Model Checking

Multivariate Portmanteau Statistics

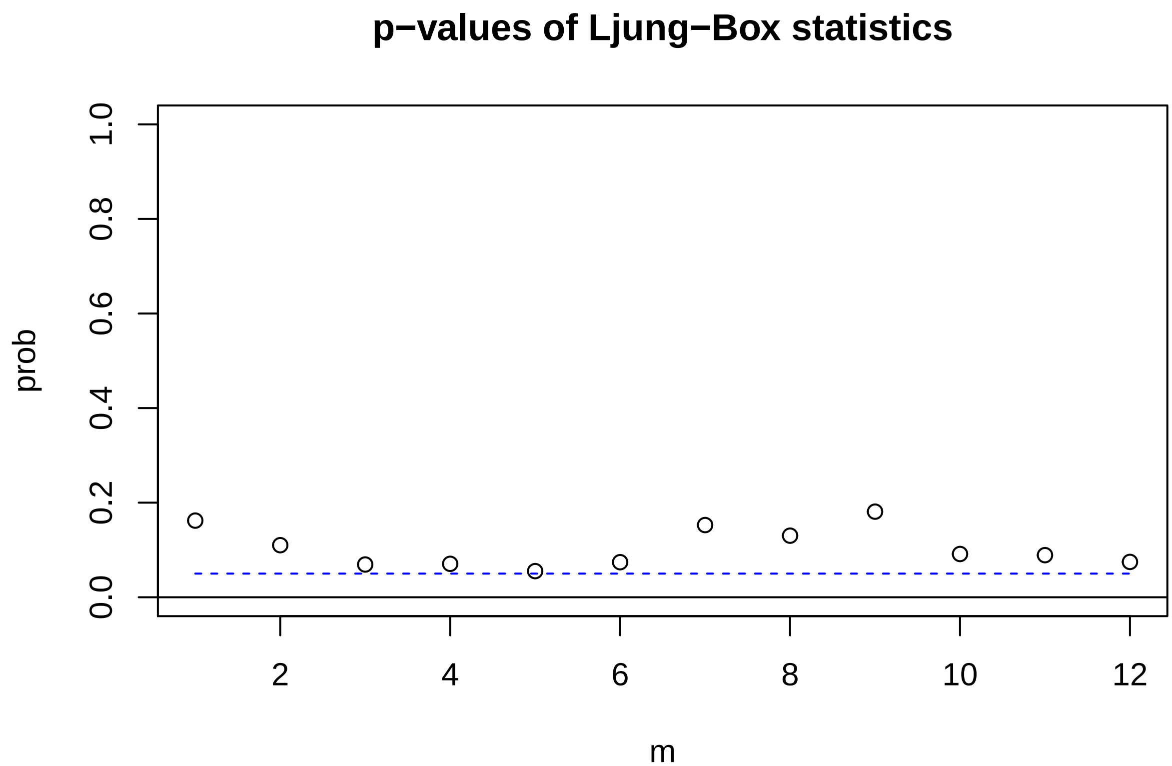
Let R_ℓ be the theoretical lag ℓ cross-correlation matrix of innovation a_t

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

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

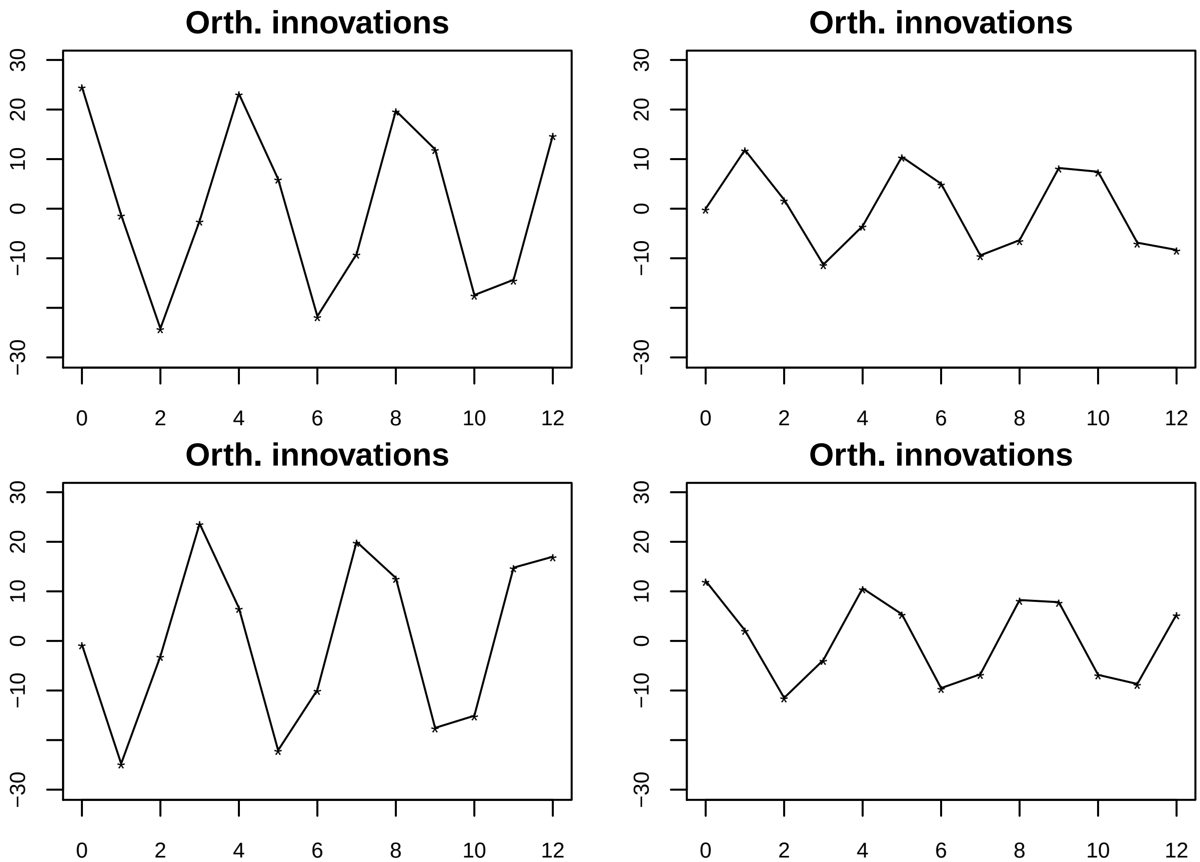
```r
mq(m1R$residuals, lag = 12)
```

```
## Ljung-Box Statistics:
##          m       Q(m)     df    p-value
##  [1,]   1.00      6.54    4.00     0.16
##  [2,]   2.00     13.05    8.00     0.11
##  [3,]   3.00     19.89   12.00     0.07
##  [4,]   4.00     24.95   16.00     0.07
##  [5,]   5.00     30.99   20.00     0.06
##  [6,]   6.00     34.63   24.00     0.07
##  [7,]   7.00     35.62   28.00     0.15
##  [8,]   8.00     41.09   32.00     0.13
##  [9,]   9.00     43.55   36.00     0.18
## [10,]  10.00     52.34   40.00     0.09
## [11,]  11.00     57.09   44.00     0.09
## [12,]  12.00     62.76   48.00     0.07
```

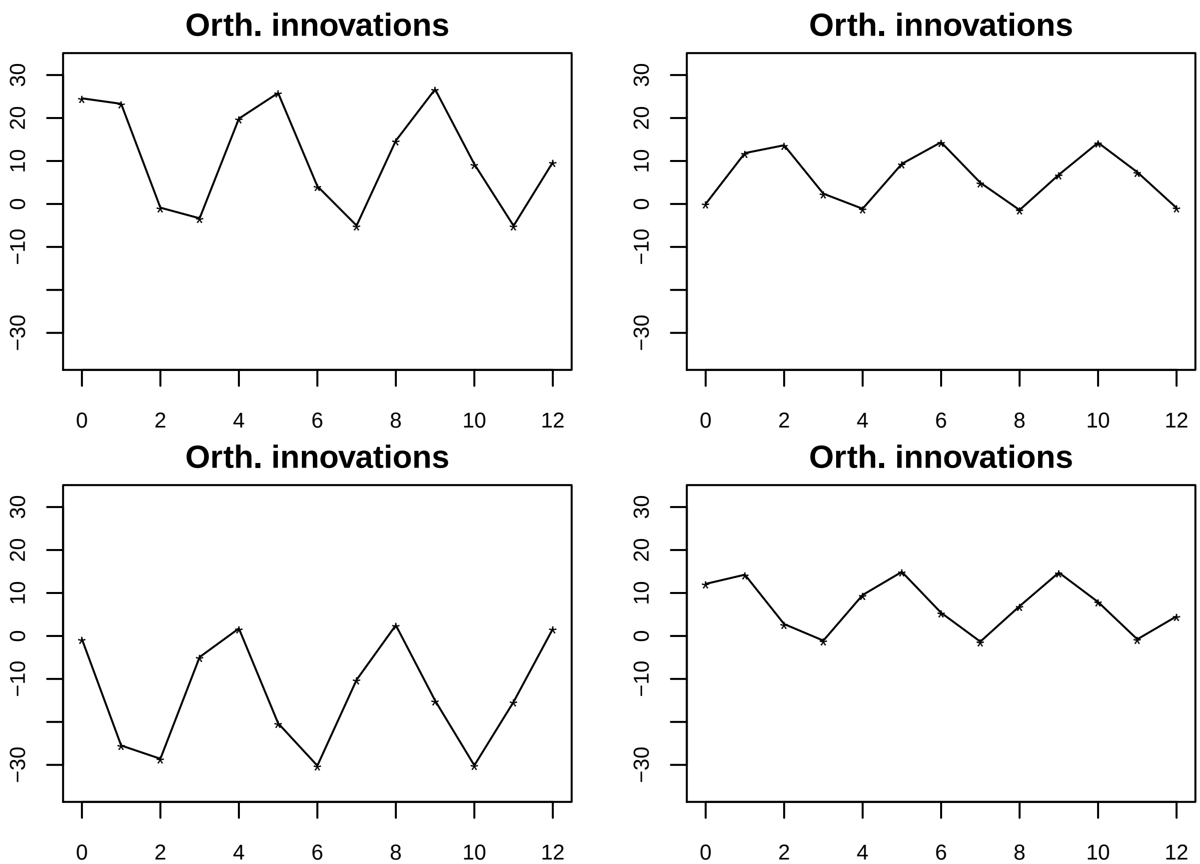


Impulse

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




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



Model Forecasting

```
tsPredSeasonal = VARpred(m1R, h = testLength - 120)
```

```
## orig  120
## Forecasts at origin:  120
##             ts1       ts2
##  [1,]   -77.968  148.8950
##  [2,]   147.324  104.4173
##  [3,]    98.679 -129.1486
##  [4,]  -128.499 -121.6778
##  [5,]  -115.982  107.2392
##  [6,]   107.481  135.1420
##  [7,]   129.625  -83.8039
##  [8,]   -84.888 -144.6697
##  [9,]  -139.456   59.4857
## [10,]    61.344  150.2227
## [11,]   145.424  -34.9176
## [12,]   -37.468 -151.8614
## [13,]  -147.574   10.7073
## [14,]    13.853  149.7379
## [15,]   146.040   12.5756
## [16,]     8.939 -144.0882
## [17,]  -141.041  -34.4121
## [18,]   -30.398  135.2221
## [19,]   132.868   54.3438
```

```
## [20,]   50.067 -123.5122
## [21,] -121.874  -71.9807
## [22,]  -67.558  109.3819
## [23,]  108.464   87.0079
## [24,]   82.554  -93.2924
## [25,]  -93.082  -99.1888
## [26,]  -94.813   75.7299
## [27,]   76.197  108.3677
## [28,]  104.173  -57.1923
## [29,]  -58.293 -114.4693
## [30,] -110.548   38.1770
## [31,]   39.853  117.4964
## [32,]  113.930  -19.1683
## [33,]  -21.351 -117.5265
## [34,] -114.384    0.6268
## [35,]    3.238  114.7057
## [36,]  112.043   17.0209
## [37,]   14.065 -109.2422
## [38,] -107.100  -33.3907
## [39,]  -30.179  101.3980
## [40,]   99.802   48.1484
## [41,]   44.771  -91.4800
## [42,]  -90.443  -61.0153
## [43,]  -57.562   79.8304
## [44,]   79.350   71.7720
## [45,]   68.331  -66.8164
## [46,]  -66.877  -80.2613
## [47,]  -76.915   52.8205
## [48,]   53.394   86.3885
## [49,]   83.213  -38.2301
## [50,]  -39.277  -90.1208
## [51,]  -87.185   23.4282
## [52,]   24.901   91.4852
## [53,]   88.849   -8.7839
## [54,]  -10.625  -90.5647
## [55,]  -88.277   -5.3554
## [56,]   -3.208   87.4936
## [57,]   85.592   18.6716
## [58,]   16.285  -82.4517
## [59,]  -80.965  -30.8823
## [60,]  -28.327   75.6581
## Standard Errors of predictions:
##         [,1]   [,2]
##  [1,]  24.58  12.14
##  [2,]  27.32  27.66
##  [3,]  36.52  30.11
##  [4,]  38.30  38.49
##  [5,]  44.92  40.48
##  [6,]  46.50  46.42
##  [7,]  51.56  48.40
##  [8,]  53.21  52.80
##  [9,]  57.11  54.93
## [10,]  58.93  58.19
## [11,]  61.90  60.50
```

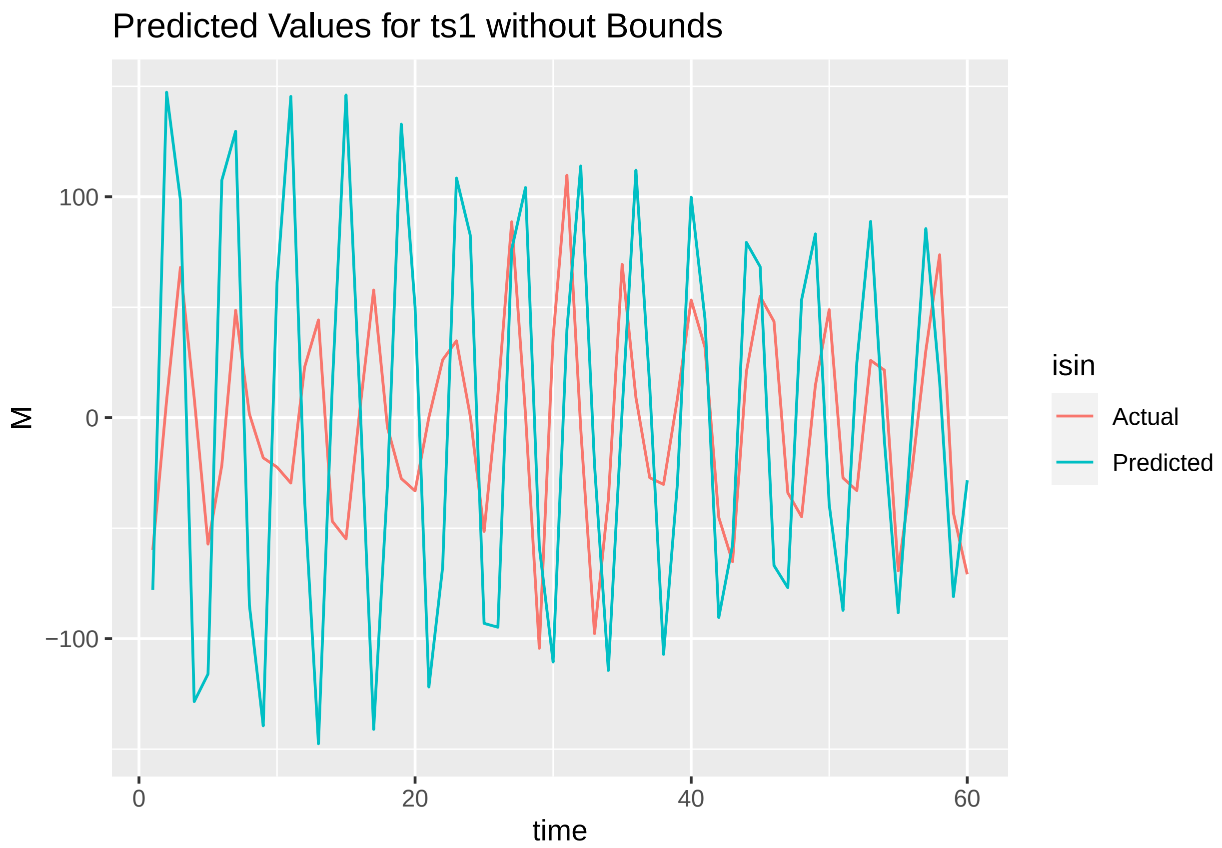
```
## [12,]  63.91  62.89
## [13,]  66.12  65.36
## [14,]  68.31  67.06
## [15,]  69.92  69.64
## [16,]  72.23  70.84
## [17,]  73.37  73.44
## [18,]  75.73  74.29
## [19,]  76.54  76.84
## [20,]  78.88  77.48
## [21,]  79.48  79.90
## [22,]  81.72  80.43
## [23,]  82.22  82.66
## [24,]  84.30  83.19
## [25,]  84.78  85.16
## [26,]  86.64  85.76
## [27,]  87.18  87.45
## [28,]  88.79  88.15
## [29,]  89.42  89.55
## [30,]  90.76  90.38
## [31,]  91.52  91.50
## [32,]  92.60  92.46
## [33,]  93.47  93.32
## [34,]  94.30  94.39
## [35,]  95.29  95.02
## [36,]  95.91  96.17
## [37,]  96.97  96.63
## [38,]  97.42  97.82
## [39,]  98.53  98.16
## [40,]  98.86  99.34
## [41,]  99.97  99.61
## [42,] 100.23 100.74
## [43,] 101.30 100.99
## [44,] 101.53 102.03
## [45,] 102.53 102.31
## [46,] 102.78 103.24
## [47,] 103.67 103.56
## [48,] 103.96 104.35
## [49,] 104.73 104.74
## [50,] 105.08 105.40
## [51,] 105.72 105.86
## [52,] 106.14 106.38
## [53,] 106.65 106.91
## [54,] 107.14 107.31
## [55,] 107.53 107.89
## [56,] 108.07 108.19
## [57,] 108.37 108.82
## [58,] 108.95 109.03
## [59,] 109.17 109.67
## [60,] 109.77 109.84
## Root mean square errors of predictions:
##         [,1]   [,2]
##  [1,]  24.89  12.29
##  [2,] 173.16 357.69
##  [3,] 349.74 173.38
```

```
##  [4,] 169.96 346.28
##  [5,] 339.57 184.23
##  [6,] 178.89 329.16
##  [7,] 323.76 202.72
##  [8,] 195.88 307.25
##  [9,] 303.17 224.22
## [10,] 216.41 281.73
## [11,] 278.91 245.31
## [12,] 236.98 253.99
## [13,] 252.32 263.73
## [14,] 255.26 225.79
## [15,] 225.03 278.13
## [16,] 269.81 199.25
## [17,] 199.06 287.71
## [18,] 279.80 176.93
## [19,] 176.83 292.13
## [20,] 284.83 161.58
## [21,] 160.95 291.33
## [22,] 284.80 155.25
## [23,] 153.54 285.54
## [24,] 279.89 158.09
## [25,] 155.04 275.19
## [26,] 270.50 168.05
## [27,] 163.75 260.90
## [28,] 257.21 182.10
## [29,] 176.83 243.52
## [30,] 240.80 197.40
## [31,] 191.51 224.05
## [32,] 222.23 211.86
## [33,] 205.65 203.72
## [34,] 202.68 224.07
## [35,] 217.81 184.01
## [36,] 183.52 233.14
## [37,] 227.05 166.61
## [38,] 166.37 238.59
## [39,] 232.84 153.28
## [40,] 152.92 240.22
## [41,] 234.96 145.54
## [42,] 144.66 238.07
## [43,] 233.42 144.00
## [44,] 142.32 232.38
## [45,] 228.42 148.03
## [46,] 145.46 223.58
## [47,] 220.34 156.03
## [48,] 152.67 212.22
## [49,] 209.71 166.11
## [50,] 162.13 199.03
## [51,] 197.22 176.56
## [52,] 172.19 184.88
## [53,] 183.68 186.11
## [54,] 181.55 170.78
## [55,] 170.06 193.86
## [56,] 189.31 157.86
## [57,] 157.44 199.27
```

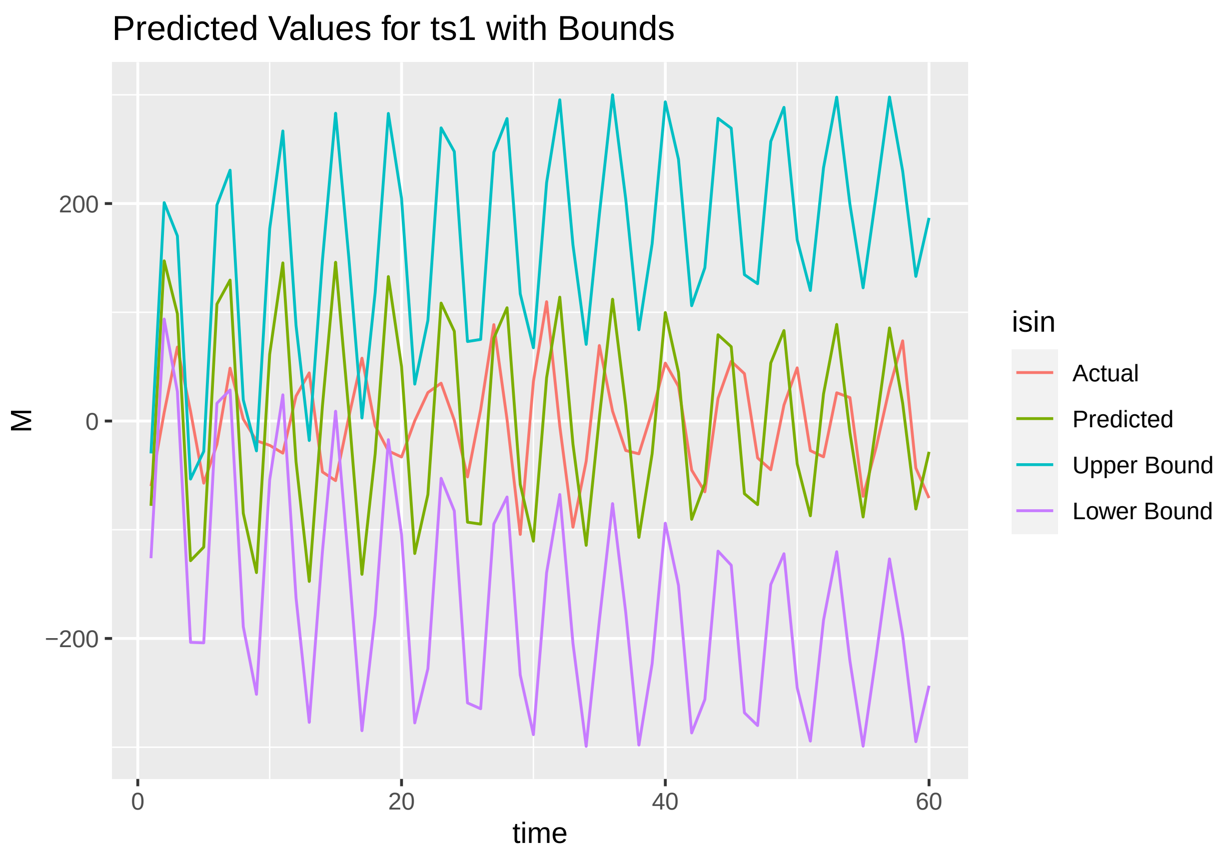
```
## [58,] 194.86 147.28
## [59,] 146.92 202.02
## [60,] 197.90 140.05
```

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

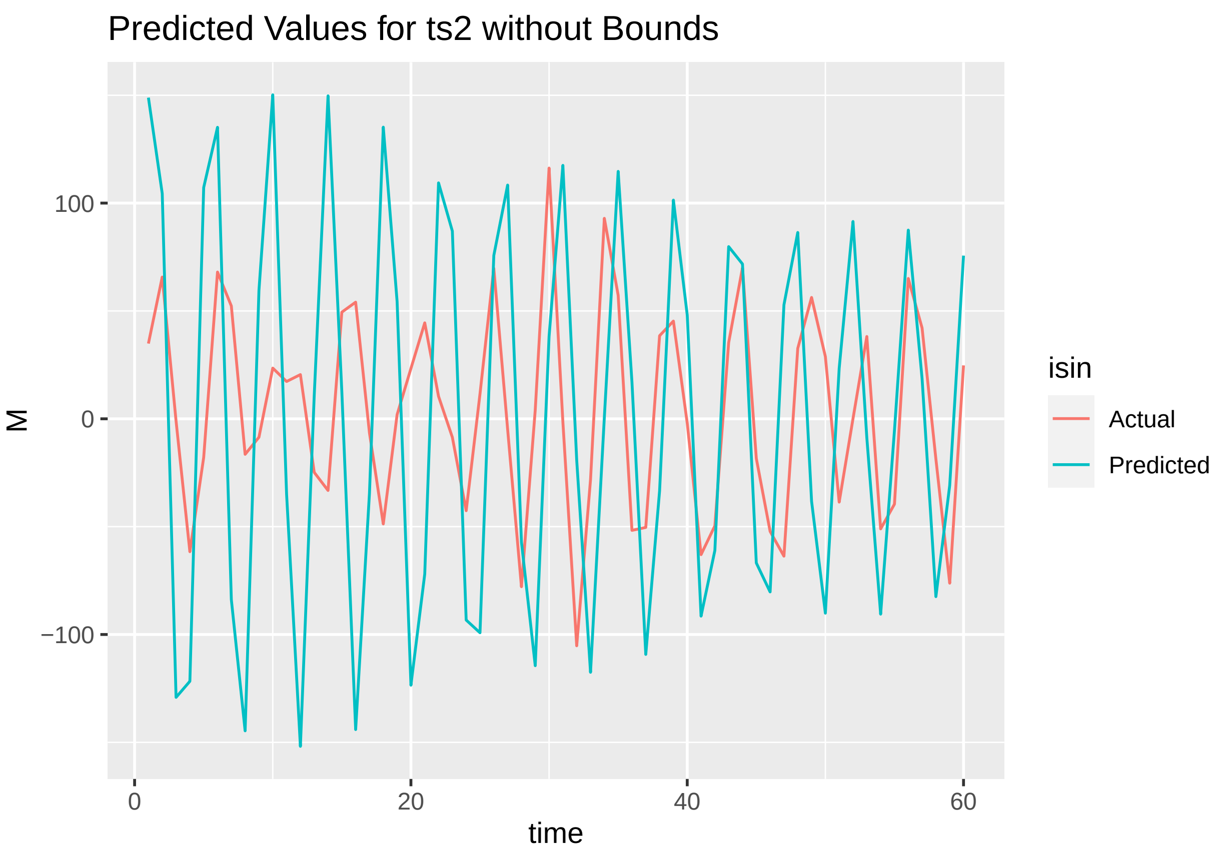
```
tsTestSeasonal = tsTest %>% diffM(d = 60) %>% diffM(d = 60)
tsTestSeasonal = tsTestSeasonal %>% as_tibble()
tsTestSeasonal[,3] = 1:(testLength - 120)
tsTestSeasonal = tsTestSeasonal %>% as_tsibble(index = "index")

df1 = data.frame(time = tsTestSeasonal$index, M = tsTestSeasonal$ts1, isin = "Actual")
df2 = data.frame(time = tsTestSeasonal$index, M = tsPredSeasonal$pred[,1], isin = "Predicted")
df3 = data.frame(time = tsTestSeasonal$index, M = upperConf[,1], isin = "Upper Bound")
df4 = data.frame(time = tsTestSeasonal$index, M = lowerConf[,1], isin = "Lower Bound")
df = rbind(df1, df2)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicted Values for ts1 withou
```



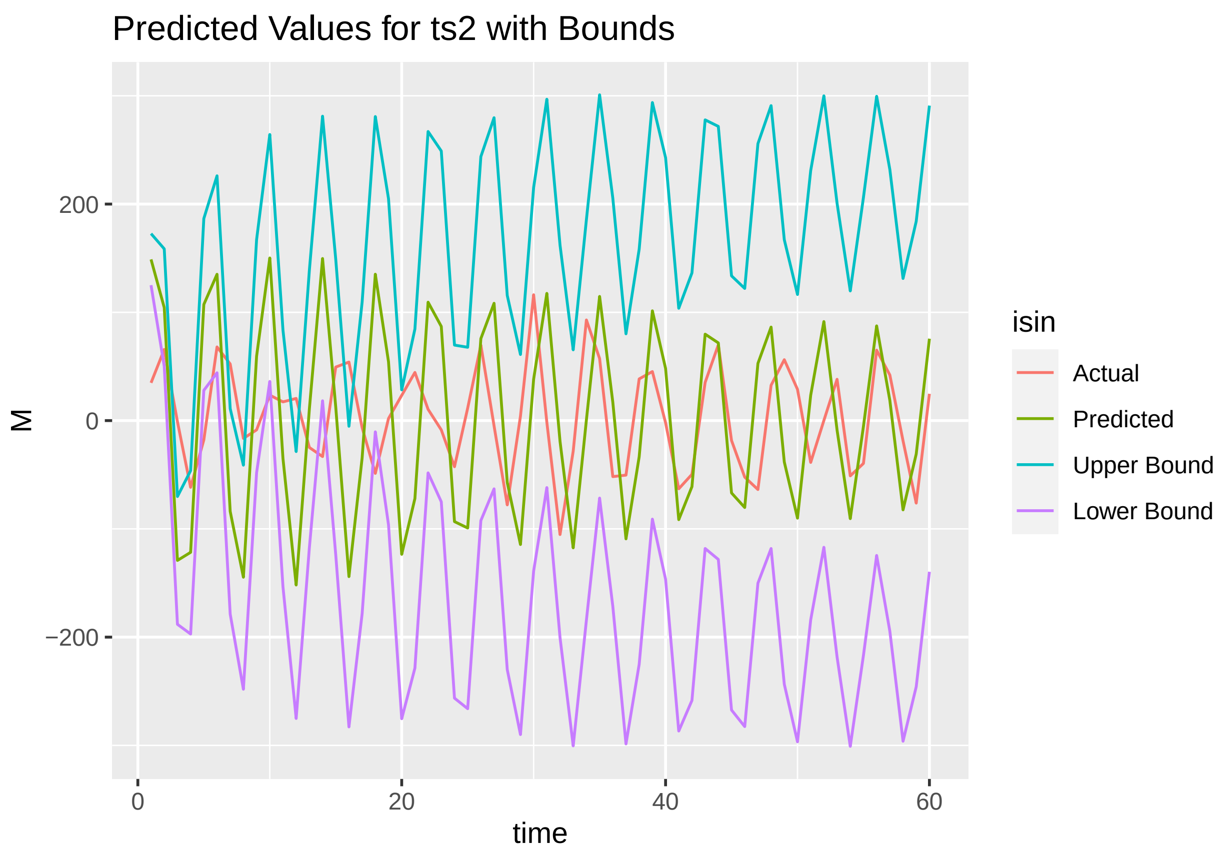

```
df = rbind(df1, df2, df3, df4)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicted Values for ts1 with Bo
```




```
df1 = data.frame(time = tsTestSeasonal$index, M = tsTestSeasonal$ts2, isin = "Actual")
df2 = data.frame(time = tsTestSeasonal$index, M = tsPredSeasonal$pred[,2], isin = "Predicted")
df3 = data.frame(time = tsTestSeasonal$index, M = upperConf[,2], isin = "Upper Bound")
df4 = data.frame(time = tsTestSeasonal$index, M = lowerConf[,2], isin = "Lower Bound")
df = rbind(df1, df2)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicted Values for ts2 withou
```




```
df = rbind(df1, df2, df3, df4)
ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicted Values for ts2 with Bo
```



Inverting the Difference

$y_t = \sum_{i=0}^{t/60} y'_{t-60i} + y_0$

```
# # Inverts
# tsPred = vector("numeric", testLength)
# for (t in 1:testLength) {
#
#   # Gets the current value
#   differenceSum = 0
#   for (i in 1:((t / 60) + 1)) {
#
#     differenceSum = differenceSum + tsPredSeasonal$pred[1 + 60*(i-1),1]
#
#   }
#   differenceSum = differenceSum + ts$ts1[length]
#
#   print(differenceSum)
#
# }
```

```
# 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 = "
# df3 = data.frame(time = seq(length, length + testLength, length=testLength), M = pred$pred[,1], isin =
# 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, df3)
# ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Prediction for ts1  Bounds an
# df = rbind(df1, df2, df3)
# ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("without for ts1 Prediction Bo
# df = rbind(df1, df2, df3, df4, df5)
# ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("without for ts1 with Bounds")
#
# 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 = "
# df3 = data.frame(time = seq(length, length + testLength, length=testLength), M = pred$pred[,2], isin =
# 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, df3)
# ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Predicted for ts1 without Bou
# df = rbind(df1, df2, df3)
# ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Prediction for ts1 without Bo
# df = rbind(df1, df2, df3, df4, df5)
# ggplot(df, aes(x = time, y = M, color = isin)) + geom_line() + ggtitle("Prediction for ts1 with Bound
```

MAE

```
# mean(abs(tsTest$ts1 - pred$pred[,1]))
# mean(abs(tsTest$ts2 - pred$pred[,2]))
```

MSE

```
# mean((tsTest$ts1 - pred$pred[,1])*(tsTest$ts1 - pred$pred[,1]))
# mean((tsTest$ts2 - pred$pred[,2])*(tsTest$ts2 - pred$pred[,2]))
```