# 15_linear_regression_vs_exp_smooth.R

felixreichel

2021-11-16

```
# Course: Time series analysis
# Exercise: 15th / Linear Regression vs. Exponential Smoothing
#                  with deterministic seasonal pattern
# Author: Felix Reichel
```

```
require(astsa)
```

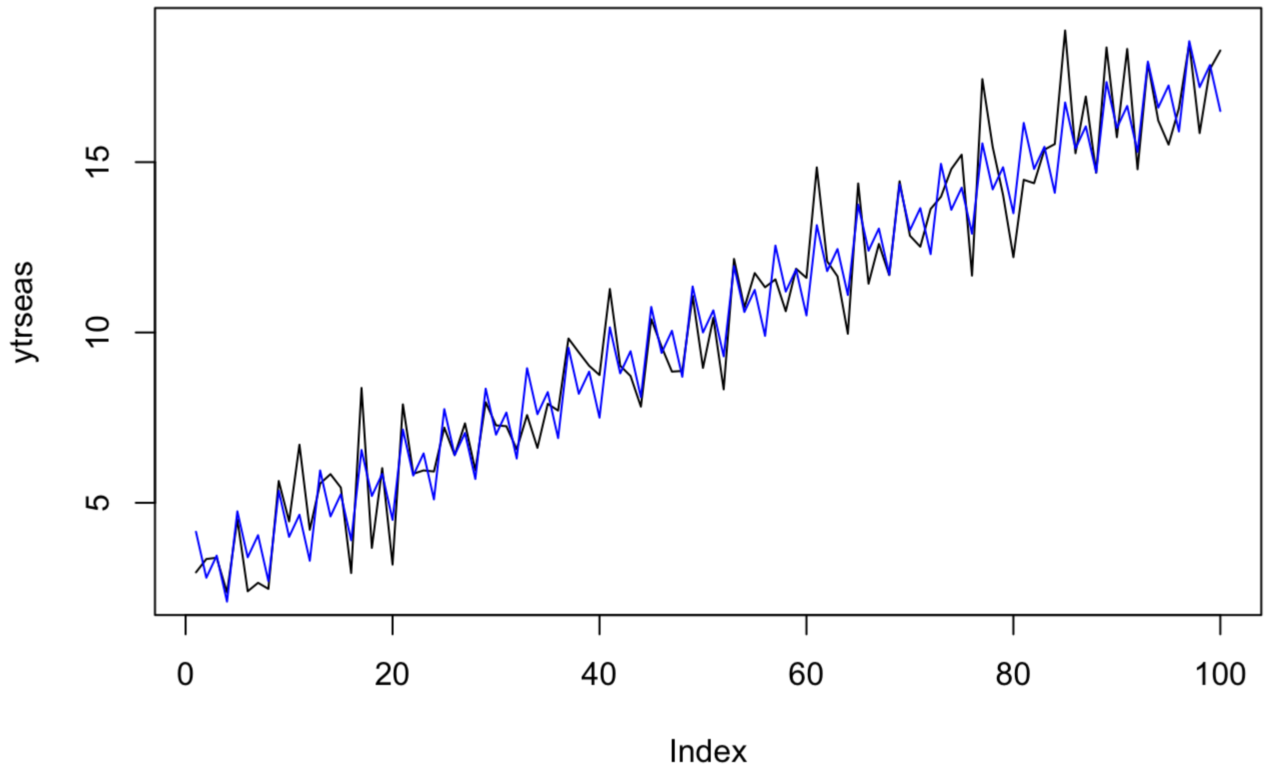
```
## Loading required package: astsa
```

```
require(tseries)
```

```
## Loading required package: tseries
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
# Simulate data from a model with linear trend and a deterministic (quarterly)
# seasonal pattern (T = 100, σ^2 = 1) and plot the series.

TT <- 100
sigma <- sqrt(1)
periods <- 4

ltr <-1:TT
seas_pattern <- c(4, 2.5, 3, 1.5)
seas <- rep(seas_pattern, TT/periods)

a = 0.15

set.seed(2345)

err <- rnorm(TT)*sigma
ytrseas <- a*ltr + seas + err

plot(ytrseas, type="l")
lines(a * ltr + seas, col="blue",xlab="t")
```



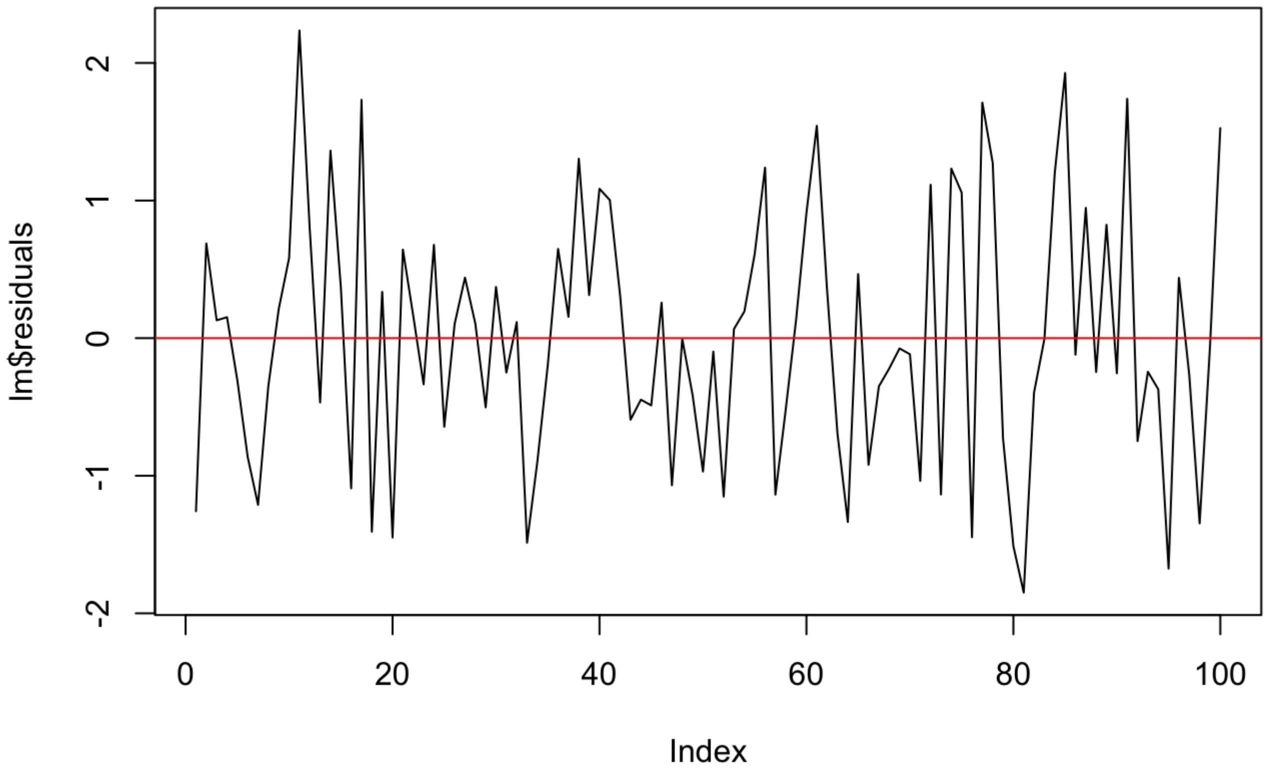

```
# Fit a linear regression model to the series and perform a residual analysis

s <- C(as.factor(rep(1:periods, TT/periods)), contr.sum)
lm = lm(ytrseas ~ ltr + s)
summary(lm)
```

```
##
## Call:
## lm(formula = ytrseas ~ ltr + s)
##
## Residuals:
##      Min        1Q    Median        3Q       Max
## -1.84920 -0.60569 -0.03493  0.61930  2.23592
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.708772   0.188229  14.391  < 2e-16 ***
## ltr          0.151426   0.003237  46.786  < 2e-16 ***
## s1           1.356511   0.161771   8.385 4.61e-13 ***
## s2          -0.350346   0.161706  -2.167   0.0328 *
## s3           0.094974   0.161706   0.587   0.5584
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9336 on 95 degrees of freedom
## Multiple R-squared:  0.9595, Adjusted R-squared:  0.9578
## F-statistic: 563.4 on 4 and 95 DF,  p-value: < 2.2e-16
```

```r
# Residual analysis
lm$residuals
```

```
##            1            2            3            4            5            6 
## -1.258133552  0.688023467  0.129571361  0.152104752 -0.307010500 -0.864252761 
##            7            8            9           10           11           12 
## -1.211692827 -0.345323016  0.211698109  0.581693124  2.235917156  0.785491492 
##           13           14           15           16           17           18 
## -0.467798184  1.362618154  0.372242861 -1.091738737  1.732076271 -1.406626137 
##           19           20           21           22           23           24 
##  0.336371623 -1.449087602  0.643429948  0.162036705 -0.335680301  0.677767538 
##           25           26           27           28           29           30 
## -0.644115995  0.103217192  0.439833086  0.105666878 -0.503267635  0.372148330 
##           31           32           33           34           35           36 
## -0.250068310  0.116319445 -1.486747185 -0.894600944 -0.198440105  0.648391565 
##           37           38           39           40           41           42 
##  0.154939617  1.304078520  0.312981180  1.085901691  1.003070703  0.305180853 
##           43           44           45           46           47           48 
## -0.592880179 -0.447292329 -0.488991101  0.257531867 -1.069486207 -0.006296200 
##           49           50           51           52           53           54 
## -0.414953479 -0.969505144 -0.097788822 -1.151652454  0.067031818  0.193538775 
##           55           56           57           58           59           60 
##  0.611257770  1.239090609 -1.137571174 -0.517939130  0.132400722  0.909845855 
##           61           62           63           64           65           66 
##  1.543165429  0.342127727 -0.695787478 -1.336362313  0.465650059 -0.920508338 
##           67           68           69           70           71           72 
## -0.350321159 -0.220795623 -0.074849508 -0.117032699 -1.037326713  1.114668318 
##           73           74           75           76           77           78 
## -1.136448830  1.231999034  1.057609523 -1.446835697  1.711465453  1.274032723 
##           79           80           81           82           83           84 
## -0.730689977 -1.511851731 -1.849198444 -0.394296735 -0.005608761  1.201771128 
##           85           86           87           88           89           90 
##  1.927210686 -0.120157493  0.946706403 -0.246054272  0.823928909 -0.255412065 
##           91           92           93           94           95           96 
##  1.739449491 -0.748565343 -0.244559694 -0.371451677 -1.675010881  0.438359334 
##           97           98           99          100 
## -0.270021722 -1.346443347 -0.063559458  1.526476712
```

```r
# Plot residuals
plot(lm$residuals, type="l")
abline(h = mean(lm$residuals), col="red")
```




```r
require(lmtest)
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

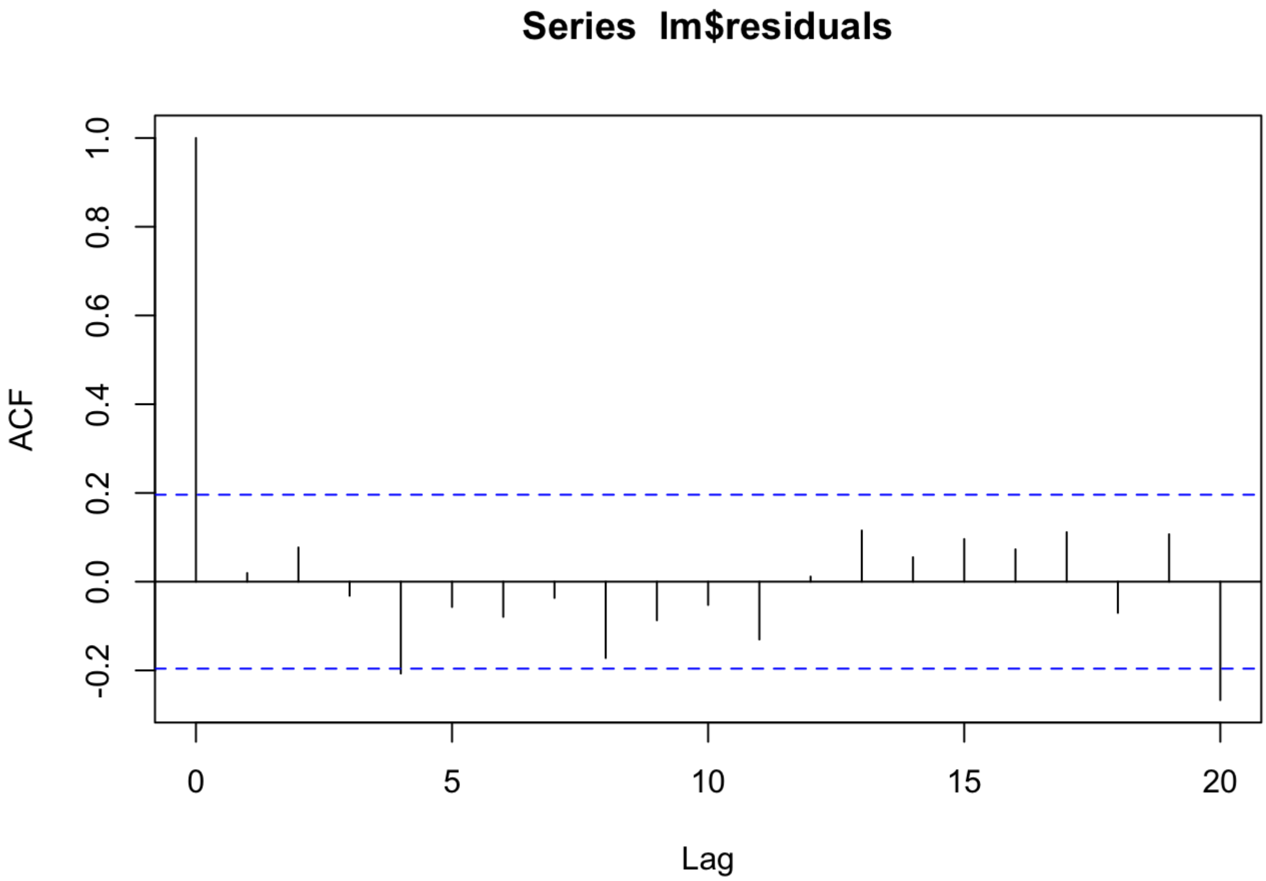
```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
# Durbin Watson
dwtest(lm) # one sided test: rho(1)>0
```

```
##
##  Durbin-Watson test
##
## data:  lm
## DW = 1.9137, p-value = 0.3332
## alternative hypothesis: true autocorrelation is greater than 0
```

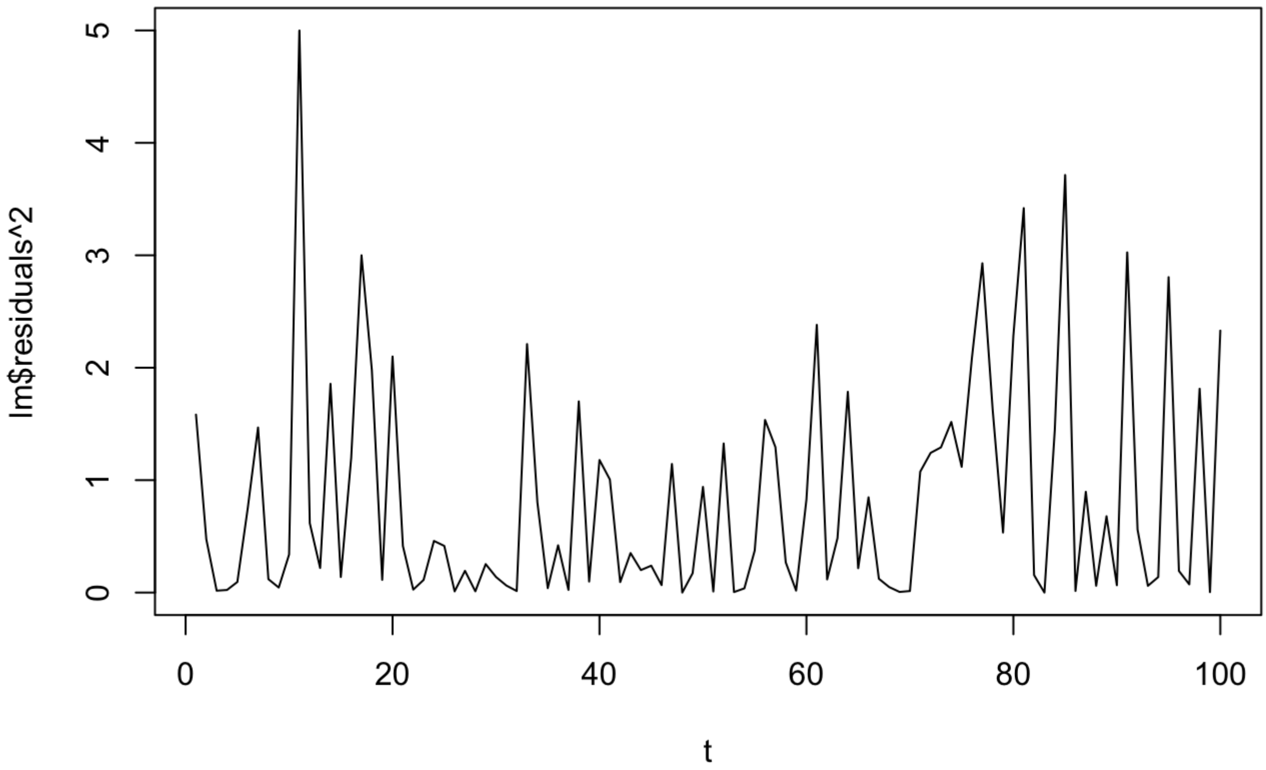
```r
acf(lm$residuals) # ci: (- 1.96/sqrt(TT),+ 1.96/sqrt(TT))
```




```
Box.test(lm$residuals, lag = 3, type = "Ljung")
```

```
##
##  Box-Ljung test
##
## data:  lm$residuals
## X-squared = 0.76427, df = 3, p-value = 0.858
```

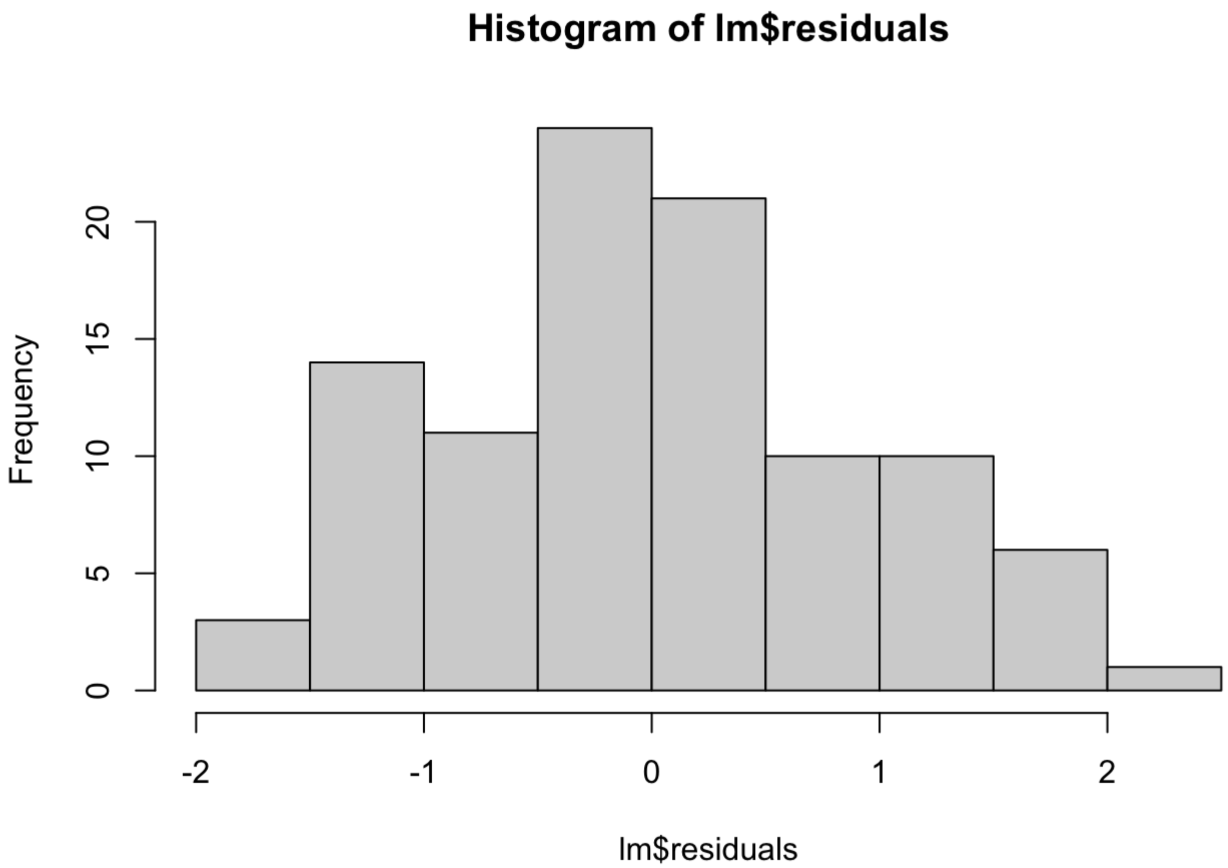
```
# heteroscedasticity
plot(lm$residuals^2,type="l",xlab="t")
```



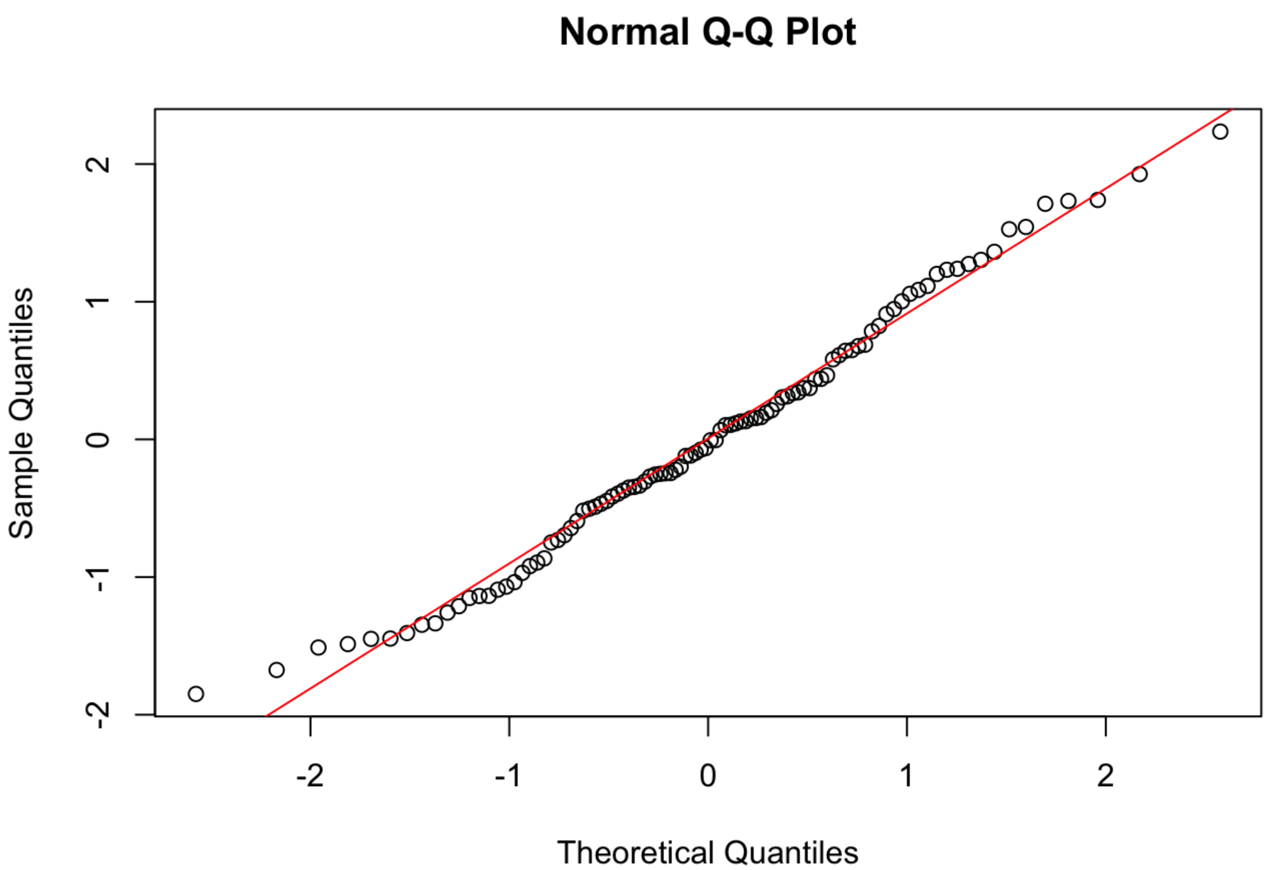

```
bptest(lm)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm
## BP = 3.9423, df = 4, p-value = 0.4139
```

```
# normal distribution
hist(lm$residuals)
```




```
qqnorm(lm$residuals)
qqline(lm$residuals,col="red")
```



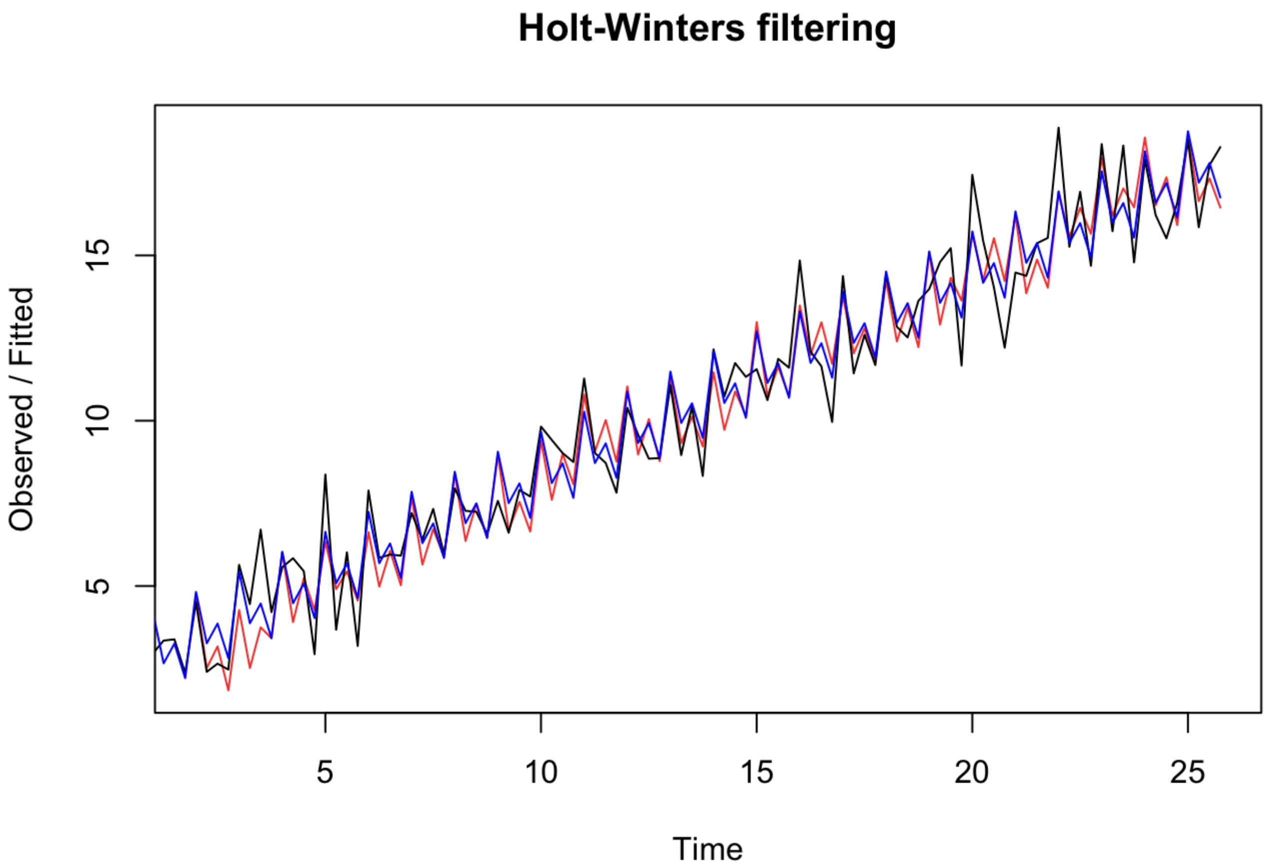
```
jarque.bera.test(lm$residuals)
```

```
##
##  Jarque Bera Test
##
## data:  lm$residuals
## X-squared = 1.8349, df = 2, p-value = 0.3995
```

```
# Use the appropriate exponential smoothing method for the data and compare
# the insample performance to that of the linear regression.

hw_exp_train <- HoltWinters(ts(ytrseas, frequency = periods), seasonal = "additive")

df <- data.frame(ltr = 1:TT, C(as.factor(rep(1:periods, TT/periods)), contr.sum))
lm_fit <- ts(predict(lm, newdata = df), frequency = periods)
```

```
## Warning: contrasts dropped from factor s
```

```
plot(hw_exp_train)
lines(lm_fit, col="blue")
```




```
# MSE, MAE and MAPE
require(Metrics)
```

```
## Loading required package: Metrics
```

```
hw_exp_fit_actual <- c(hw_exp_train$fitted[,1])
lm_fit_actual <- c(lm_fit)[5:100]
expected <- ytrseas[5:100]

mse(expected, hw_exp_fit_actual)
```

```
## [1] 1.140345
```

```
mse(expected, lm_fit_actual)
```

```
## [1] 0.8406305
```

```
mae(expected, hw_exp_fit_actual)
```

```
## [1] 0.8687842
```

```
mae(expected, lm_fit_actual)
```

```
## [1] 0.7430036
```

```
mape(expected, hw_exp_fit_actual)
```

```
## [1] 0.1002307
```

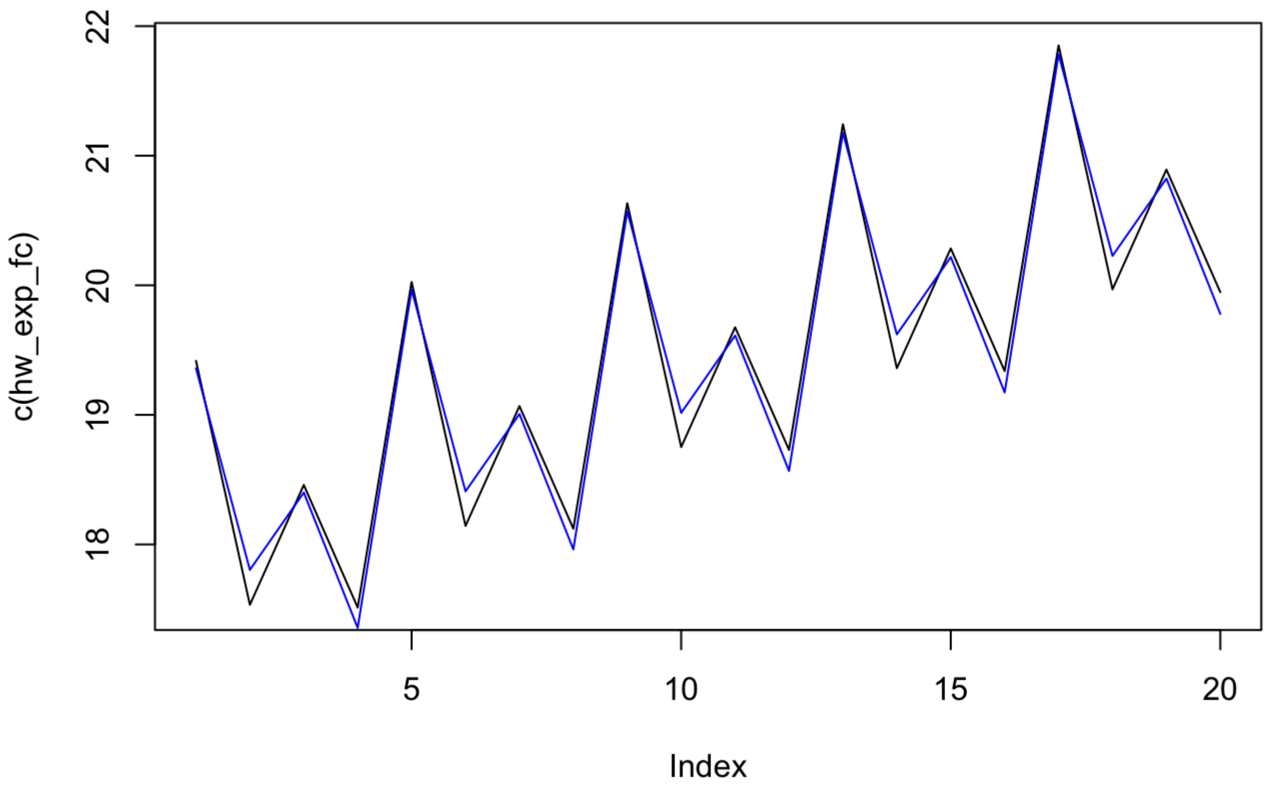
```
mape(expected, lm_fit_actual)
```

```
## [1] 0.08823866
```

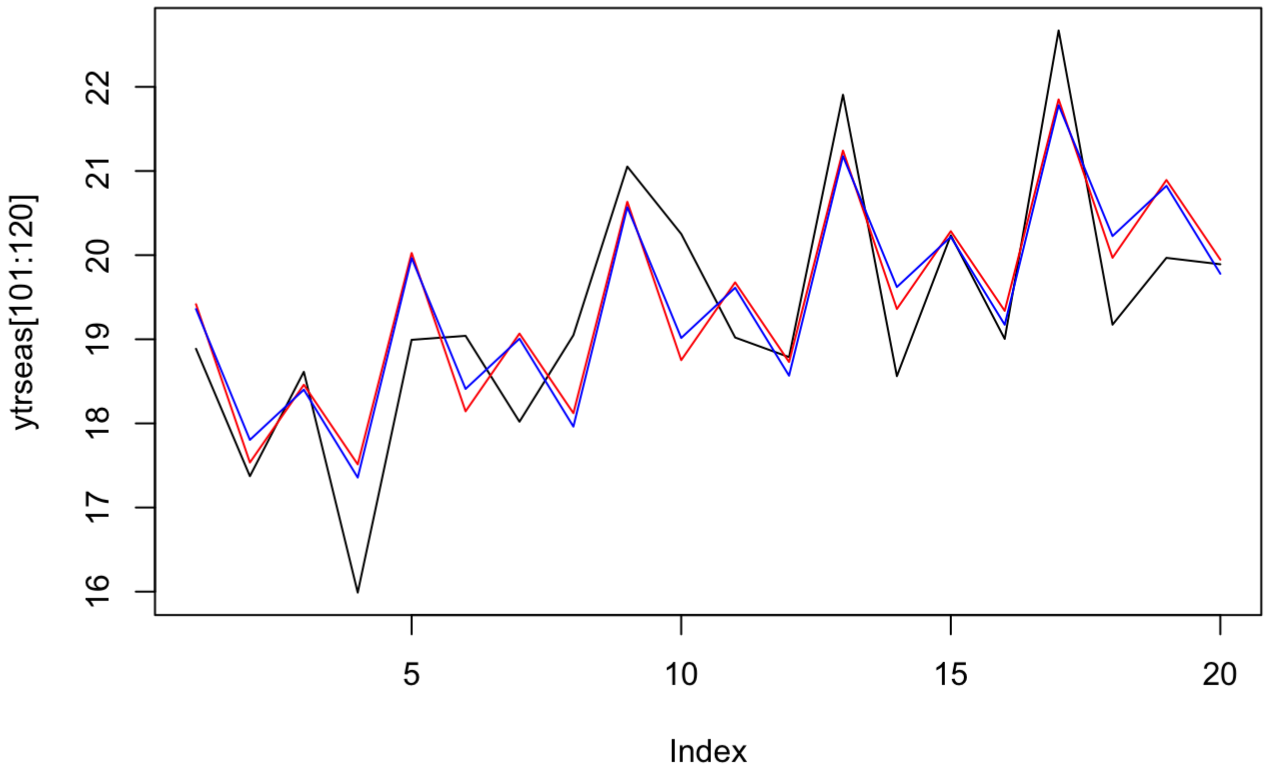
```
# Forecast the next 20 values with both methods.
fc <- 20
Tf <- TT+fc

lm_df = data.frame(ltr=1:(TT+fc), s = C(as.factor(rep(1:periods, Tf/periods)), contr.
sum))
lm_pred = predict(lm,newdata = lm_df)
```

```
## Warning: contrasts dropped from factor s
```

```
hw_exp_fc = predict(hw_exp_train, n.ahead = fc)

plot(c(hw_exp_fc), type="l")
lines(lm_pred[101:120], col="blue")
```




```
# Now simulate a realisation of these 20 values and compare out-of sample performanc
e.
# Plot the original and the new simulated data as well as the forecasts
# from both methods. Which method performs better?

ltr <-1:Tf
seas_pattern <- c(4, 2.5, 3, 1.5)
seas <- rep(seas_pattern, Tf/periods)

set.seed(2345)

err <- rnorm(Tf)*sigma
ytrseas <- a*ltr + seas + err

plot(ytrseas[101:120], type="l")
lines(c(hw_exp_fc), col="red")
lines(lm_pred[101:120], col="blue")
```




```
# MSE, MAE and MAPE
hw_exp_fit_actual <- c(hw_exp_fc)
lm_fit_actual <- c(lm_pred)[101:120]
expected <- ytrseas[101:120]

mse(expected, hw_exp_fit_actual)
```

```
## [1] 0.6346781
```

```
mse(expected, lm_fit_actual)
```

```
## [1] 0.6152005
```

```
mae(expected, hw_exp_fit_actual)
```

```
## [1] 0.6673394
```

```
mae(expected, lm_fit_actual)
```

```
## [1] 0.6786683
```

```
mape(expected, hw_exp_fit_actual)
```

```
## [1] 0.03504032
```

```
mape(expected, lm_fit_actual)
```

```
## [1] 0.03561631
```