Team 3 Final Project

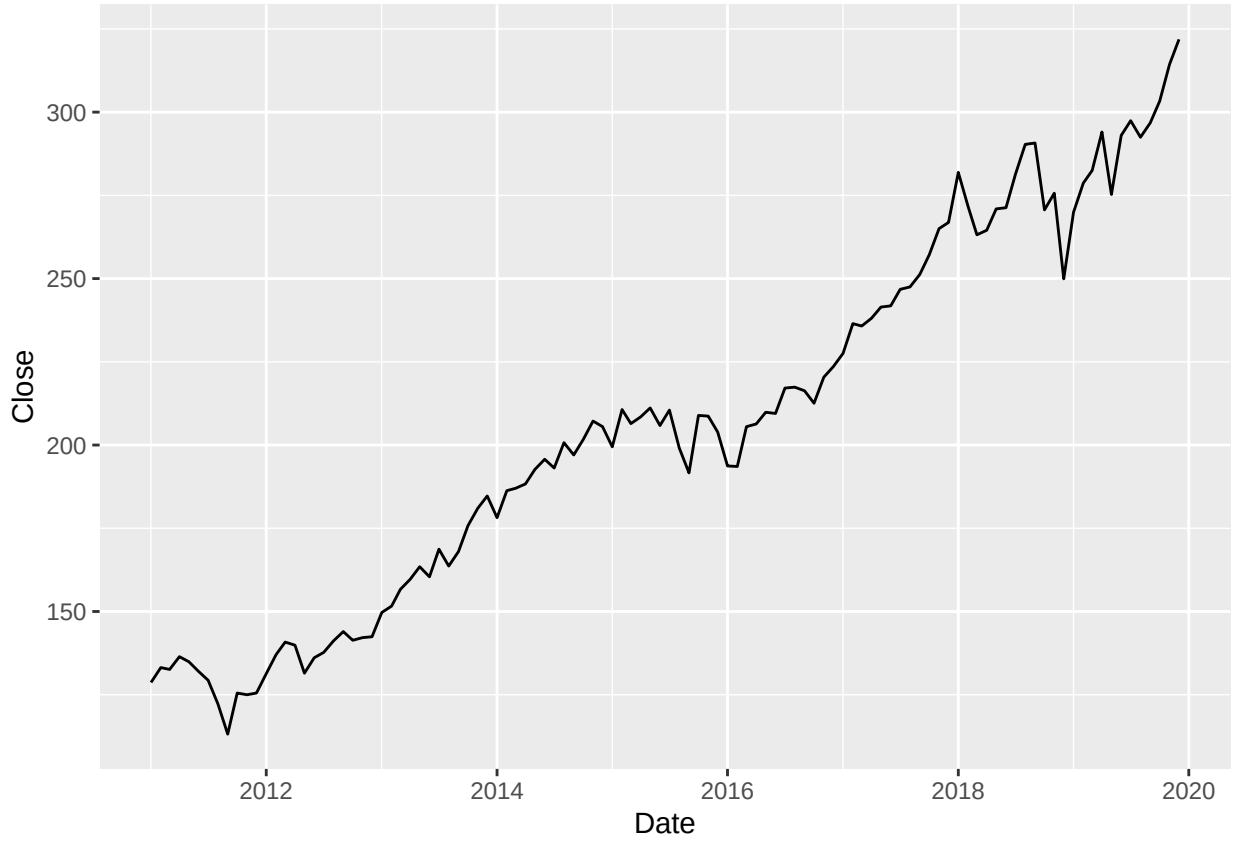
Group 3: Gus Horkan, Roman Paolucci, Jorge Jimenez

10/08/2020

Contents

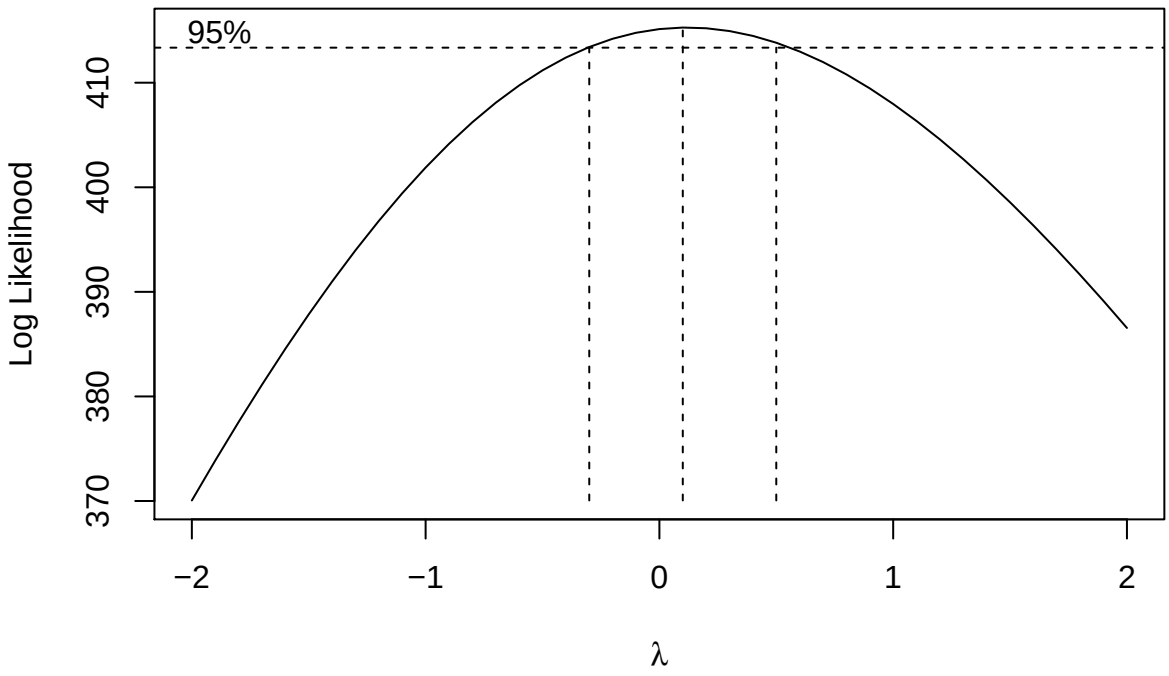
```
SandP <- SandP %>%
  mutate(Date = as.Date(Date)) %>%
  select(Date, Close)
SandP2 <- SandP %>%
  filter(Date > "2020-01-01")
SandP <- SandP %>%
  filter(Date < "2020-01-01")

SandPTS <- xts(SandP$Close, order.by = as.POSIXct(SandP$Date))
```

```
ggplot(SandP, aes(x = Date, y = Close)) + geom_line()
```



The plot looks like it may have some nonconstant variance.

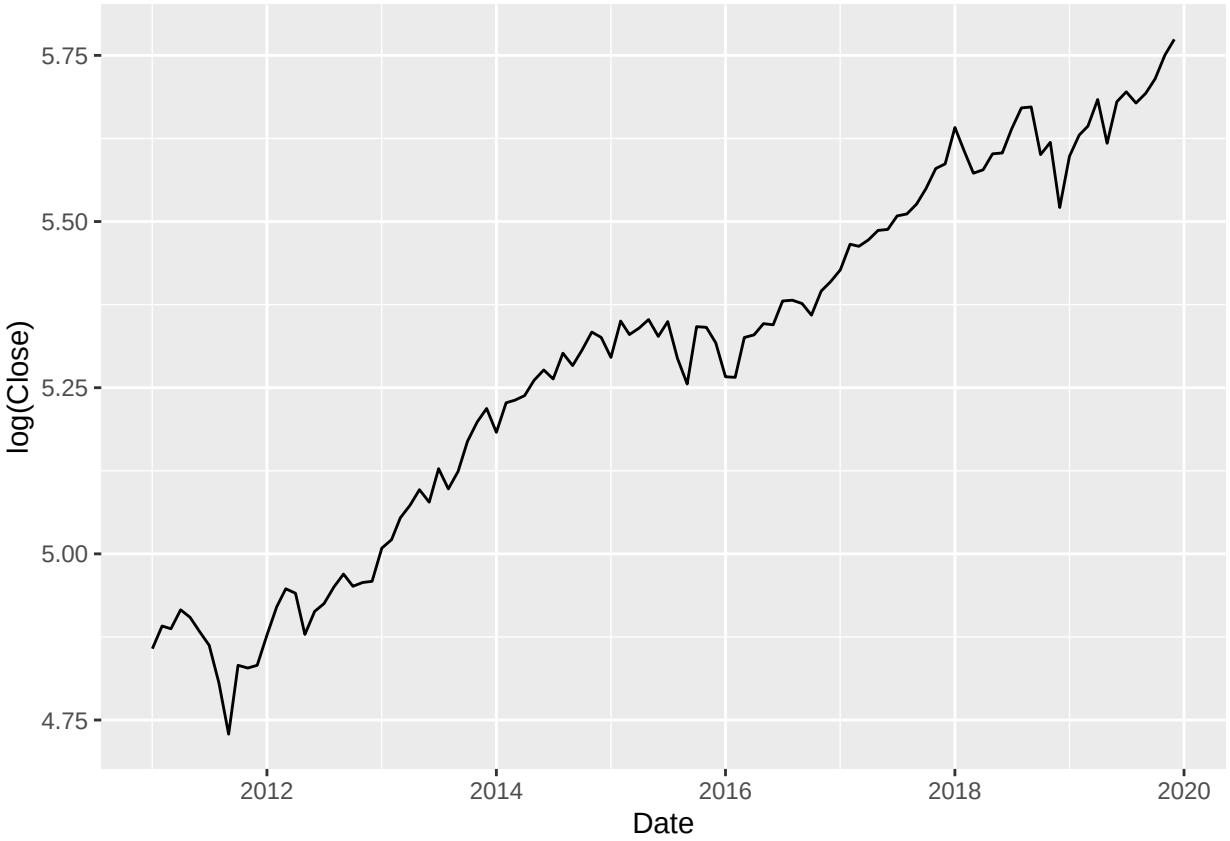
```
BoxCox.ar(SandPTS)$ci
```




```
[1] -0.3  0.5
```

The box-cox test contains zero, therefore we will use a log transformation.

```
ggplot(SandP, aes(x = Date, y = log(Close))) + geom_line()
```



The time series plot looks better after the log transformation.

```
SandPTS <- ts(log(SandPTS))
adf.test(SandPTS)
```

```
    Augmented Dickey-Fuller Test

data:  SandPTS
Dickey-Fuller = -2.2506, Lag order = 4, p-value = 0.4727
alternative hypothesis: stationary
```

```
SandPdiff <- diff(SandPTS)
```

The dickey-fuller test indicates that the base model is not stationary.

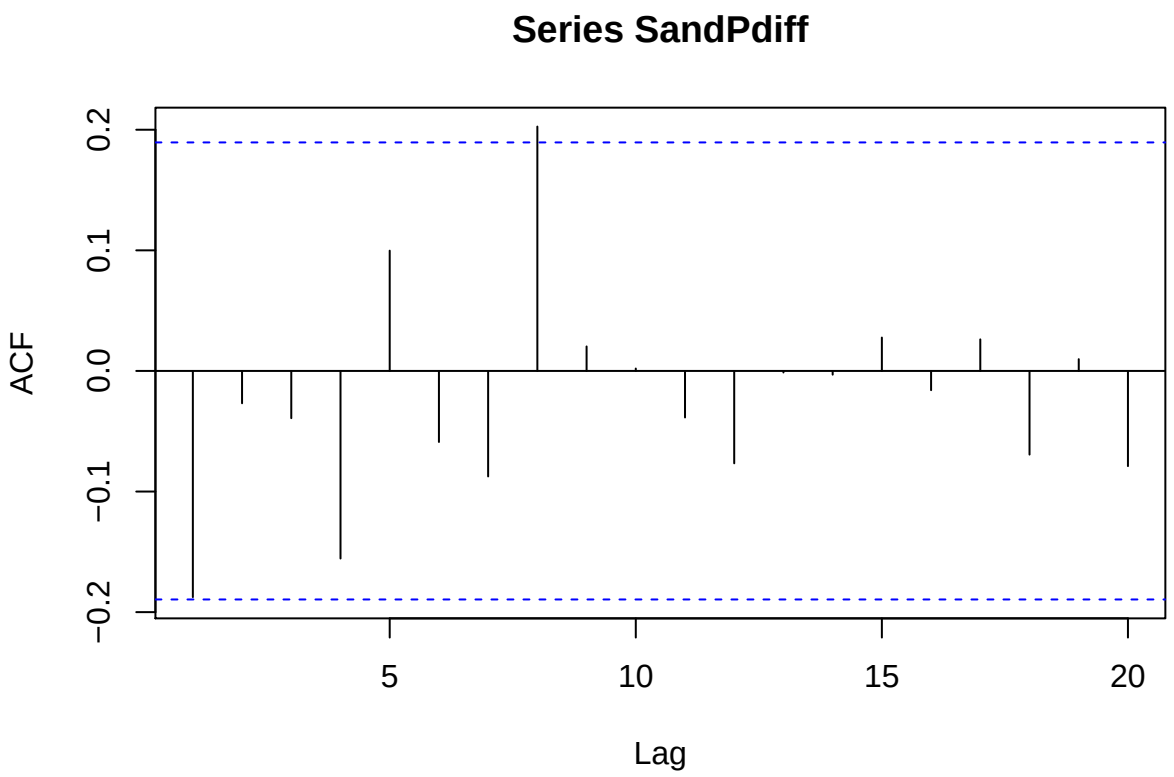
```
adf.test(SandPdiff)
```

```
Warning in adf.test(SandPdiff): p-value smaller than printed p-value
```

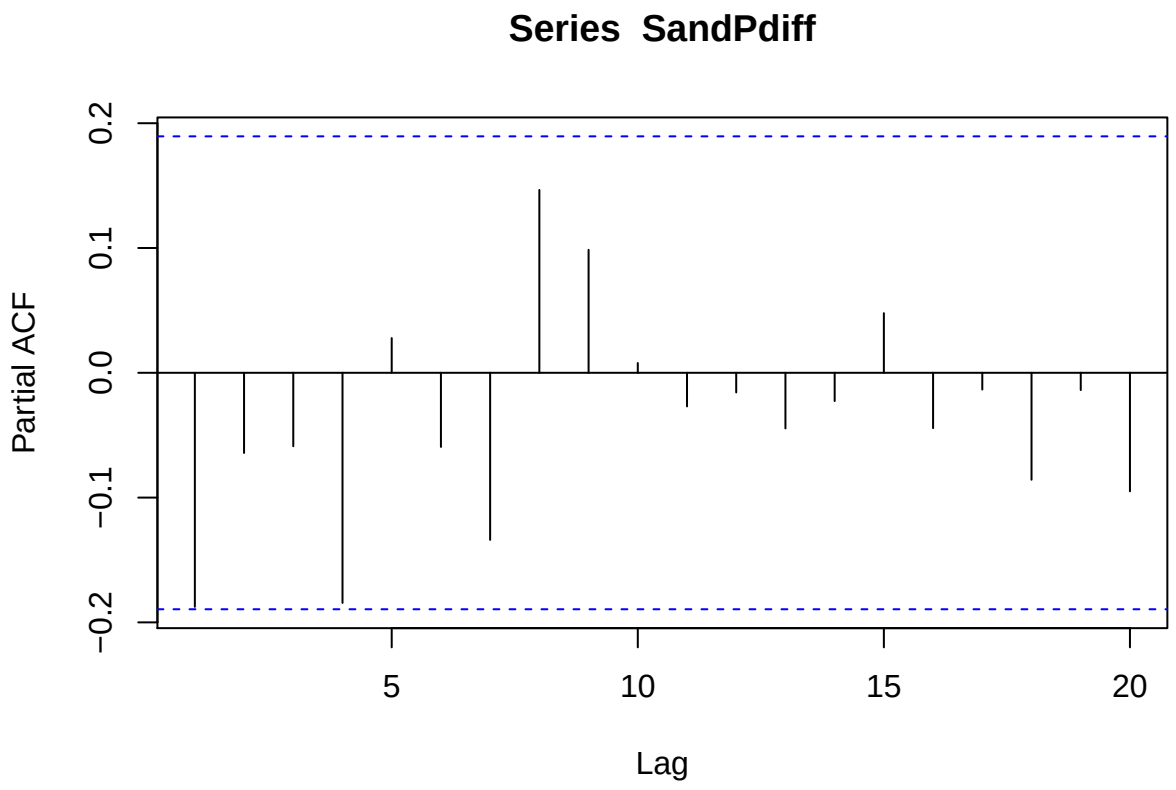
```
    Augmented Dickey-Fuller Test

data:  SandPdiff
Dickey-Fuller = -5.2713, Lag order = 4, p-value = 0.01
alternative hypothesis: stationary
```

The model is stationary after differencing it once.

```
acf(SandPdiff)
```




```
pacf(SandPdiff)
```

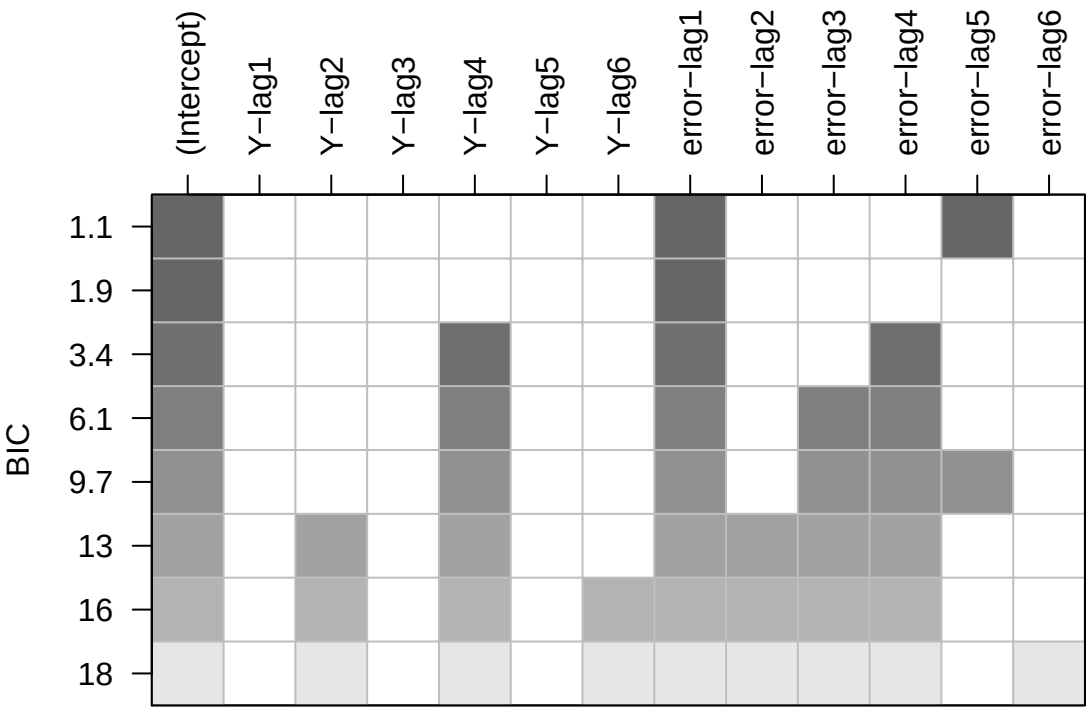



```
eacf(SandPdiff)
```

```
AR/MA
  0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 o o o o o o o x o o o  o  o  o
1 x o o o o o o o o o o  o  o  o
2 x o o o o o o o o o o  o  o  o
3 x x x o o o o o o o o  o  o  o
4 o x x x o o o o o o o  o  o  o
5 x o x o o o o o o o o  o  o  o
6 x x o x o o o o o o o  o  o  o
7 x x x x x o o o o o o  o  o  o
```

The partial acf output does not have any lags outside of the expected errors.

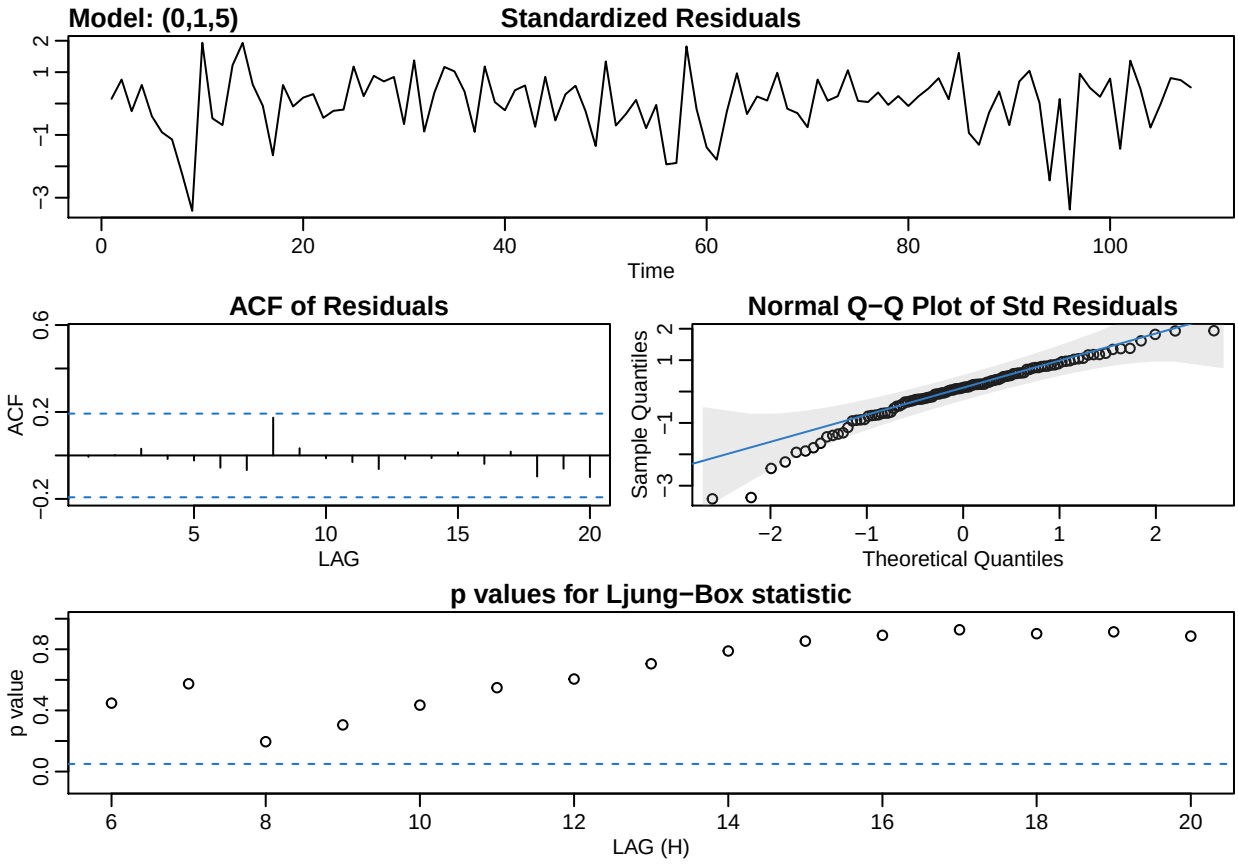
```
plot(armasubsets(SandPdiff, nar = 6, nma = 6))
```



There are several models that may be appropriate.

```
sarima(SandPTS, 0, 1, 5)
```

```
initial  value -3.389534
iter   2 value -3.426072
iter   3 value -3.429313
iter   4 value -3.432067
iter   5 value -3.432534
iter   6 value -3.432565
iter   7 value -3.432568
iter   8 value -3.432569
iter   8 value -3.432569
iter   8 value -3.432569
final  value -3.432569
converged
initial  value -3.431690
iter   2 value -3.431702
iter   3 value -3.431739
iter   4 value -3.431739
iter   5 value -3.431739
iter   5 value -3.431739
iter   5 value -3.431739
final  value -3.431739
converged
```



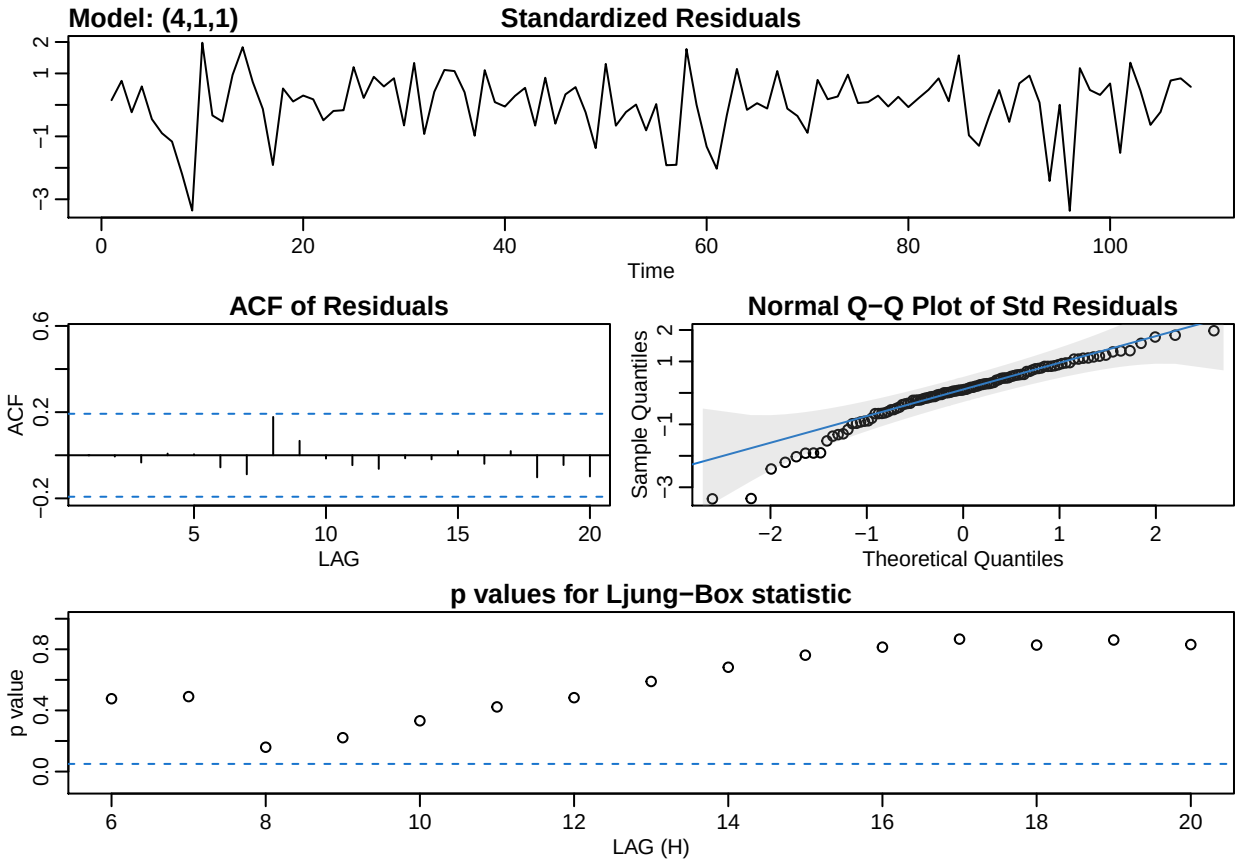

```
$fit

Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
    Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,
    optim.control = list(trace = trc, REPORT = 1, reltol = tol))

Coefficients:
          ma1      ma2      ma3      ma4     ma5  constant
      -0.2029  -0.0512  -0.1419  -0.1098  0.1505    0.0083
s.e.   0.0957   0.0948   0.1067   0.0873  0.1120    0.0020

sigma^2 estimated as 0.001043:  log likelihood = 215.37,  aic = -416.74

$degrees_of_freedom
[1] 101

$ttable
         Estimate     SE t.value p.value
ma1       -0.2029 0.0957 -2.1208  0.0364
ma2       -0.0512 0.0948 -0.5405  0.5900
ma3       -0.1419 0.1067 -1.3299  0.1866
ma4       -0.1098 0.0873 -1.2578  0.2114
ma5        0.1505 0.1120  1.3435  0.1821
constant   0.0083 0.0020  4.0915  0.0001
```

```
$AIC
[1] -3.894761

$AICc
[1] -3.88691

$BIC
[1] -3.719903
```

```r
SandP015 <- arima(SandPTS, order = c(0, 1, 5))
```

```r
sarima(SandPTS, 4, 1, 1)
```

```
initial  value -3.377146
iter   2 value -3.401971
iter   3 value -3.415597
iter   4 value -3.415927
iter   5 value -3.415928
iter   6 value -3.415966
iter   7 value -3.415967
iter   8 value -3.415967
iter   9 value -3.415971
iter  10 value -3.415984
iter  11 value -3.416001
iter  12 value -3.416015
iter  13 value -3.416035
iter  14 value -3.416078
iter  15 value -3.416173
iter  16 value -3.416186
iter  17 value -3.416193
iter  18 value -3.416194
iter  19 value -3.416195
iter  20 value -3.416197
iter  21 value -3.416204
iter  22 value -3.416212
iter  23 value -3.416216
iter  24 value -3.416217
iter  24 value -3.416217
iter  24 value -3.416217
final  value -3.416217
converged
initial  value -3.428989
iter   2 value -3.429007
iter   3 value -3.429041
iter   4 value -3.429048
iter   5 value -3.429057
iter   6 value -3.429133
iter   7 value -3.429154
iter   8 value -3.429160
iter   9 value -3.429161
iter   9 value -3.429161
iter   9 value -3.429161
final  value -3.429161
converged
```



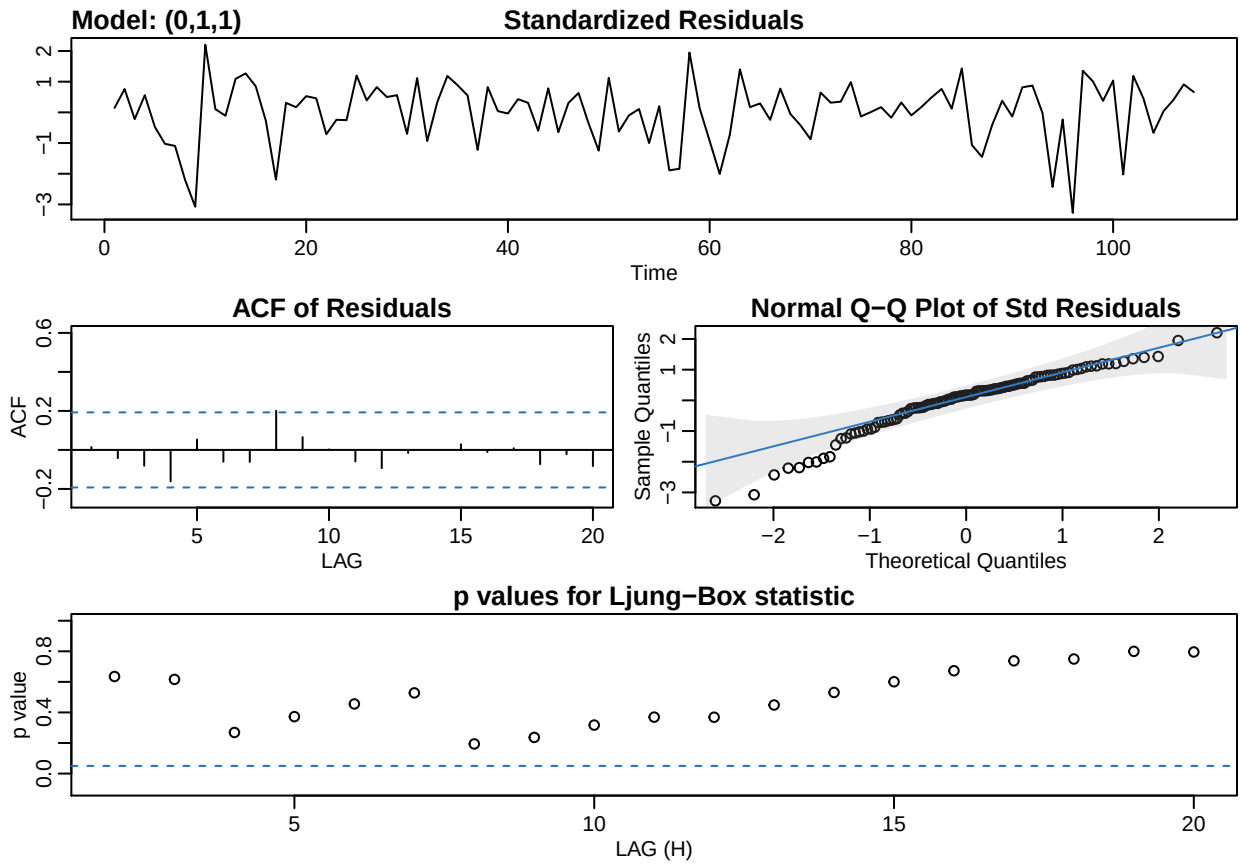

```
$fit

Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
    Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,
    optim.control = list(trace = trc, REPORT = 1, reltol = tol))

Coefficients:
          ar1      ar2      ar3      ar4     ma1  constant
      -0.3799  -0.1272  -0.1120  -0.1933  0.1705    0.0084
s.e.   0.4225   0.1312   0.1052   0.0948  0.4274    0.0020

sigma^2 estimated as 0.001049:  log likelihood = 215.09,  aic = -416.19

$degrees_of_freedom
[1] 101

$ttable
         Estimate     SE t.value p.value
ar1       -0.3799 0.4225 -0.8991  0.3707
ar2       -0.1272 0.1312 -0.9700  0.3343
ar3       -0.1120 0.1052 -1.0653  0.2893
ar4       -0.1933 0.0948 -2.0393  0.0440
ma1        0.1705 0.4274  0.3989  0.6908
constant   0.0084 0.0020  4.0944  0.0001
```

```
$AIC
[1] -3.889603

$AICc
[1] -3.881753

$BIC
[1] -3.714745
```

```r
SandP411 <- arima(SandPTS, order = c(4, 1, 1))
```

```r
sarima(SandPTS, 0, 1, 1)
```

```
initial  value -3.389534
iter   2 value -3.409647
iter   3 value -3.410121
iter   4 value -3.410121
iter   5 value -3.410122
iter   5 value -3.410122
iter   5 value -3.410122
final  value -3.410122
converged
initial  value -3.410009
iter   2 value -3.410010
iter   2 value -3.410010
iter   2 value -3.410010
final  value -3.410010
converged
```




```
$fit

Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
    Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,
    optim.control = list(trace = trc, REPORT = 1, reltol = tol))

Coefficients:
          ma1  constant
      -0.2196    0.0085
s.e.   0.1049    0.0025

sigma^2 estimated as 0.001091:  log likelihood = 213.04,  aic = -420.09

$degrees_of_freedom
[1] 105

$ttable
         Estimate     SE t.value p.value
ma1       -0.2196 0.1049 -2.0935  0.0387
constant   0.0085 0.0025  3.3897  0.0010

$AIC
[1] -3.926068

$AICc
```

```
[1] -3.92499

$BIC
[1] -3.851129
```

```r
SandP011 <- arima(SandPTS, order = c(0, 1, 1))
```

```r
sarima(SandPTS, 4, 1, 4)
```

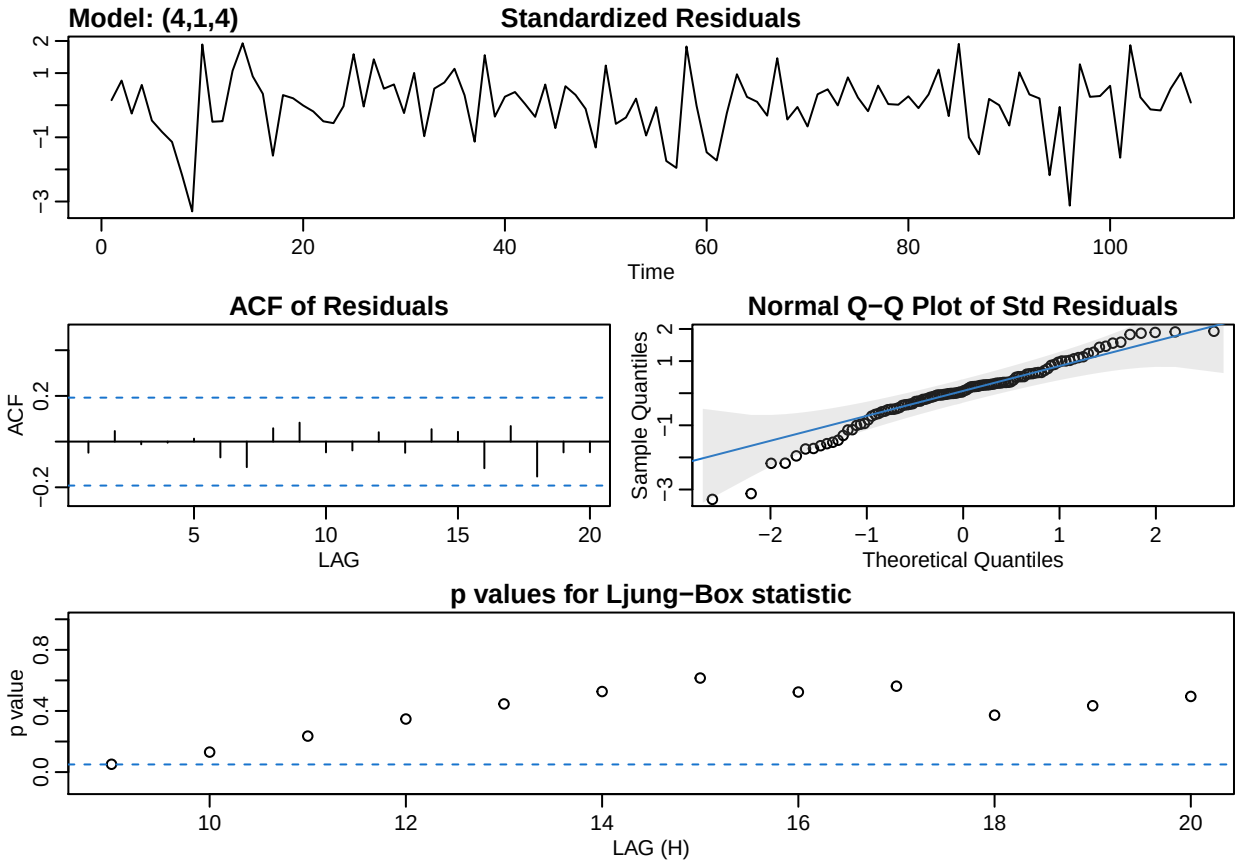
```
initial  value -3.377146
iter   2 value -3.381571
iter   3 value -3.414100
iter   4 value -3.415456
iter   5 value -3.416942
iter   6 value -3.421574
iter   7 value -3.423061
iter   8 value -3.423075
iter   9 value -3.430900
iter  10 value -3.433793
iter  11 value -3.437280
iter  12 value -3.442074
iter  13 value -3.455092
iter  14 value -3.469776
iter  15 value -3.476636
iter  16 value -3.477753
iter  17 value -3.483887
iter  18 value -3.498283
iter  19 value -3.511488
iter  20 value -3.515535
iter  21 value -3.520911
iter  22 value -3.526419
iter  23 value -3.530148
iter  24 value -3.532614
iter  25 value -3.537018
iter  26 value -3.537705
iter  27 value -3.541374
iter  28 value -3.544623
iter  29 value -3.550804
iter  30 value -3.557485
iter  31 value -3.558324
iter  32 value -3.558635
iter  33 value -3.569101
iter  34 value -3.572167
iter  35 value -3.577111
iter  36 value -3.580384
iter  37 value -3.582056
iter  38 value -3.585357
iter  39 value -3.587814
iter  40 value -3.591282
iter  41 value -3.596998
iter  42 value -3.601076
iter  43 value -3.606648
iter  44 value -3.610394
```

```
iter  45 value -3.615493
iter  46 value -3.618377
iter  47 value -3.630736
iter  48 value -3.633882
iter  49 value -3.634357
iter  50 value -3.635496
iter  51 value -3.635593
iter  52 value -3.635651
iter  53 value -3.635652
iter  54 value -3.637554
iter  55 value -3.644007
iter  56 value -3.644471
iter  57 value -3.644512
iter  58 value -3.644518
iter  59 value -3.650902
iter  60 value -3.650996
iter  61 value -3.651405
iter  62 value -3.651796
iter  63 value -3.651842
iter  64 value -3.651868
iter  65 value -3.651881
iter  66 value -3.652588
iter  67 value -3.652590
iter  68 value -3.652855
iter  69 value -3.654111
iter  69 value -3.654111
iter  70 value -3.656558
iter  71 value -3.656705
iter  72 value -3.656707
iter  73 value -3.656715
iter  74 value -3.657192
iter  75 value -3.657278
iter  76 value -3.657455
iter  76 value -3.657455
iter  77 value -3.658695
iter  78 value -3.659087
iter  79 value -3.659157
iter  80 value -3.659158
iter  81 value -3.659410
iter  81 value -3.659410
iter  82 value -3.660128
iter  83 value -3.660188
iter  84 value -3.660195
iter  85 value -3.660196
iter  86 value -3.660242
iter  86 value -3.660242
iter  87 value -3.660979
iter  88 value -3.661056
iter  89 value -3.661067
iter  90 value -3.661067
iter  91 value -3.661121
iter  92 value -3.661121
iter  92 value -3.661121
iter  93 value -3.661652
```

```
iter  94 value -3.661706
iter  95 value -3.661709
iter  96 value -3.661710
iter  97 value -3.661773
iter  98 value -3.661776
iter  98 value -3.661776
iter  99 value -3.662000
iter 100 value -3.662013
final  value -3.662013
stopped after 100 iterations
initial  value -3.389534
iter   2 value -3.394042
iter   3 value -3.425211
iter   4 value -3.426235
iter   5 value -3.428060
iter   6 value -3.428752
iter   7 value -3.430533
iter   8 value -3.436020
iter   9 value -3.437143
iter  10 value -3.439016
iter  11 value -3.439701
iter  12 value -3.440176
iter  13 value -3.440580
iter  14 value -3.441016
iter  15 value -3.441175
iter  16 value -3.441473
iter  17 value -3.441661
iter  18 value -3.441818
iter  19 value -3.441889
iter  20 value -3.441954
iter  21 value -3.441964
iter  22 value -3.441972
iter  23 value -3.441978
iter  24 value -3.441990
iter  25 value -3.442009
iter  26 value -3.442046
iter  27 value -3.442086
iter  28 value -3.442120
iter  29 value -3.442150
iter  30 value -3.442166
iter  31 value -3.442170
iter  32 value -3.442171
iter  33 value -3.442172
iter  34 value -3.442190
iter  35 value -3.442202
iter  36 value -3.442220
iter  37 value -3.442232
iter  38 value -3.442251
iter  39 value -3.442267
iter  40 value -3.442270
iter  41 value -3.442276
iter  42 value -3.442288
iter  43 value -3.442413
iter  44 value -3.442700
```

```
iter  45 value -3.442761
iter  46 value -3.443164
iter  47 value -3.443769
iter  48 value -3.444018
iter  49 value -3.444088
iter  50 value -3.444520
iter  51 value -3.444760
iter  52 value -3.445327
iter  53 value -3.445645
iter  54 value -3.445967
iter  55 value -3.446487
iter  56 value -3.446544
iter  57 value -3.447217
iter  58 value -3.447483
iter  59 value -3.447605
iter  60 value -3.447714
iter  61 value -3.447756
iter  62 value -3.447791
iter  63 value -3.448078
iter  64 value -3.448301
iter  65 value -3.448597
iter  66 value -3.448911
iter  67 value -3.449349
iter  68 value -3.449793
iter  69 value -3.449857
iter  70 value -3.450223
iter  71 value -3.450564
iter  72 value -3.450818
iter  73 value -3.451043
iter  74 value -3.451534
iter  75 value -3.451791
iter  76 value -3.451842
iter  77 value -3.451985
iter  78 value -3.452023
iter  79 value -3.452047
iter  80 value -3.452079
iter  81 value -3.452112
iter  82 value -3.452205
iter  83 value -3.452303
iter  84 value -3.452367
iter  85 value -3.452401
iter  86 value -3.452466
iter  87 value -3.452588
iter  88 value -3.452854
iter  89 value -3.452974
iter  90 value -3.453064
iter  91 value -3.453137
iter  92 value -3.453283
iter  93 value -3.453428
iter  94 value -3.453483
iter  95 value -3.453932
iter  96 value -3.454026
iter  97 value -3.454042
iter  98 value -3.454093
```

```
iter  99 value -3.454116
iter 100 value -3.454122
final  value -3.454122
stopped after 100 iterations

Warning in stats::arima(xdata, order = c(p, d, q), seasonal = list(order =
c(P, : possible convergence problem: optim gave code = 1
```




```
$fit

Call:
stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
    Q), period = S), xreg = constant, transform.pars = trans, fixed = fixed,
    optim.control = list(trace = trc, REPORT = 1, reltol = tol))

Coefficients:
          ar1     ar2     ar3      ar4      ma1      ma2      ma3     ma4
      -0.0872  0.2124  0.2970  -0.7305  -0.0333  -0.3422  -0.4045  0.7385
s.e.   0.2373  0.3354  0.4684   0.3945   0.2392   0.3539   0.4179  0.3117
       constant
         0.0082
s.e.     0.0022

sigma^2 estimated as 0.0009612:  log likelihood = 217.76,  aic = -415.53
```

```
$degrees_of_freedom
[1] 98

$ttable
         Estimate     SE t.value p.value
ar1       -0.0872 0.2373 -0.3675  0.7141
ar2        0.2124 0.3354  0.6333  0.5280
ar3        0.2970 0.4684  0.6342  0.5274
ar4       -0.7305 0.3945 -1.8517  0.0671
ma1       -0.0333 0.2392 -0.1391  0.8897
ma2       -0.3422 0.3539 -0.9669  0.3360
ma3       -0.4045 0.4179 -0.9681  0.3354
ma4        0.7385 0.3117  2.3695  0.0198
constant   0.0082 0.0022  3.7213  0.0003

$AIC
[1] -3.88345

$AICc
[1] -3.866107

$BIC
[1] -3.633653
```

```
SandP414 <- arima(SandPTS, order = c(4, 1, 4), fixed = c(0, 0, NA, NA, 0, 0, NA, NA))
```

```
Warning in stats::arima(x = x, order = order, seasonal = seasonal, xreg =
xreg, : some AR parameters were fixed: setting transform.pars = FALSE
```

The Lyung-Box statistic is the best for arima(0, 1, 5), but all of the MAs are insignificant except for the first one, so we decided to use arima(0, 1,)

```
shapiro.test(rstandard(SandP011))
```

```
    Shapiro-Wilk normality test

data:  rstandard(SandP011)
W = 0.9633, p-value = 0.004484
```

The residuals do not appear to be normal because of a few outliers on the lower end on the qqnorm plot.

```
runs(rstandard(SandP011))
```

```
$pvalue
[1] 0.389

$observed.runs
[1] 41

$expected.runs
```

```
[1] 45.2037

$n1
[1] 31

$n2
[1] 77

$k
[1] 0
```

There is no statistical evidence that the observations are not independent.

$\hat{Y}_t = Y_{t-1} + .2196e_{t-1}$

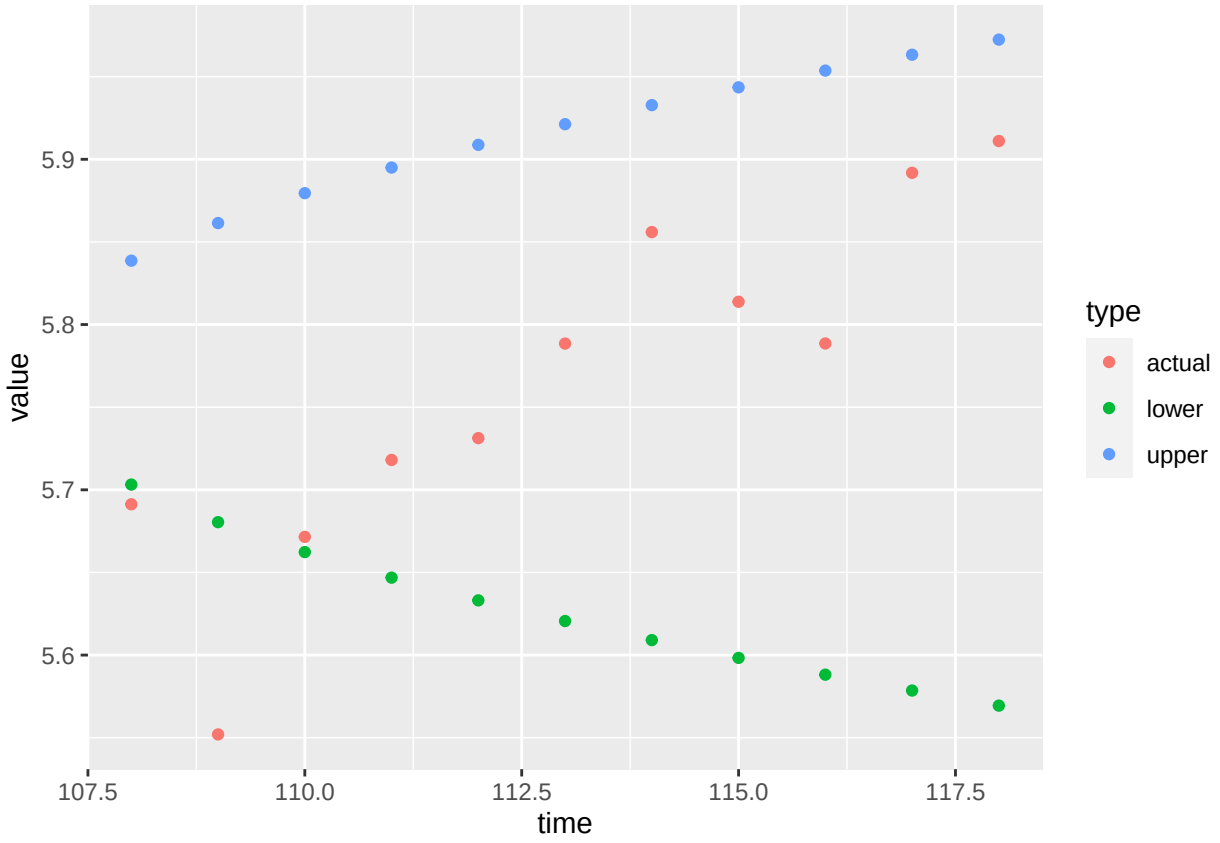
```
SandP.predict <- predict(SandP011, n.ahead=11)
lower.end <-(SandP.predict$pred - qnorm(0.975,0,1) * SandP.predict$se)
upper.end <-(SandP.predict$pred + qnorm(0.975,0,1) * SandP.predict$se)
```

```
predictsand <- data.frame(time = 108:118, lower.end, upper.end, actual = log(SandP2$Close))
lowerp <- cbind(predictsand$time, predictsand$lower.end)
upperp <- cbind(predictsand$time, predictsand$upper.end)
actualp <- cbind(predictsand$time, predictsand$actual)
predicts <- as.data.frame(rbind(lowerp, upperp, actualp))
colnames(predicts) <- c("time", "value")
predicts$type <- c(rep("lower", 11), rep("upper", 11), rep("actual", 11))
predicts
```

```
   time    value   type
1   108 5.703185  lower
2   109 5.680421  lower
3   110 5.662329  lower
4   111 5.646847  lower
5   112 5.633094  lower
6   113 5.620594  lower
7   114 5.609056  lower
8   115 5.598288  lower
9   116 5.588153  lower
10  117 5.578552  lower
11  118 5.569407  lower
12  108 5.838648  upper
13  109 5.861412  upper
14  110 5.879504  upper
15  111 5.894985  upper
16  112 5.908738  upper
17  113 5.921239  upper
18  114 5.932776  upper
19  115 5.943544  upper
20  116 5.953679  upper
21  117 5.963281  upper
22  118 5.972426  upper
23  108 5.691237 actual
24  109 5.551990 actual
```

```
25  110 5.671535 actual
26  111 5.718080 actual
27  112 5.731268 actual
28  113 5.788491 actual
29  114 5.855960 actual
30  115 5.813802 actual
31  116 5.788552 actual
32  117 5.891810 actual
33  118 5.911041 actual
```

```
ggplot(predicts, aes(time, value)) + geom_point(aes(color = type))
```



Only the first two predictions missed the forecast, and the first barely missed it. The forecasting was actually fairly good for the model.