Employment Analysis

Leala Darby, Scott Howard, Georgia Fardell

08/11/2020

Organise analysis in preparation for constructing poster.

First load all required packages:

```
library(car)
library(tseries)
library(astsa)
```

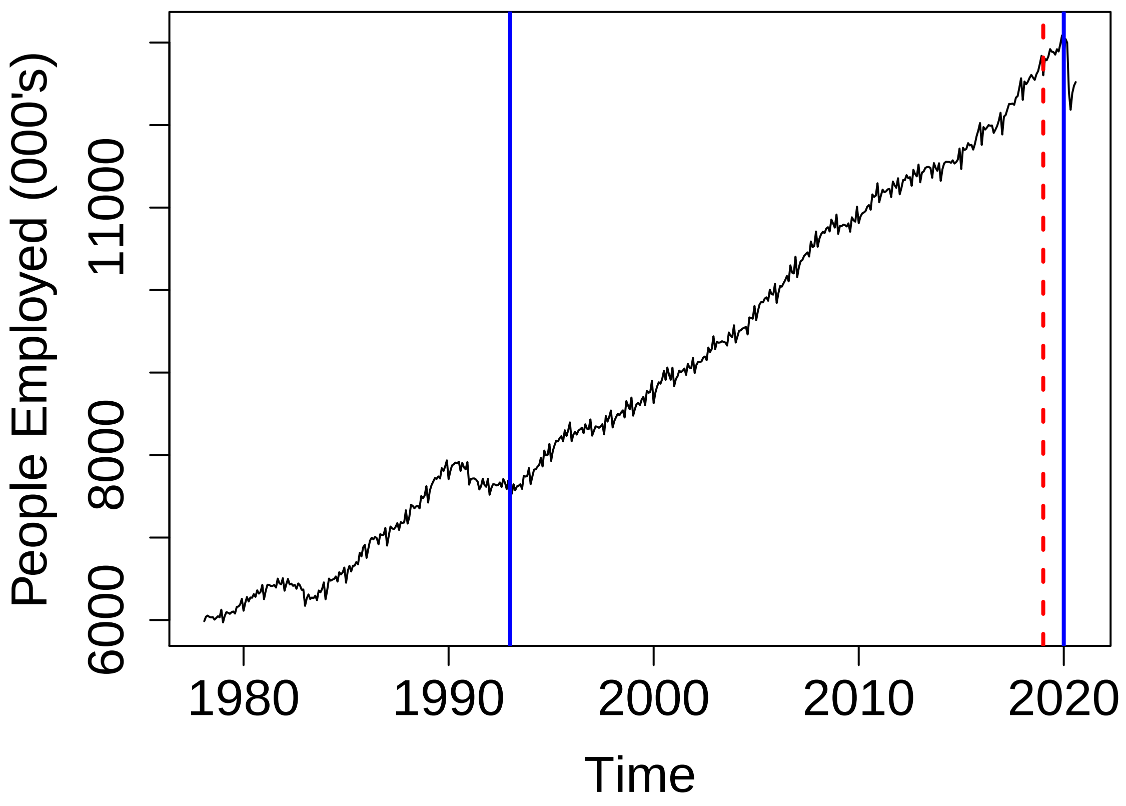
Load in the data:

```
dat <- read.csv("employment_data.csv", fileEncoding = 'UTF-8-BOM')
head(dat)
```

```
##   Observation.times Time.series.values
## 1            Feb-78             5985.7
## 2            Mar-78             6040.6
## 3            Apr-78             6054.2
## 4            May-78             6038.3
## 5            Jun-78             6031.3
## 6            Jul-78             6036.1
```

Create a time series object from the data and plot. The blue lines are visually detected structural breakpoints - contextual reasoning is recession in the 90s and COVID-19 in 2020. The red line indicates the training/test split.

```
ts_dat_test <- ts(dat[, 2], start = c(1978, 2), end = c(2020, 8), frequency = 12)
#jpeg("plot_ts.jpg", width = 990, height = 400)
par(cex.lab = 1.6, mar = c(5, 7, 1, 1))
plot.ts(ts_dat_test, ylab="People Employed (000's)", axes=FALSE) #updated units
axis(1, cex.axis = 1.6)
axis(2, cex.axis = 1.6)
box()
abline(v = 1993, col = "blue", lwd = 2)
abline(v = 2020, col = "blue", lwd = 2)
abline(v = 2019, col = "red", lty = 2, lwd = 2)
```



People Employed (000's)
6000
8000
11000
1980
1990
2000
2010
2020
Time

```
#dev.off()
```

Instructed to truncate data from January 1993 to December 2019 (inclusive)

```
dat[dat$Observation.times == "Jan-93",]
```

```
##     Observation.times Time.series.values
## 180            Jan-93             7533.7
```

```
dat[dat$Observation.times == "Dec-19",]
```

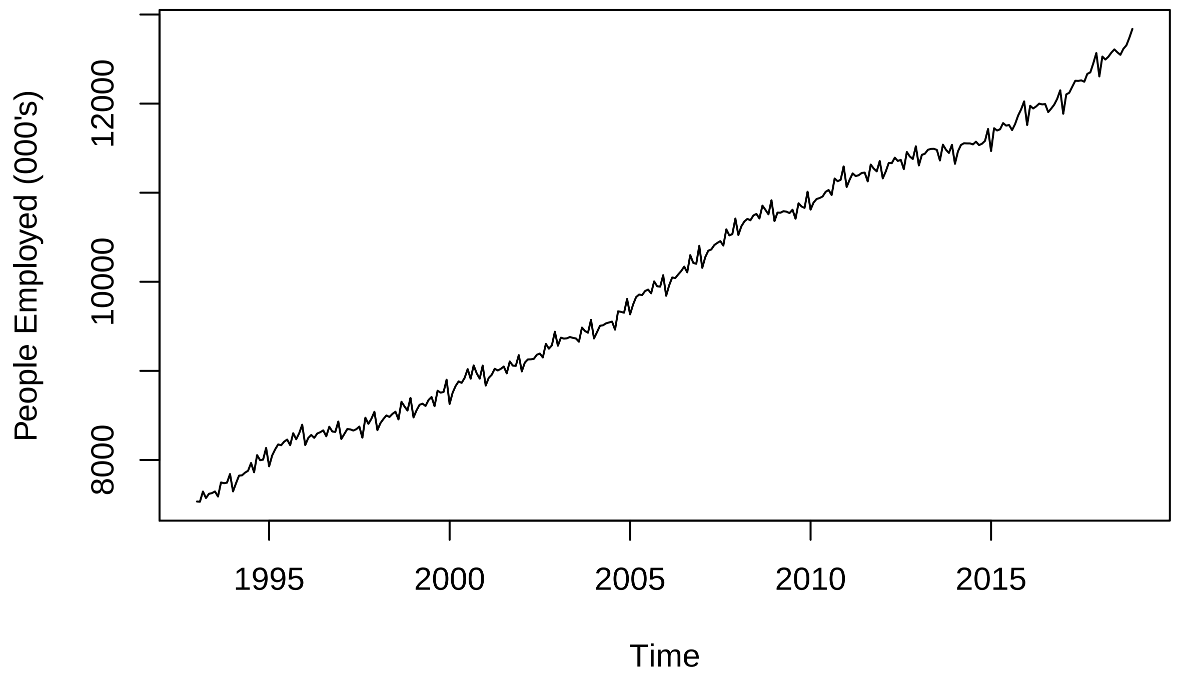
```
##     Observation.times Time.series.values
## 503            Dec-19            13087.1
```

```
dat[dat$Observation.times == "Jan-19",]
```

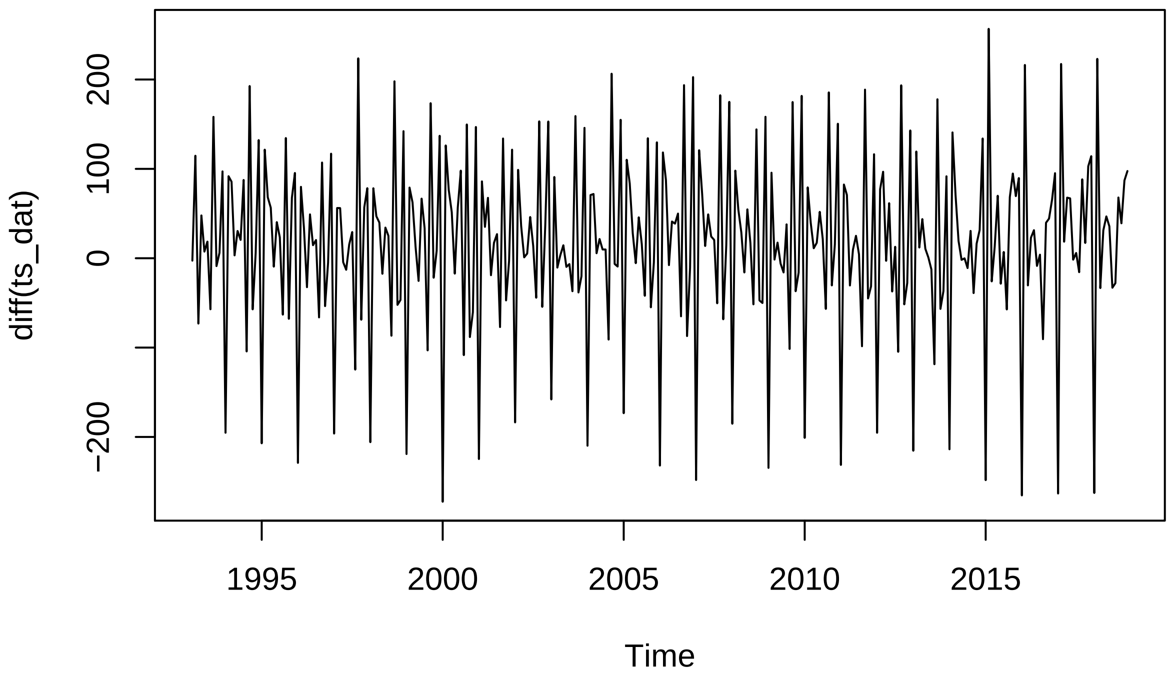
```
##     Observation.times Time.series.values
## 492            Jan-19            12603.1
```

We only need rows 180-503 for the truncated dataset. Splitting into training and test sets where the test set is all of 2019, we have train [180:491] and test [492:503].

```
trunc_dat <- dat[180:503,] # all data after truncating
train_dat <- dat[180:491,] # training data
test_dat <- dat[492:503,] # test data
join_dat <- dat[491:492,] # the month between training data and predictions (for plotting)
# ts for model fitting
ts_dat <- ts(train_dat[, 2], start = c(1993, 1), end = c(2018, 12), frequency = 12)
# ts for model testing
test_ts <- ts(test_dat[, 2], start = c(2019, 1), end = c(2019, 12), frequency = 12)
# between train and test
ts_join <- ts(join_dat[, 2], start = c(2018, 12), end = c(2019, 1), frequency = 12)
plot.ts(ts_dat, ylab="People Employed (000's)")
```



```r
plot.ts(diff(ts_dat)) # We are not actually taking the difference yet!
```



```
# This 2nd plot is just to help observe trends in variance.
```

The trend in mean is readily observable. Difficult to determine a trend in variance - there appears to be frequent changes, which are easier to see after incorporating lags of 1. Check statistically for stationarity using the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test, which has the following hypotheses:

$$H_o : \text{TS is stationary}$$
$$H_a : \text{TS in not stationary}$$

```r
kpss.test(ts_dat)
```

```
## Warning in kpss.test(ts_dat): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  ts_dat
## KPSS Level = 5.3101, Truncation lag parameter = 5, p-value = 0.01
```

The small p-value indicates that we should reject the null and conclude that the ts is not stationary.

As a rough test of constant variance (Levene's isn't really valid because time series data isn't independent)

```r
length(ts_dat)
```

```
## [1] 312
```

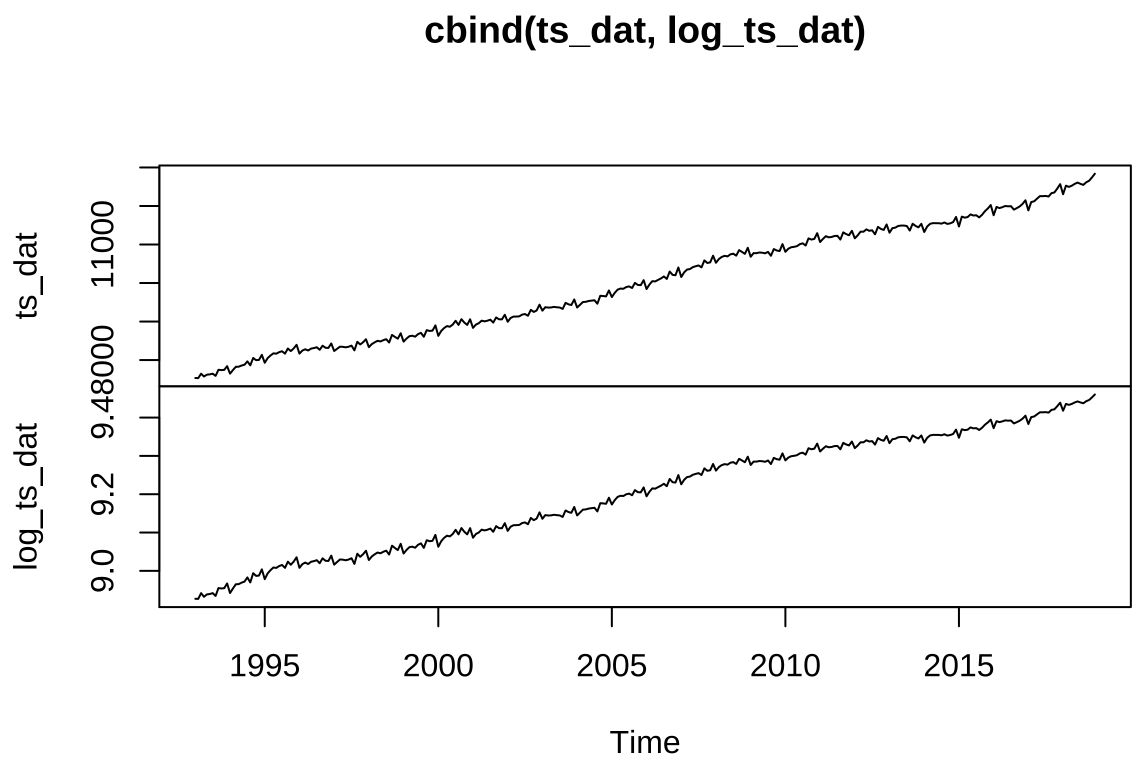
```r
Group <- c(rep(1,78), rep(2, 78), rep(3, 78), rep(4, 78))
leveneTest(ts_dat, Group)
```

```
## Warning in leveneTest.default(ts_dat, Group): Group coerced to factor.
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value  Pr(>F)
## group   3  2.6591 0.04837 *
##       308
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The small p-value of 0.04837 confirms that the data exhibits heteroscedasticity. Therefore we will perform a log transformation to attempt to reduce this:

```r
log_ts_dat <- log(ts_dat)
plot.ts(cbind(ts_dat, log_ts_dat))
```



```r
leveneTest(log_ts_dat, Group)
```

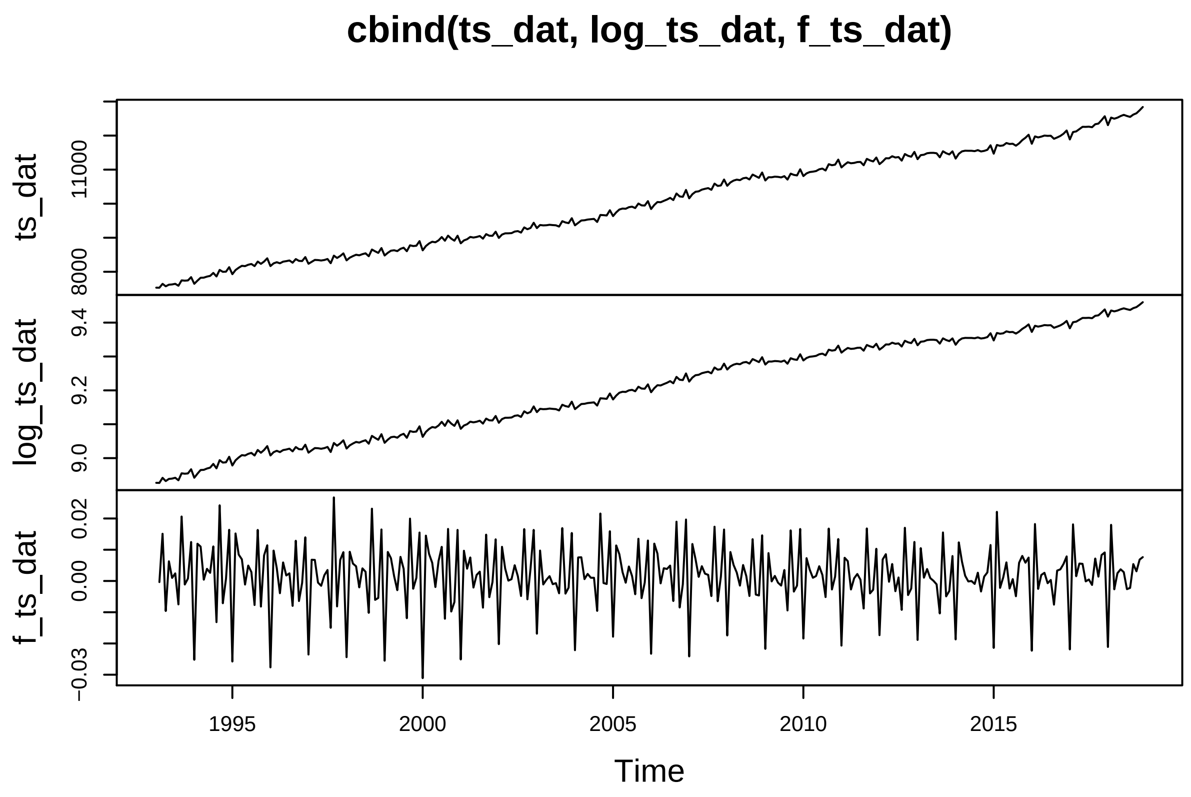
```
## Warning in leveneTest.default(log_ts_dat, Group): Group coerced to factor.
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   3  0.4451 0.7209
##       308
```

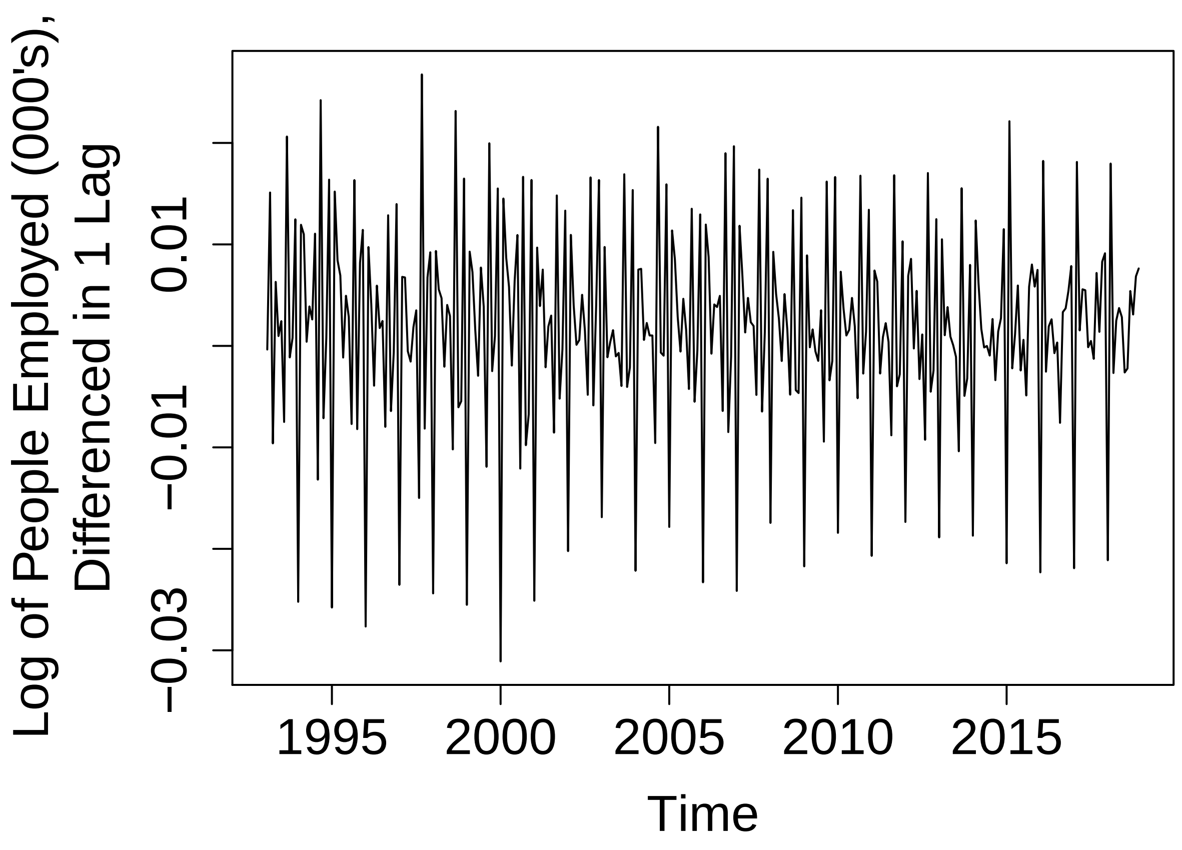
At a significance level of 5%, the p-value above of 0.7209 provides very weak evidence and we fail to reject the null hypothesis of equal variance among groups. Thus the heteroscedasticity has been reduced.

Next, to reduce the trend in mean, apply differencing of 1 lag to our TS with stabilised variance:

```r
f_ts_dat <- diff(log_ts_dat, 1)
plot.ts(cbind(ts_dat, log_ts_dat, f_ts_dat))
```



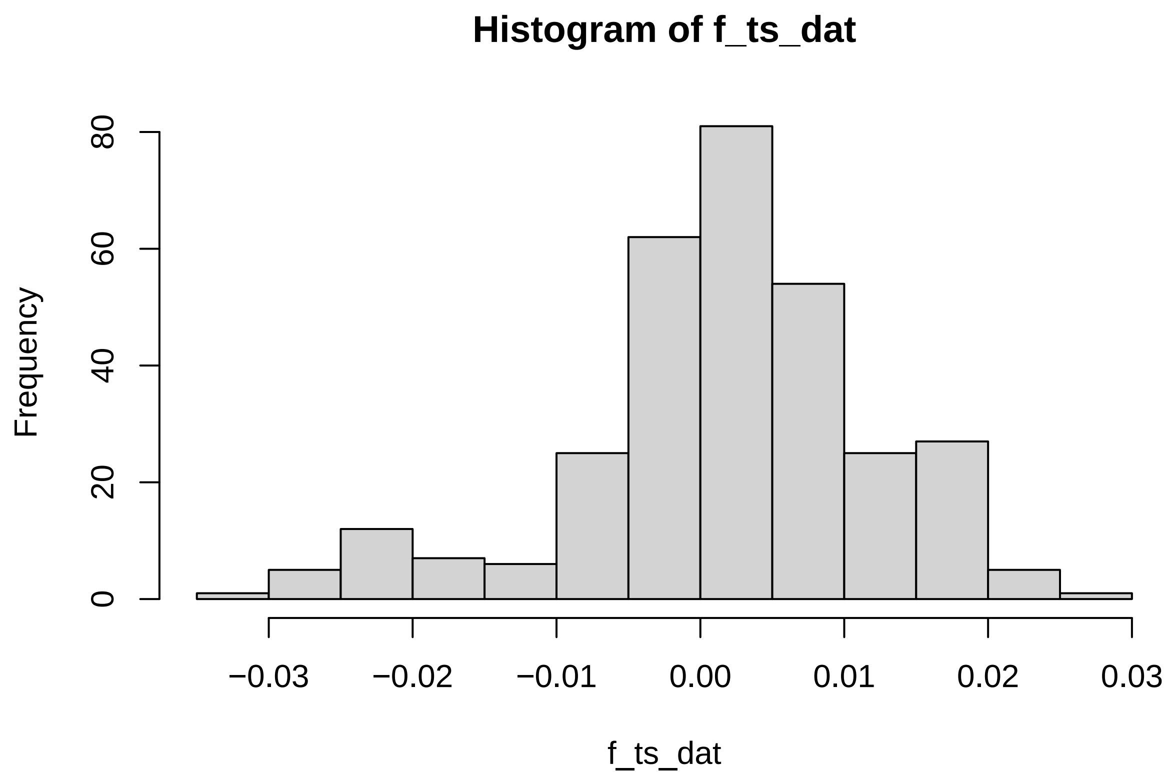
```r
#jpeg("stationary_ts.jpg", width = 990, height = 400)
par(cex.lab = 1.6, mar = c(5, 7, 1, 1))
plot.ts(f_ts_dat, ylab="Log of People Employed (000's), \nDifferenced in 1 Lag", axes=FALSE)
axis(1, cex.axis = 1.6)
axis(2, cex.axis = 1.6)
box()
```


Log of People Employed (000's),
Differenced in 1 Lag
0.01
−0.01
−0.03
1995
2000
2005
2010
2015
Time

```
#dev.off()
```

To confirm constant mean and variance and a Gaussian distribution for the time series, a Shapiro-Wilk normality test is performed:

```r
hist(f_ts_dat)
```



```r
shapiro.test(f_ts_dat)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  f_ts_dat
## W = 0.96193, p-value = 2.913e-07
```

The small p-value indicates likely non-normality, but this test isn't really valid for TS. Instead, check statistically for stationarity using the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test:

```
kpss.test(log_ts_dat)
```

```
## Warning in kpss.test(log_ts_dat): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  log_ts_dat
## KPSS Level = 5.3037, Truncation lag parameter = 5, p-value = 0.01
```

```
kpss.test(f_ts_dat)
```

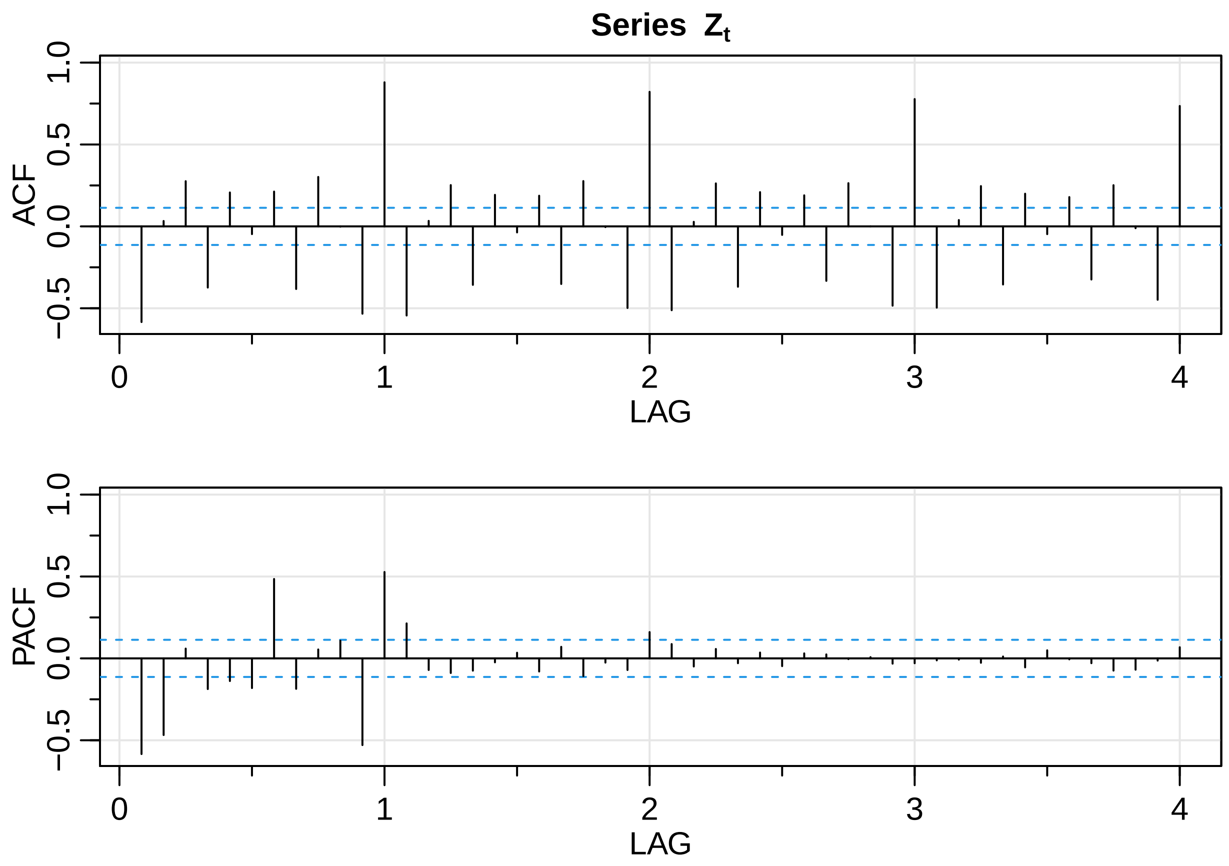
```
## Warning in kpss.test(f_ts_dat): p-value greater than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  f_ts_dat
## KPSS Level = 0.065128, Truncation lag parameter = 5, p-value = 0.1
```

The final ts has a high p-value of 0.1, which is statistically significant at a significance level of 5%. Therefore we fail to reject the null hypothesis, and have reasonable evidence that the final ts is stationary.

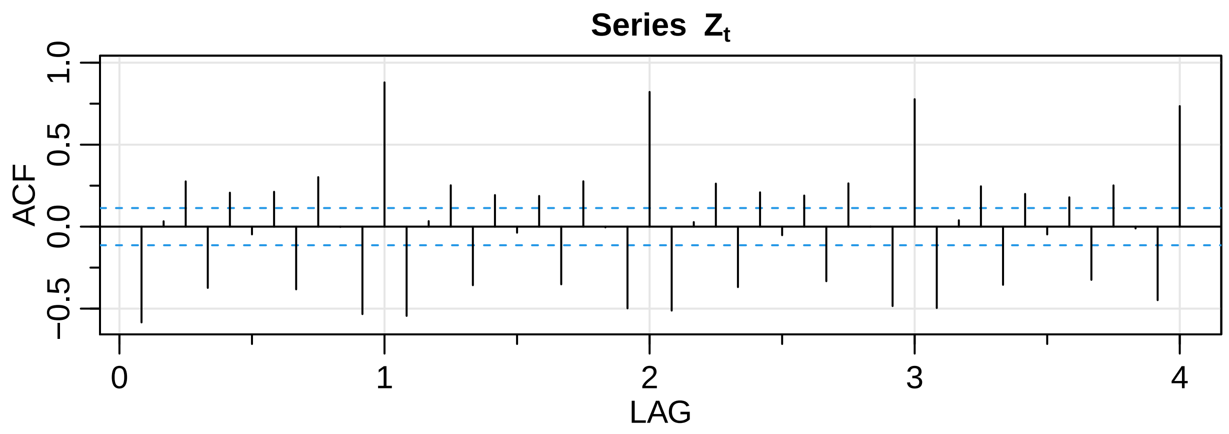
Next, the ACF and PACF of the differenced ts are plotted for analysis.

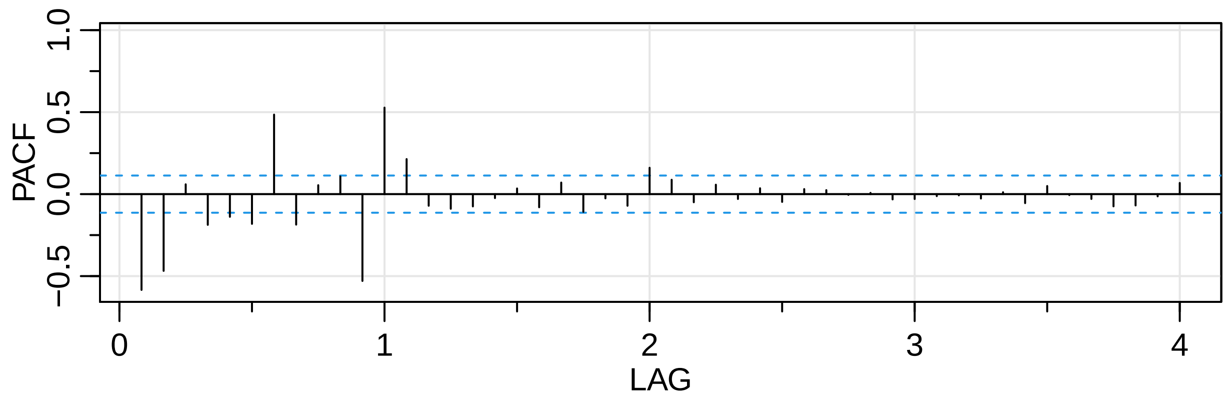
```
acf2(f_ts_dat, main=expression(bold("Series " ~ Z[t])))
```



```
##       [,1]  [,2] [,3]  [,4]  [,5]  [,6] [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF  -0.58  0.03 0.28 -0.37  0.21 -0.05 0.21 -0.38 0.30  0.00 -0.53  0.88 -0.54
## PACF -0.58 -0.47 0.06 -0.19 -0.14 -0.18 0.48 -0.19 0.05  0.11 -0.53  0.53  0.21
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF   0.03  0.25 -0.36  0.19 -0.04  0.19 -0.35  0.28  0.00 -0.50  0.82 -0.51
## PACF -0.07 -0.09 -0.08 -0.02  0.03 -0.08  0.07 -0.11 -0.03 -0.07  0.16  0.09
##      [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37]
## ACF   0.03  0.26 -0.37  0.21 -0.05  0.19 -0.33  0.26  0.00 -0.48  0.78 -0.50
## PACF -0.05  0.06 -0.03  0.04 -0.05  0.03  0.02  0.00  0.01 -0.03 -0.03 -0.01
##      [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48]
## ACF   0.04  0.25 -0.35  0.20 -0.05  0.18 -0.32  0.25 -0.01 -0.45  0.74
## PACF -0.01 -0.03  0.01 -0.05  0.05 -0.01 -0.03 -0.07 -0.07 -0.01  0.07
```

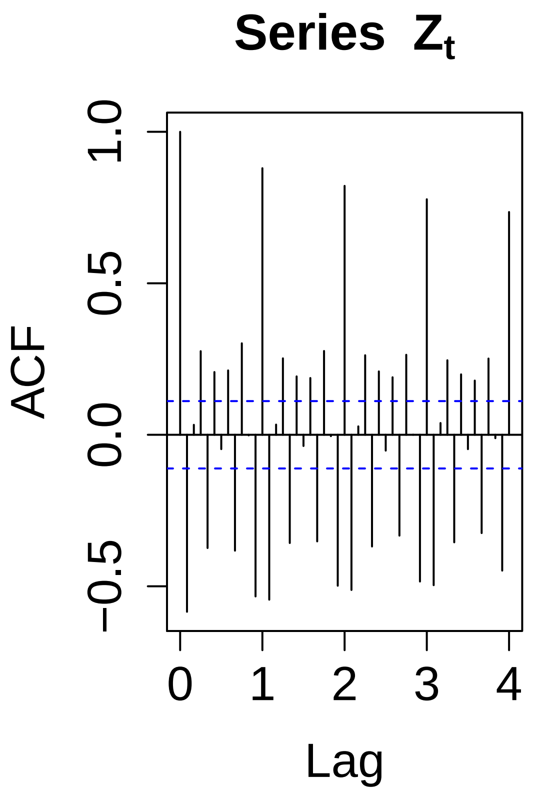
```
par(mfrow=c(1,2))
acf2(f_ts_dat, main=expression(bold("Series " ~ Z[t])))
```





```
##       [,1]  [,2] [,3]  [,4]  [,5]  [,6] [,7]  [,8] [,9] [,10] [,11] [,12] [,13]
## ACF  -0.58  0.03 0.28 -0.37  0.21 -0.05 0.21 -0.38 0.30  0.00 -0.53  0.88 -0.54
## PACF -0.58 -0.47 0.06 -0.19 -0.14 -0.18 0.48 -0.19 0.05  0.11 -0.53  0.53  0.21
##      [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24] [,25]
## ACF   0.03  0.25 -0.36  0.19 -0.04  0.19 -0.35  0.28  0.00 -0.50  0.82 -0.51
## PACF -0.07 -0.09 -0.08 -0.02  0.03 -0.08  0.07 -0.11 -0.03 -0.07  0.16  0.09
##      [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36] [,37]
## ACF   0.03  0.26 -0.37  0.21 -0.05  0.19 -0.33  0.26  0.00 -0.48  0.78 -0.50
## PACF -0.05  0.06 -0.03  0.04 -0.05  0.03  0.02  0.00  0.01 -0.03 -0.03 -0.01
##      [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48]
## ACF   0.04  0.25 -0.35  0.20 -0.05  0.18 -0.32  0.25 -0.01 -0.45  0.74
## PACF -0.01 -0.03  0.01 -0.05  0.05 -0.01 -0.03 -0.07 -0.07 -0.01  0.07
```

```
#jpeg("acf_ts.jpg", width = 990, height = 300)
par(cex.lab = 1.5, cex.axis = 1.5, cex.main = 1.6, mar = c(5, 5, 4, 2))
acf(f_ts_dat,lag.max=48,  main=expression(bold("Series " ~ Z[t])))
#dev.off()
```


Series Z_t
ACF
1.0
0.5
0.0
−0.5
0
1
2
3
4
Lag

Seasonal patterns in the ACF for TS_name (Figure [above]) show a slow decay in the dominant lags. To mitigate this, the time series was differenced in 12 lags giving [differenced model] for which the ACF and PACF are plotted in Figure [below].

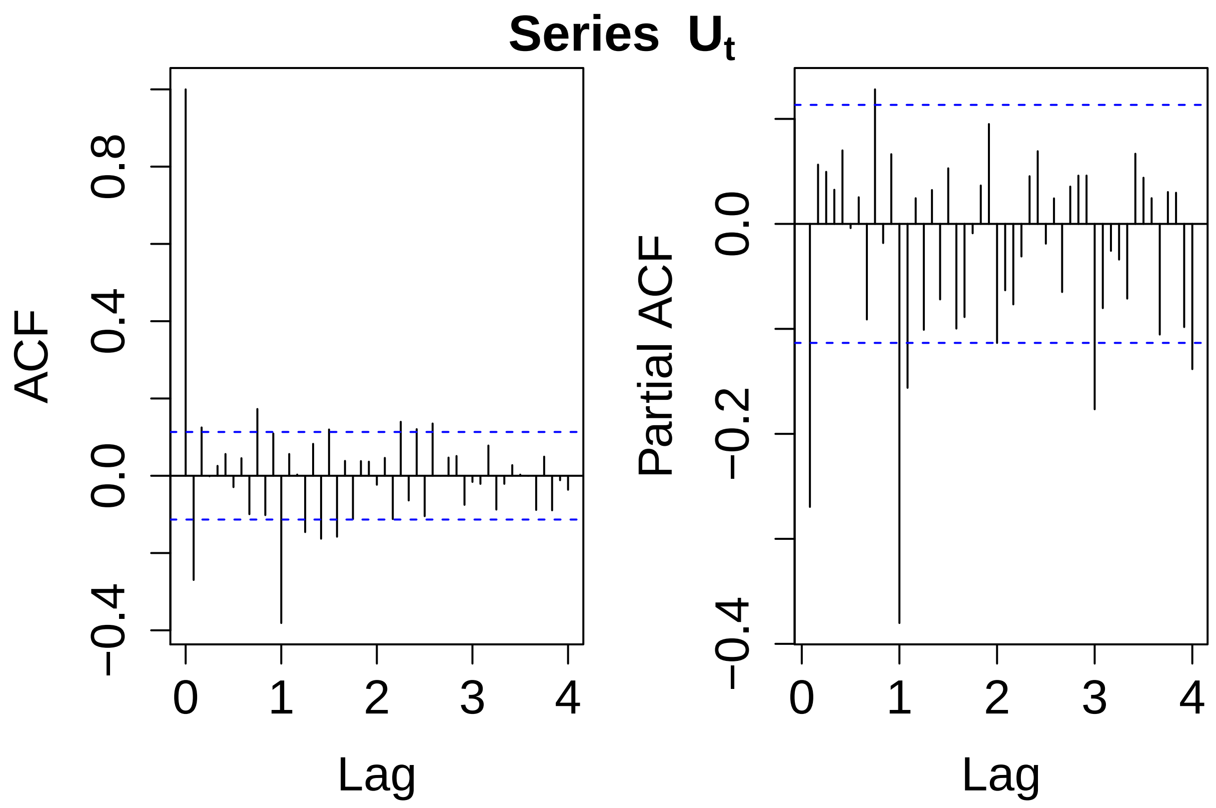
```
ts_dat_12 <- diff(f_ts_dat, 12)
kpss.test(ts_dat_12) #Big enough to call stationary
```

```
## Warning in kpss.test(ts_dat_12): p-value greater than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  ts_dat_12
## KPSS Level = 0.026132, Truncation lag parameter = 5, p-value = 0.1
```

```
#acf2(ts_dat_12)

#jpeg("seasonal_acf.jpg", width = 990, height = 350)
par(cex.lab = 1.5, cex.main = 1.6, cex.axis = 1.5, mar = c(4.5, 4.5, 3, 1), mfrow=c(1,2))
acf(ts_dat_12, lag.max = 48, main = "")
pacf(ts_dat_12, lag.max = 48, main = "")
mtext(expression(bold("Series " ~ U[t])), line = -3, outer = TRUE, cex = 1.6)
```

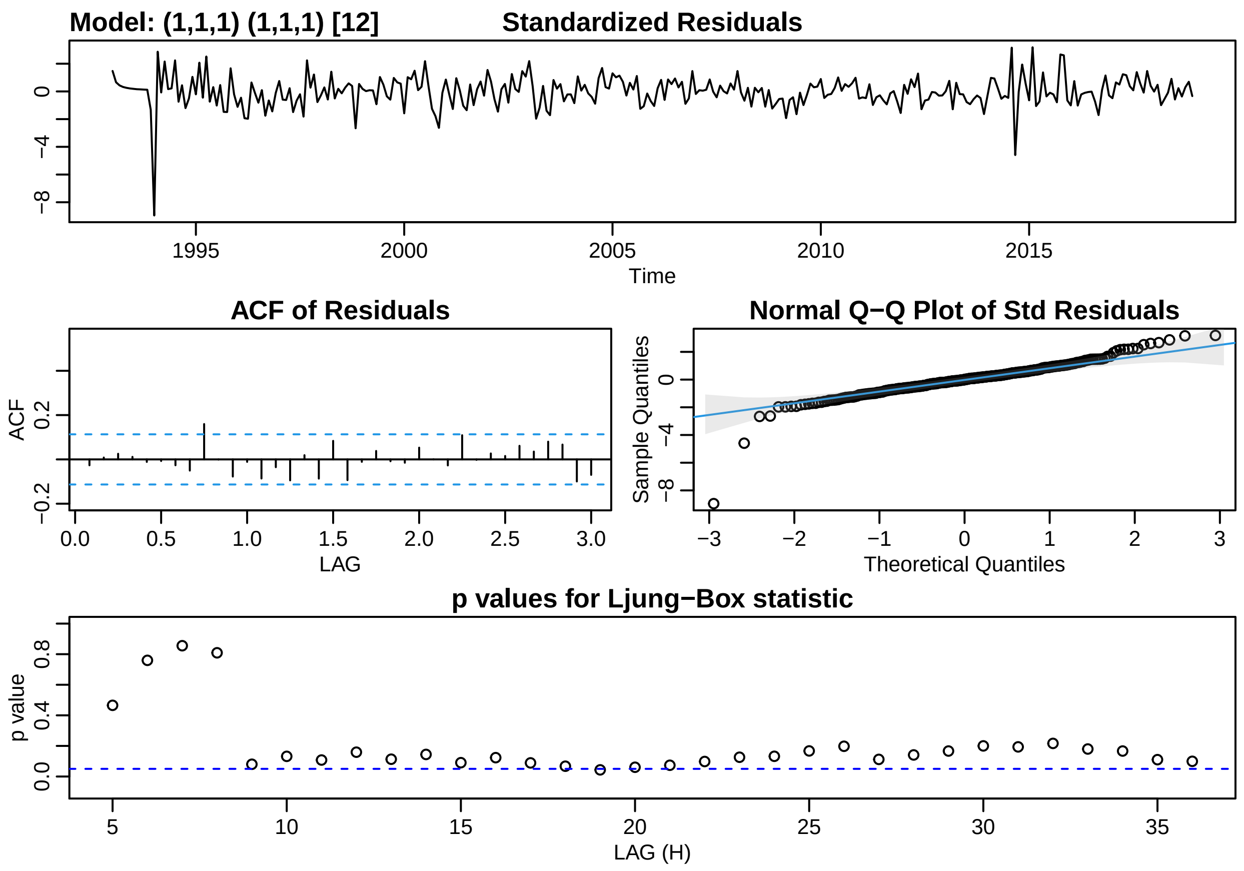


```
#dev.off()
```

At the seasonal level, these indicate a cutoff at 1 in the ACF and tailing off in PACF, possibly indicating P=0 and Q=1. Lags 1,2,...,11 suggest several choices, so estimates of $0 \leq p \leq 1$ and $0 \leq q \leq 1$ are made and explored.

```r
sarima(log_ts_dat, p = 1, d = 1, q = 1, P = 1, D = 1, Q = 1, S = 12) #AICc -8.123946
```

```
## initial  value -5.492400
## iter   2 value -5.587250
## iter   3 value -5.645844
## iter   4 value -5.649114
## iter   5 value -5.653614
## iter   6 value -5.658430
## iter   7 value -5.659765
## iter   8 value -5.660295
## iter   9 value -5.660350
## iter  10 value -5.660404
## iter  11 value -5.660436
## iter  12 value -5.660548
## iter  13 value -5.660569
## iter  14 value -5.660599
## iter  15 value -5.660613
## iter  16 value -5.660616
## iter  16 value -5.660616
## iter  16 value -5.660616
## final  value -5.660616
## converged
## initial  value -5.643060
## iter   2 value -5.644716
## iter   3 value -5.646227
## iter   4 value -5.647175
## iter   5 value -5.647271
## iter   6 value -5.647276
## iter   7 value -5.647279
## iter   8 value -5.647283
## iter   9 value -5.647288
## iter  10 value -5.647290
## iter  10 value -5.647290
## final  value -5.647290
## converged
```

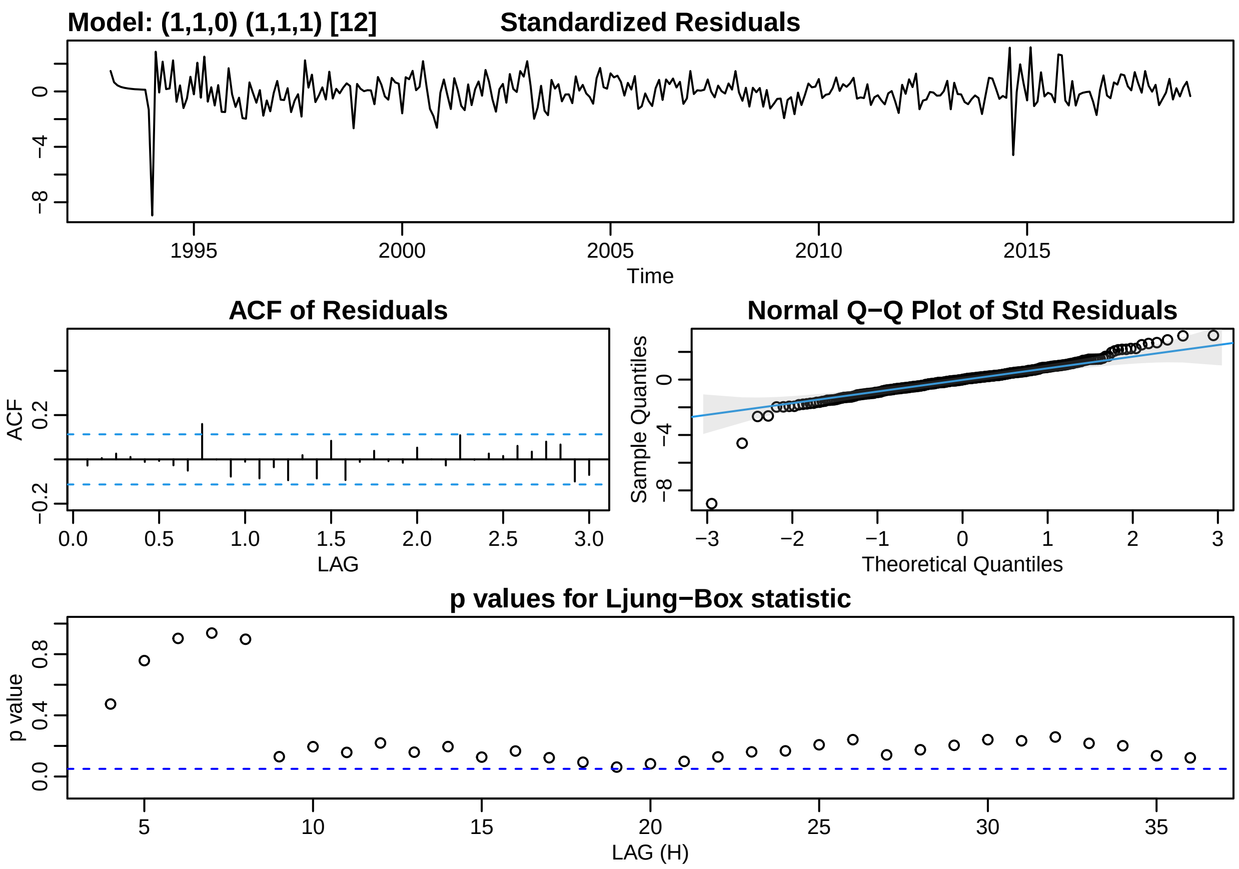


```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1      ma1    sar1     sma1
##       -0.3001  -0.0114  0.1042  -0.6954
## s.e.   0.1534   0.1573  0.0937   0.0725
##
## sigma^2 estimated as 1.218e-05:  log likelihood = 1264.28,  aic = -2518.55
##
## $degrees_of_freedom
## [1] 295
##
## $ttable
##      Estimate     SE t.value p.value
## ar1   -0.3001 0.1534 -1.9568  0.0513
## ma1   -0.0114 0.1573 -0.0725  0.9423
## sar1   0.1042 0.0937  1.1118  0.2671
## sma1  -0.6954 0.0725 -9.5862  0.0000
##
## $AIC
## [1] -8.124369
```

```
##
## $AICc
## [1] -8.123946
##
## $BIC
## [1] -8.064684
```

```
# ttable says ma1 coeff has highest p-value. removing this (model trimming):
sarima(log_ts_dat, p = 1, d = 1, q = 0, P = 1, D = 1, Q = 1, S = 12) #AICc -8.13055
```

```
## initial  value -5.492400
## iter   2 value -5.615705
## iter   3 value -5.649834
## iter   4 value -5.652797
## iter   5 value -5.660080
## iter   6 value -5.660508
## iter   7 value -5.660561
## iter   8 value -5.660570
## iter   8 value -5.660571
## final  value -5.660571
## converged
## initial  value -5.643116
## iter   2 value -5.645066
## iter   3 value -5.646830
## iter   4 value -5.647172
## iter   5 value -5.647260
## iter   6 value -5.647281
## iter   7 value -5.647281
## iter   7 value -5.647281
## iter   7 value -5.647281
## final  value -5.647281
## converged
```

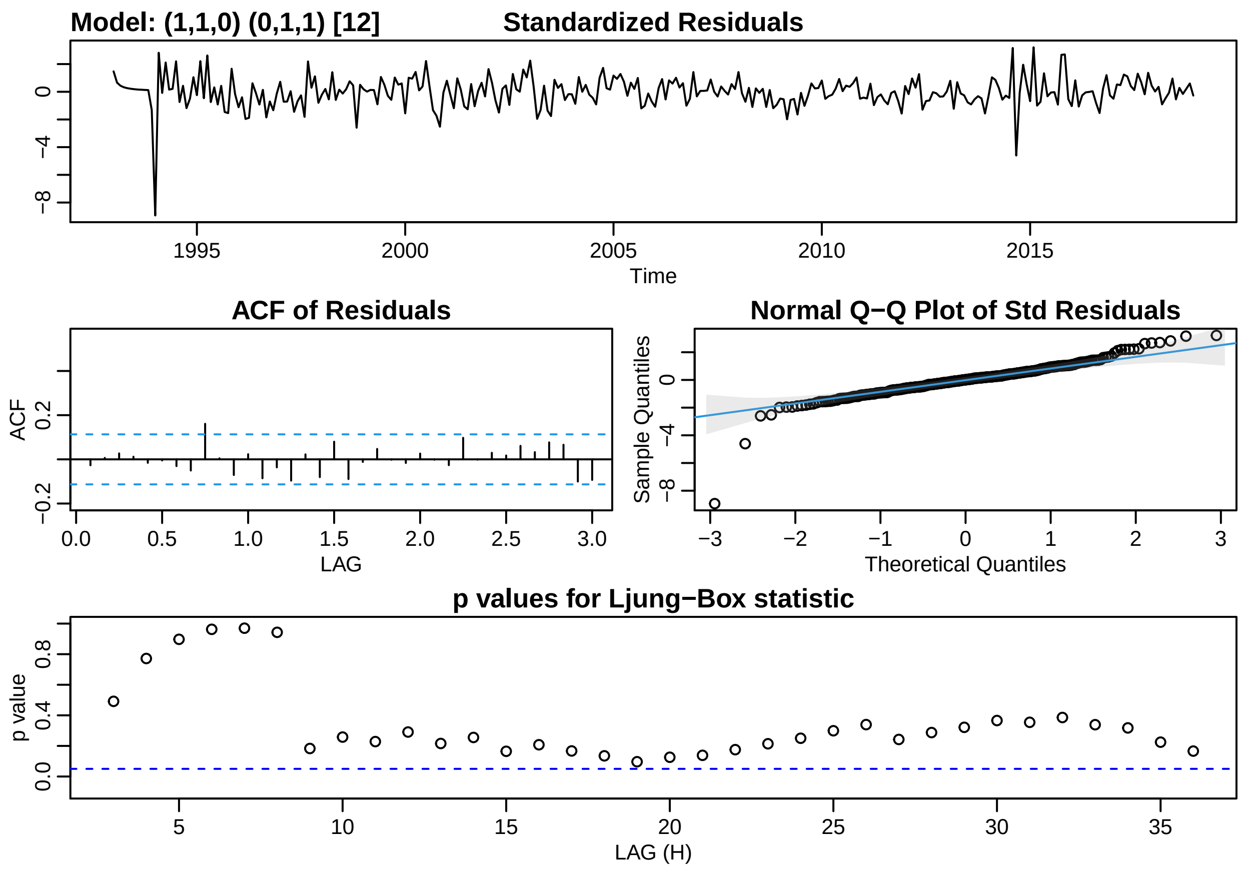


```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1    sar1     sma1
##       -0.3101  0.1030  -0.6945
## s.e.   0.0563  0.0937   0.0721
##
## sigma^2 estimated as 1.218e-05:  log likelihood = 1264.27,  aic = -2520.55
##
## $degrees_of_freedom
## [1] 296
##
## $ttable
##      Estimate     SE t.value p.value
## ar1   -0.3101 0.0563 -5.5107  0.0000
## sar1   0.1030 0.0937  1.0991  0.2726
## sma1  -0.6945 0.0721 -9.6366  0.0000
##
## $AIC
## [1] -8.130803
##
```

```
## $AICc
## [1] -8.13055
##
## $BIC
## [1] -8.083055
```

```
# ttable says sar1 coeff has highest p-value. removing this:
sarima(log_ts_dat, p = 1, d = 1, q = 0, P = 0, D = 1, Q = 1, S = 12) #AICc -8.133259
```

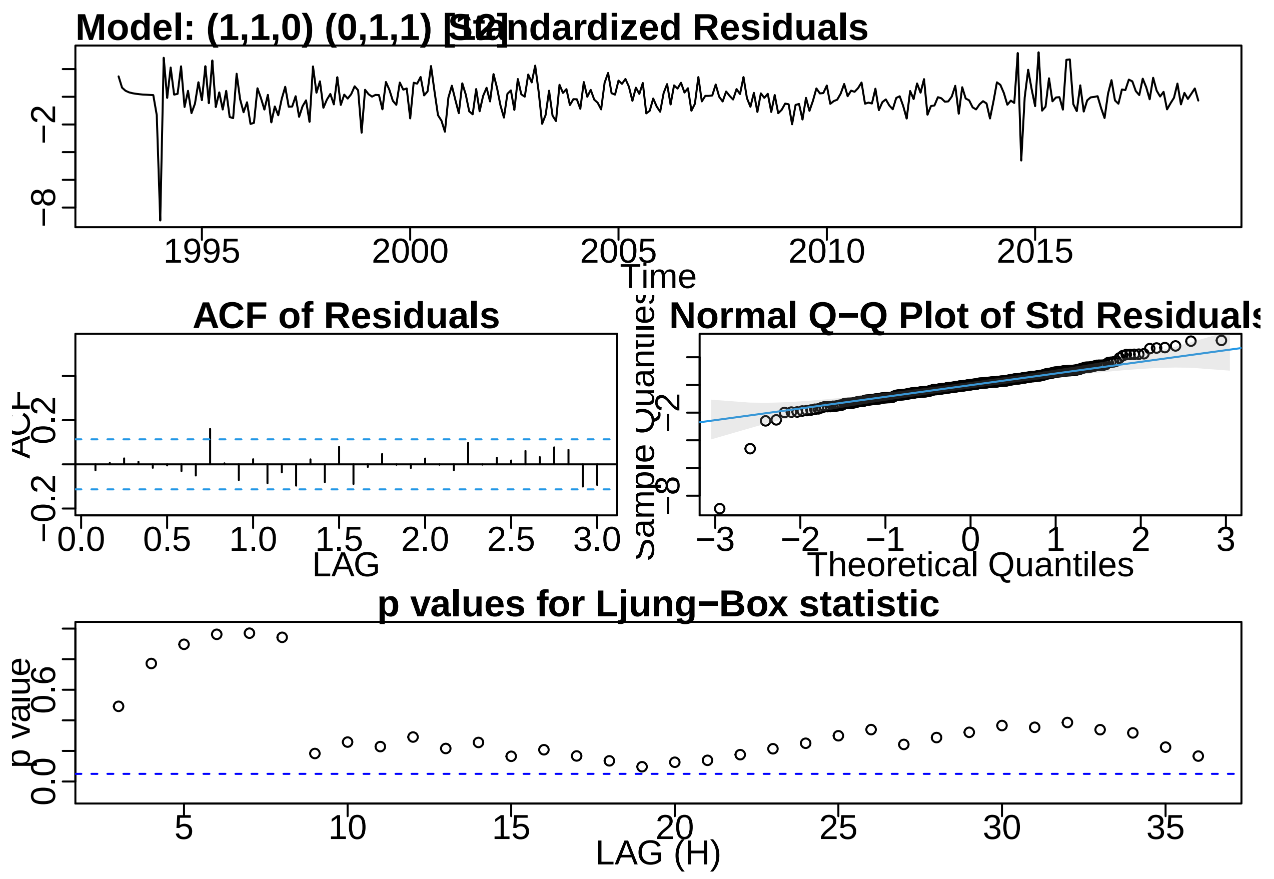
```
## initial  value -5.482214
## iter   2 value -5.637939
## iter   3 value -5.652411
## iter   4 value -5.656510
## iter   5 value -5.657397
## iter   6 value -5.657437
## iter   7 value -5.657437
## iter   7 value -5.657437
## iter   7 value -5.657437
## final  value -5.657437
## converged
## initial  value -5.644718
## iter   2 value -5.645257
## iter   3 value -5.645275
## iter   4 value -5.645275
## iter   4 value -5.645275
## iter   4 value -5.645275
## final  value -5.645275
## converged
```



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1     sma1
##       -0.3076  -0.6295
## s.e.   0.0563   0.0521
##
## sigma^2 estimated as 1.224e-05:  log likelihood = 1263.67,  aic = -2521.35
##
## $degrees_of_freedom
## [1] 297
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1   -0.3076 0.0563  -5.4671       0
## sma1  -0.6295 0.0521 -12.0747       0
##
## $AIC
## [1] -8.133385
##
## $AICc
## [1] -8.133259
##
## $BIC
## [1] -8.097574
```

```
#jpeg("diagnostic_plots.jpg", width = 1150, height = 700)
par(cex.lab = 1.6,  cex.main = 1.8, cex.axis = 1.6)
sarima(log_ts_dat, p = 1, d = 1, q = 0, P = 0, D = 1, Q = 1, S = 12)
```

```
## initial  value -5.482214
## iter   2 value -5.637939
## iter   3 value -5.652411
## iter   4 value -5.656510
## iter   5 value -5.657397
## iter   6 value -5.657437
## iter   7 value -5.657437
## iter   7 value -5.657437
## iter   7 value -5.657437
## final  value -5.657437
## converged
## initial  value -5.644718
## iter   2 value -5.645257
## iter   3 value -5.645275
## iter   4 value -5.645275
## iter   4 value -5.645275
## iter   4 value -5.645275
## final  value -5.645275
## converged
```



```
## $fit
##
## Call:
## stats::arima(x = xdata, order = c(p, d, q), seasonal = list(order = c(P, D,
##     Q), period = S), include.mean = !no.constant, transform.pars = trans, fixed = fixed,
##     optim.control = list(trace = trc, REPORT = 1, reltol = tol))
##
## Coefficients:
##           ar1     sma1
##       -0.3076  -0.6295
## s.e.   0.0563   0.0521
##
## sigma^2 estimated as 1.224e-05:  log likelihood = 1263.67,  aic = -2521.35
##
## $degrees_of_freedom
## [1] 297
##
## $ttable
##      Estimate     SE  t.value p.value
## ar1   -0.3076 0.0563  -5.4671       0
## sma1  -0.6295 0.0521 -12.0747       0
##
## $AIC
## [1] -8.133385
##
## $AICc
## [1] -8.133259
##
## $BIC
## [1] -8.097574
```

```
#dev.off()
```

For the first model, the Ljung-Box statistic is not satisfactory at lag 20, and unnesssecary coefficients were present in the model. By trimming coefficients and finding the minimum bias-corrected AIC, the model selected was $SARIMA(1,1,0)(0,1,1)_{12}$.

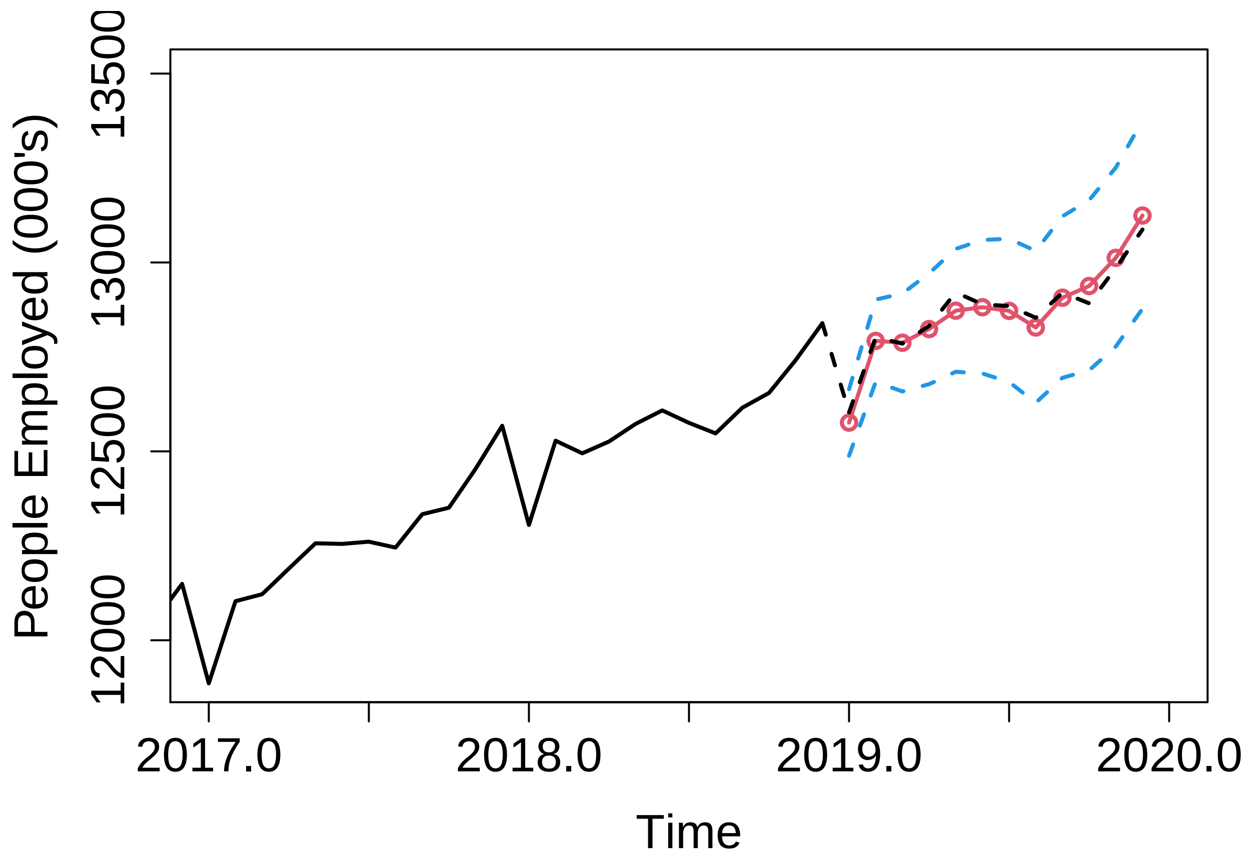
We see a couple of outliers in the standardised residuals and Q-Q plot. The ACF of the residuals is as expected for normal data. The Ljung-Box statistic has a large enough p-value at lag 20 large enough to not reject the null hypothesis of the model exhibiting lack of fit.

Fit the model from above with lowest AIC

```r
fit <- arima(log_ts_dat, c(1,1,0), seasonal = list(order = c(0,1,1), period = 12))
fore <- predict(fit, n.ahead = 12)
```

Display predictions:

```r
#jpeg("predictions_plots.jpg", width = 1118, height = 300)
par(cex.lab = 1.5, cex.axis = 1.5, cex.main = 1.6, mar = c(4.5, 4.5, 1, 1))
ts.plot(cbind(ts_dat, exp(fore$pred)), lwd = c(2, 2), col = c(1, 2), xlim = c(2017, 2020),
        ylab = "People Employed (000's)", ylim = c(11900, 13500))
lines(exp(fore$pred), type = "p", col = 2, lwd = 2)
lines(ts_join, lty = "dashed", lwd = 2)
lines(test_ts, lty = "dashed", lwd = 2)
# 95% confidence boundaries
lines(exp(fore$pred+2*fore$se), lty="dashed", col = 4, lwd = 2)
lines(exp(fore$pred-2*fore$se), lty="dashed", col = 4, lwd = 2)
```



```r
#dev.off()
```

Assess the predictions

```r
MAPE = 0
for (i in 1:12){
  MAPE = MAPE + abs((test_ts[i] - exp(fore$pred[i])) / test_ts[i])
}
MAPE = MAPE/12*100
MAPE
```

```
## [1] 0.1713158
```

```r
Acc = 1-MAPE
Acc
```

```
## [1] 0.8286842
```

MAPE can be converted to accuracy by 1-MAPE. So we have a MAPE or 0.17% or an accuracy of prediction, or 99.837% accuracy for the 12 months' forcecast.

Considering the possibility of adding a secondary time series

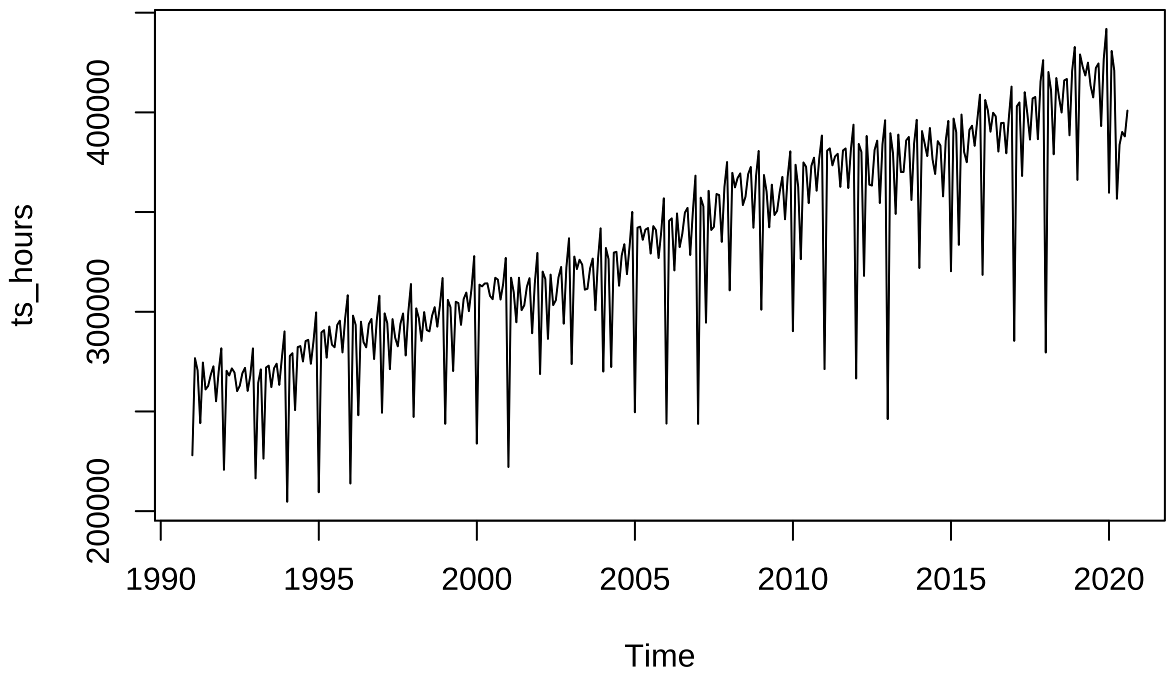
Load in the additional data:

```r
dat2 <- read.csv("hours_data.csv", fileEncoding = 'UTF-8-BOM')
head(dat2)
```

```
##   Observation.times Time.series.values
## 1            Jan-91           228022.4
## 2            Feb-91           276727.5
## 3            Mar-91           270555.0
## 4            Apr-91           244135.2
## 5            May-91           274548.9
## 6            Jun-91           261039.1
```

Create a time series object from the data and plot.

```r
ts_hours <- ts(dat2[, 2], start = c(1991, 1), end = c(2020, 8), frequency = 12)
plot.ts(ts_hours)
```

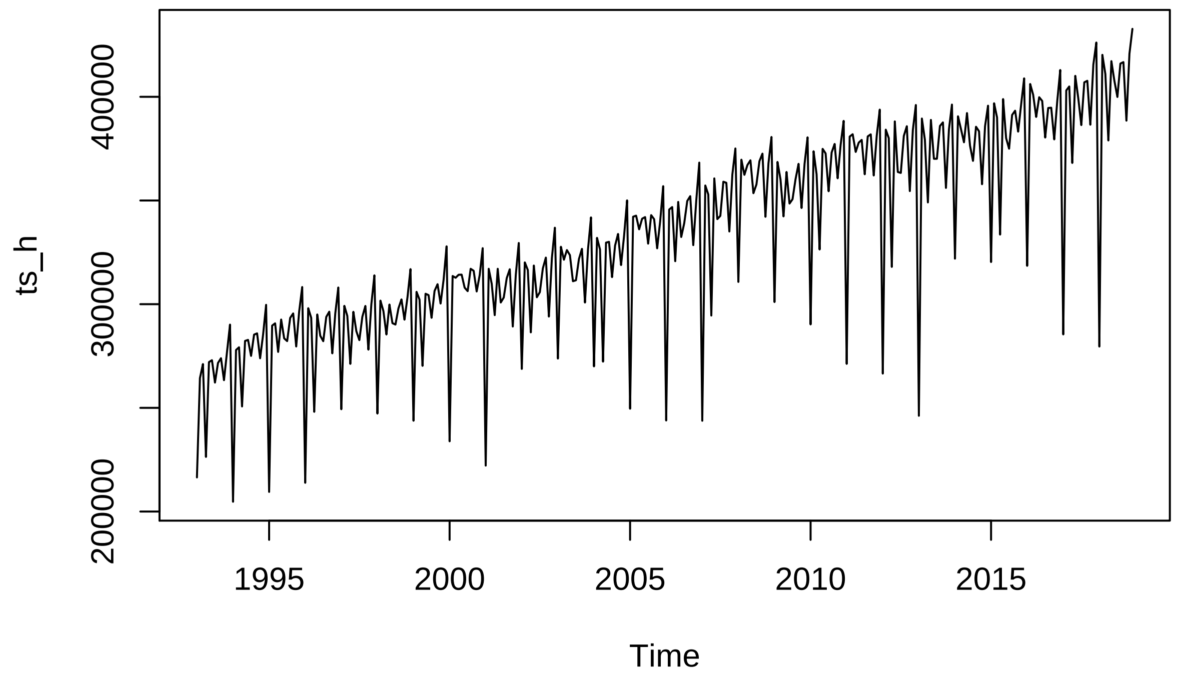


Cut the ts down to subset in appropriate time:

```
train_hours <- dat2[25:336,] # for training data
head(train_hours)
```

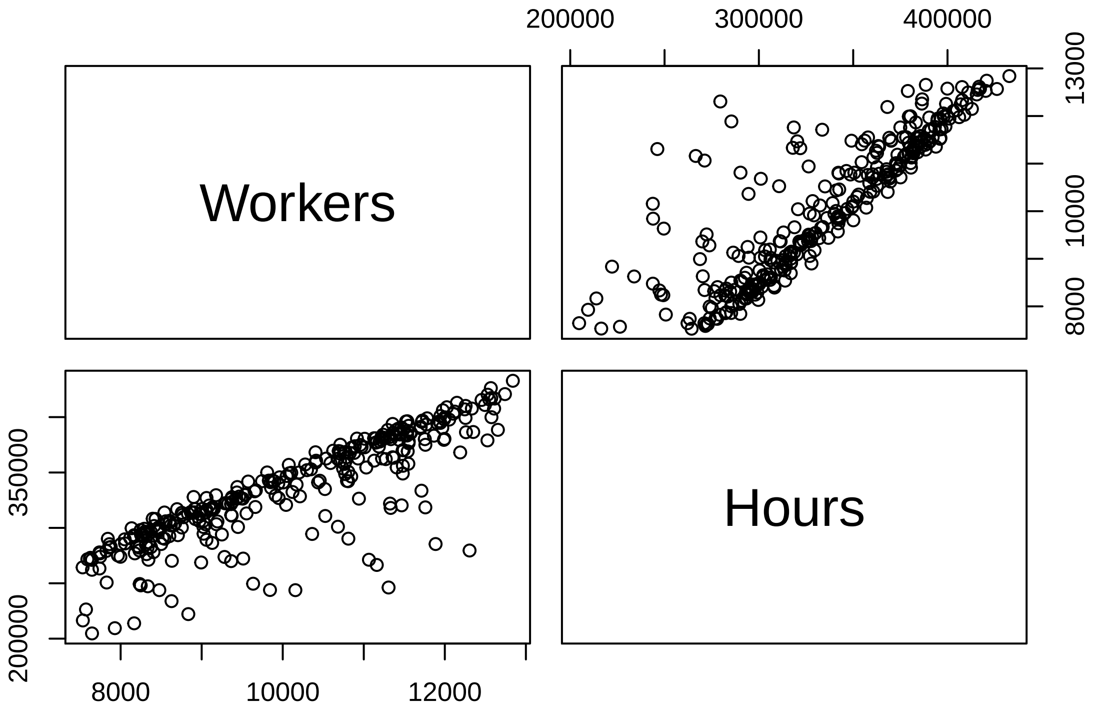
```
##    Observation.times Time.series.values
## 25            Jan-93           216445.1
## 26            Feb-93           264350.3
## 27            Mar-93           271123.6
## 28            Apr-93           226351.2
## 29            May-93           272067.0
## 30            Jun-93           272984.9
```

```
ts_h <- ts(train_hours[, 2], start = c(1993, 1), end = c(2018, 12), frequency = 12) # ts compare to mod
plot.ts(ts_h)
```



Plot a scatter plot comparing employed data and hours for those months.

```r
pairs(cbind(Workers=ts_dat, Hours=ts_h))
```


Workers
Hours
200000
300000
400000
8000
10000
12000
13000
350000