Time Series Analysis - Lab 02 (Group 7)

Anubhav Dikshit (anudi287) and Maximilian Pfundstein (maxpf364)

2019-09-23

Contents

1 Assignment 1: Computations with simulated data

1.1 Linear Regressions on Necessarily Lagged Variables and Appropriate Correlation

Task: Generate 1000 observations from AR(3) process with $\phi_1 = 0.8, \phi_2 = -0.2, \phi_3 = 0.1$. Use these data and the definition of PACF to compute ϕ_{33} from the sample, i.e. write your own code that performs linear regressions on necessarily lagged variables and then computes an appropriate correlation. Compare the result with the output of function `pacf()` and with the theoretical value of ϕ_{33}.

$\phi_{33} = corr(X_{t-3} - f_p, X_t - f_p)$ where $f_p = \sum_{j=1}^{p} \phi_j X_{\tau-j}$

```
set.seed(12345)
x_t <- arima.sim(model = list(ar = c(0.8,-0.2,0.1)), n=1000)
actual_pacf_value <- pacf(x_t, plot = FALSE)$acf[3]
df <- data.frame(x_t = as.vector(x_t))
df$x_t_lag_1 <- lag(df$x_t,1)
df$x_t_lag_2 <- lag(df$x_t,2)
```

```r
df$x_t_lag_3 <- lag(df$x_t,3)
df <- na.omit(df)

# building models and getting their residuals
model_1_res <- lm(x_t ~ x_t_lag_1 + x_t_lag_2, data = df)$residuals
model_2_res <- lm(x_t_lag_3 ~ x_t_lag_1 + x_t_lag_2, data = df)$residuals

# theortical pacf values
theotical_pacf_value <- cor(x = model_1_res, y = model_2_res, use = "na.or.complete")

cat("The theoretical and actual value of PACF are: ", theotical_pacf_value, actual_pacf_value)
```

```
## The theoretical and actual value of PACF are:  0.1146076 0.1170643
```

Analysis: The theoretical and the actual values of PACF are very similar.

1.2 Methods of Moments, Conditional Least Squares and Maximum Likelihood

Task: Simulate an AR(2) series with $\phi_1 = 0.8, \phi_2 = 0.1$ and $n = 100$. Compute the estimated parameters and their standard errors by using three methods: method of moments (Yule-Walker equations), conditional least squares and maximum likelihood (ML) and compare their results to the true values. Which method does seem to give the best result? Does theoretical value for ϕ_2 fall within confidence interval for ML estimate?

```r
set.seed(12345)
x_t <- arima.sim(model = list(ar = c(0.8,0.1)), n=100)

method_yule_walker <- ar(x_t, order = 2, method = "yule-walker", aic = FALSE)$ar
method_cls <- ar(x_t, order = 2, method = "ols", aic = FALSE)$ar
method_mle <- ar(x_t, order = 2, method = "mle", aic = FALSE)$ar

df <- data.frame(rbind(method_yule_walker, method_cls,method_mle))

kable(df, caption = "Comparison of parameters using different methods")
```

Table 1: Comparison of parameters using different methods

	ar1	ar2
method_yule_walker	0.8029146	0.1037053
method_cls	0.8066782	0.1205352
method_mle	0.7968774	0.1189369

```r
# Since varience is not given by ar we use arima function
ML_Model_CI = arima(x_t, order = c(2,0,0), method = "ML")
sigma = ML_Model_CI$var.coef[2, 2]
phi_2 = ML_Model_CI$coef[2]
CI = c(phi_2 - 1.96 * sigma, phi_2 + 1.96 * sigma)
CI
```

```
##        ar2        ar2
```

```
## 0.09924714 0.13846032
```

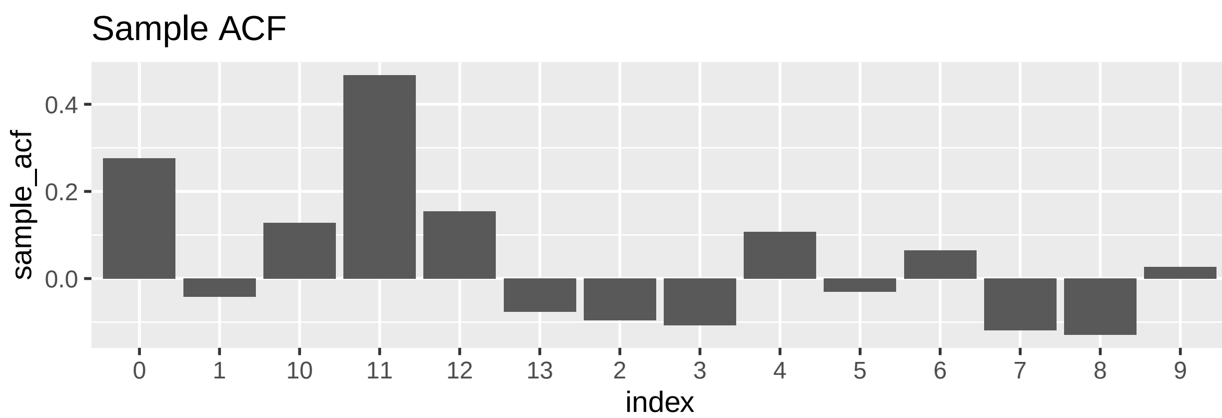
Analysis: The parameter values from yule walker method is the closet to the actual value of 0.8,0.1. Yes the theoretical value of ϕ_2 did fall within confidence interval using MLE method.

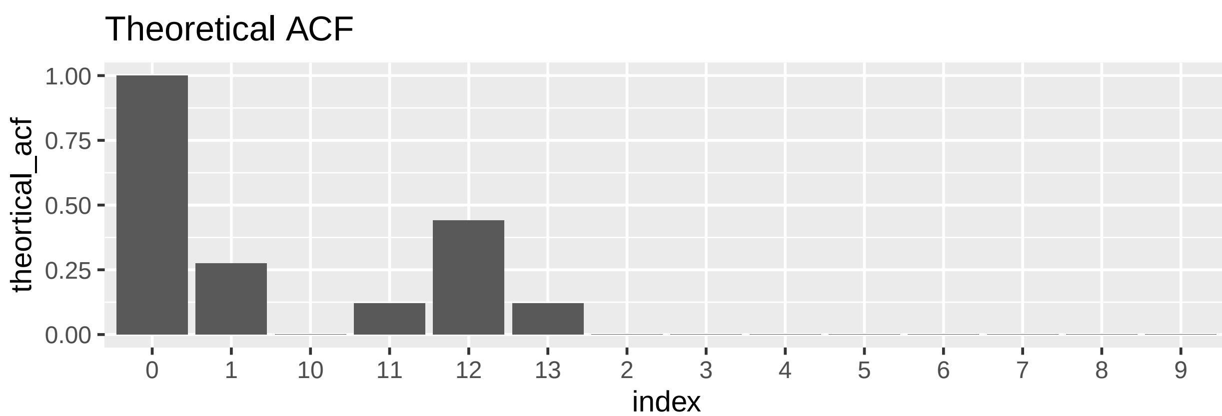
1.3 Sample and Theoretical ACF and PACF

Task: Generate 200 observations of a seasonal ARIMA$(0,0,1) \times (0,0,1)_{12}$ model with coefficients $\Theta = 0.6$ and $\theta = 0.3$ by using `arima.sim()`. Plot sample ACF and PACF and also theoretical ACF and PACF. Which patterns can you see at the theoretical ACF and PACF? Are they repeated at the sample ACF and PACF?

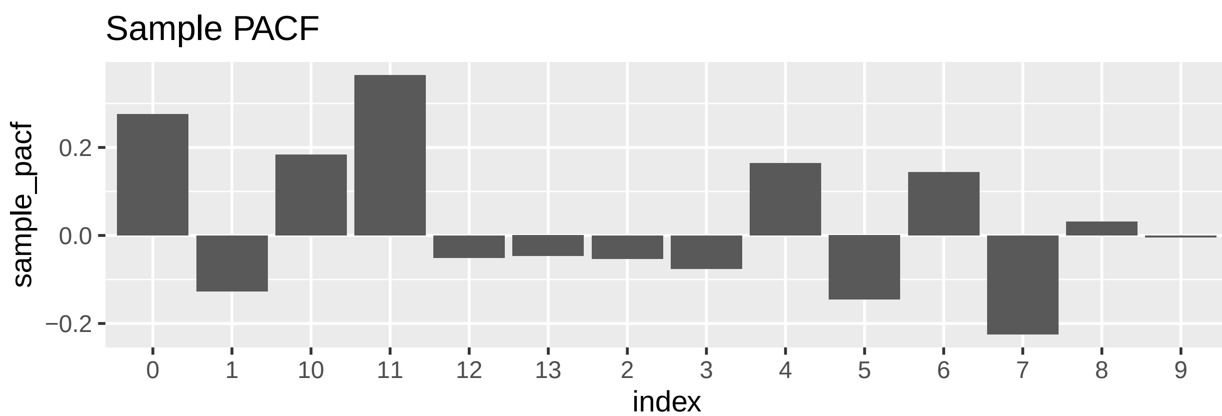
Now $ARIMA(1,1,1)(1,1,1)_4$ can be written as $(1-\phi_1 B)(1-B)(1-B^4)(1-\Phi_1 B^4)x_t = w_t(1+\theta B)(1+\Theta B^4)$ Similarly $ARIMA(0,0,1)(0,0,1)_{12}$ can be written as $x_t = w_t(1+\Theta B^{12})(1+\theta B)$ which can be simplified as $x_t = w_t(1+\Theta B^{12}+\theta B+\Theta\theta B^{13})$ given that $\theta = 0.3$ and $\Theta = 0.6$ we get $x_t = w_t(1+0.3B+0.6B^{12}+0.18B^{13})$

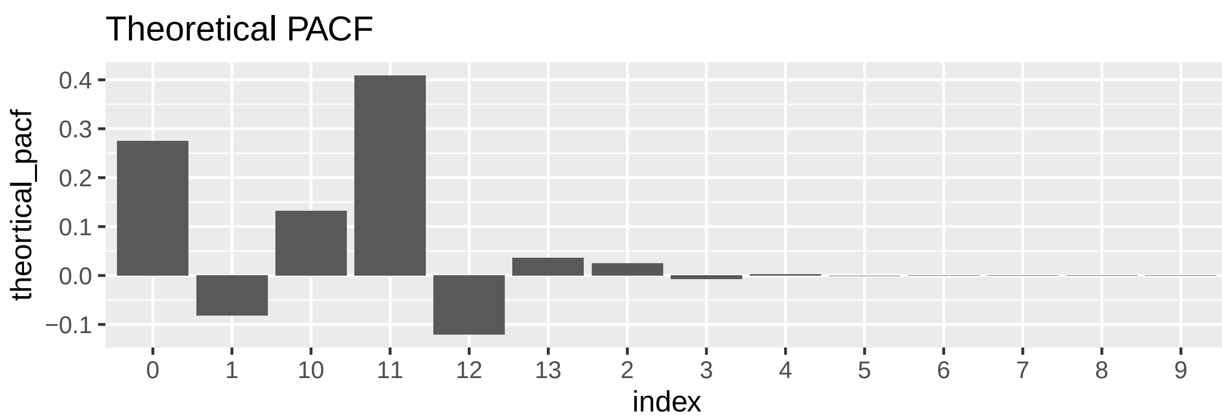
```
set.seed(12345)
x_t <- arima.sim(model = list(ma = c(0.3,rep(0,10),0.6,0.18)), n=200)

df <- data.frame(sample_acf = acf(x_t, plot = FALSE, lag.max = 14)$acf,
                 sample_pacf = pacf(x_t, plot = FALSE, lag.max = 14)$acf,
                 theortical_acf = ARMAacf(ma = c(0.3,rep(0,10),0.6,0.18), pacf = FALSE, lag.max = 13),
                 theortical_pacf = ARMAacf(ma = c(0.3,rep(0,10),0.6,0.18), pacf = TRUE, lag.max = 14))

df$index <- rownames(df)

plot1 <- ggplot(data=df, aes(x=index)) +
  geom_col(aes(y=sample_acf)) +
  ggtitle("Sample ACF")

plot2 <- ggplot(data=df, aes(x=index)) +
  geom_col(aes(y=theortical_acf)) +
  ggtitle("Theoretical ACF")

grid.arrange(plot1, plot2, ncol = 1)
```






```r
plot3 <- ggplot(data=df, aes(x=index)) +
  geom_col(aes(y=sample_pacf)) +
  ggtitle("Sample PACF")

plot4 <- ggplot(data=df, aes(x=index)) +
  geom_col(aes(y=theortical_pacf)) +
  ggtitle("Theoretical PACF")

grid.arrange(plot3, plot4, ncol = 1)
```





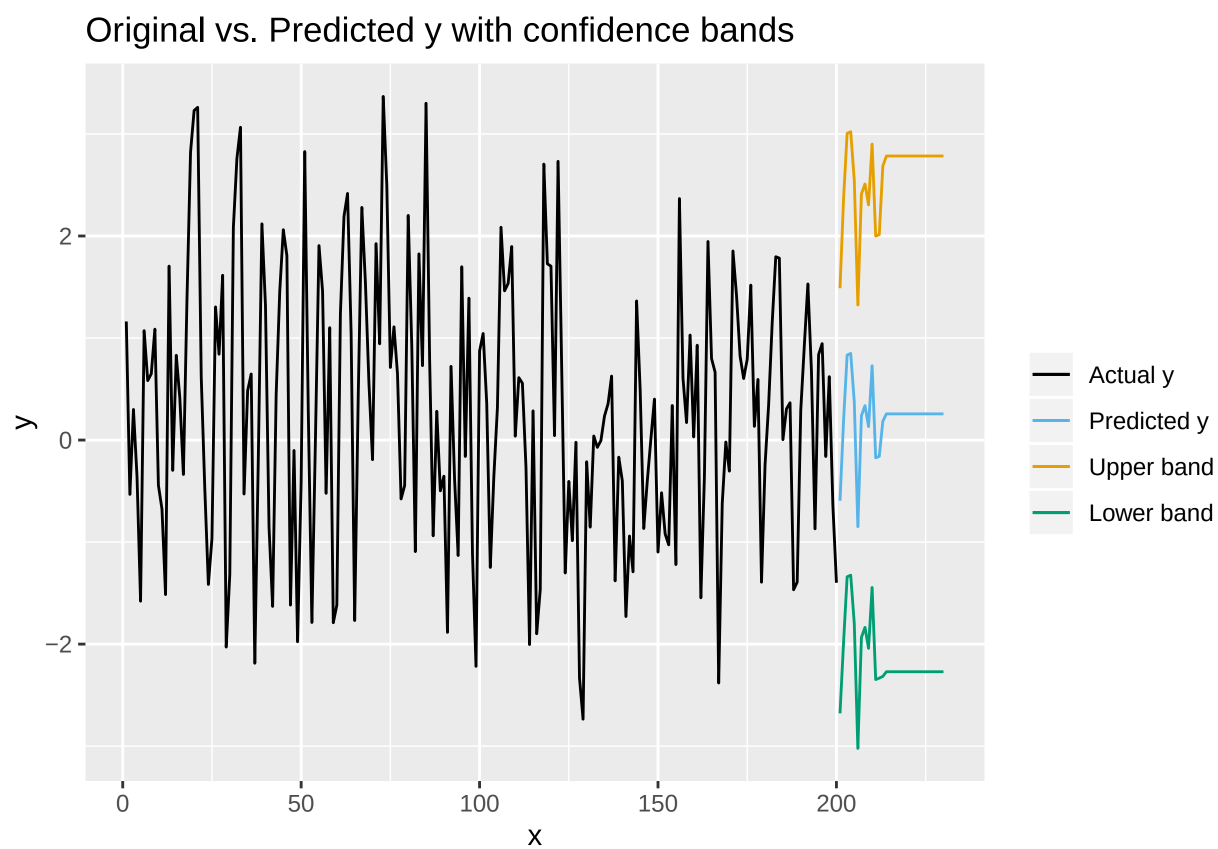
1.4 Forecast and Prediction

Task: Generate 200 observations of a seasonal ARIMA$(0,0,1) \times (0,0,1)_{12}$ model with coefficients $\Theta = 0.6$ and $\theta = 0.3$ by using `arima.sim()`. Fit ARIMA$(0,0,1) \times (0,0,1)_{12}$ model to the data, compute forecasts and a prediction band 30 points ahead and plot the original data and the forecast with the prediction band. Fit the same data with function `gausspr()` from package `kernlab` (use default settings). Plot the original data and predicted data from $t = 1$ to $t = 230$. Compare the two plots and make conclusions.

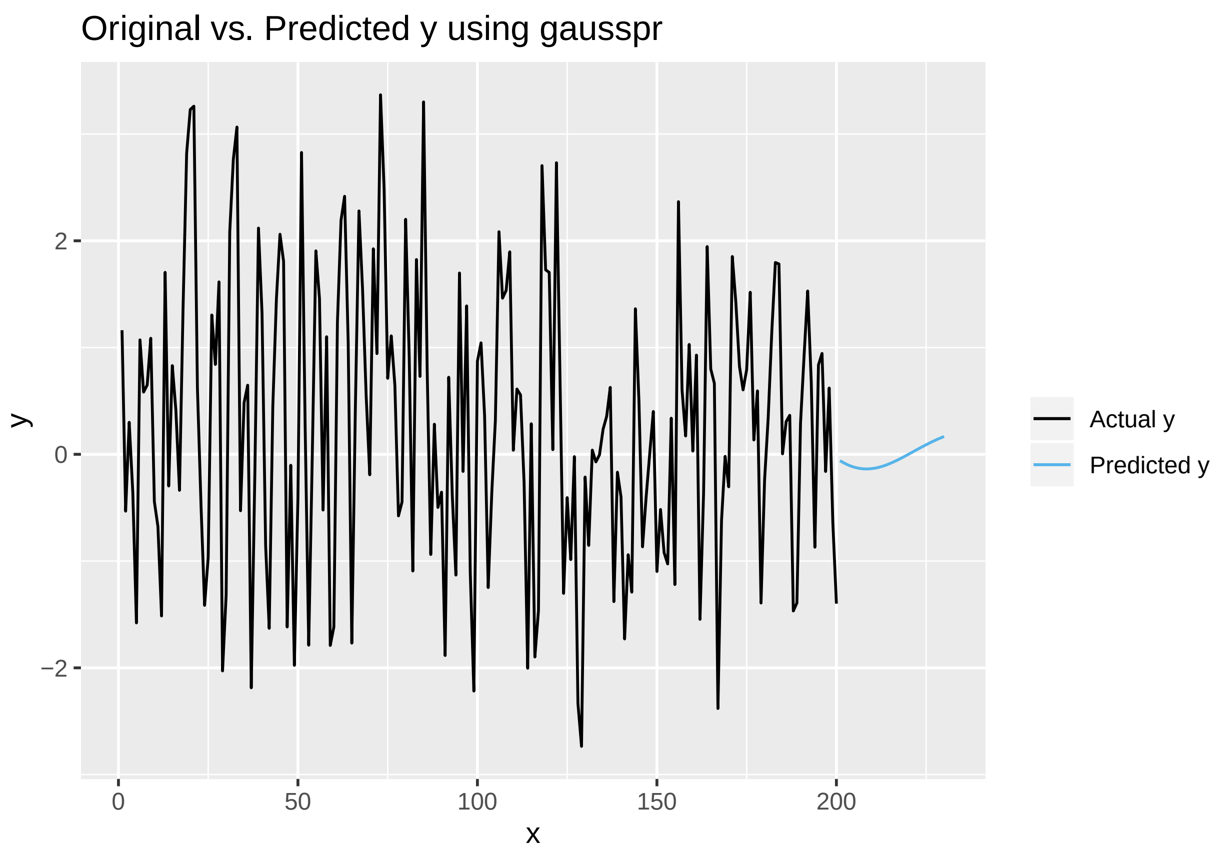
```
set.seed(12345)
x_t <- arima.sim(model = list(ma = c(0.3,rep(0,10),0.6,0.18)), n=200)
fit_x_t <- arima(x_t, order = c(0,0,1), seasonal = list(order = c(0,0,1),period = 12))
predicted_x_t <- predict(fit_x_t, n.ahead=30)
predicted_x_t_upper_band <- predicted_x_t$pred + 1.96 * predicted_x_t$se
predicted_x_t_lower_band <- predicted_x_t$pred - 1.96 * predicted_x_t$se

#kernlab

df <- data.frame(y = x_t)
df$x <- as.numeric(rownames(df))
gausspr_model <- gausspr(x=df$x, y=df$y)
```

```
## Using automatic sigma estimation (sigest) for RBF or laplace kernel
```

```
predicted_x_t_kernlab <- predict(gausspr_model, newdata=data.frame(x=201:230))

df3 <- data.frame(y = predicted_x_t_kernlab, x=201:230)


df2 <- data.frame(predicted_x_t = predicted_x_t$pred,
                  predicted_x_t_upper = predicted_x_t_upper_band,
                  predicted_x_t_lower = predicted_x_t_lower_band,
                  x = 201:230)

ggplot() +
  geom_line(data=df, aes(x=x, y=y, color="Actual y")) +
    geom_line(data=df2, aes(x=x, y=predicted_x_t, color="Predicted y")) +
      geom_line(data=df2, aes(x=x, y=predicted_x_t_upper, color="Upper band")) +
        geom_line(data=df2, aes(x=x, y=predicted_x_t_lower, color="Lower band")) +
      scale_colour_manual("", breaks = c("Actual y", "Predicted y", "Upper band", "Lower band"),
                        values = c("#000000", "#009E73", "#56B4E9", "#E69F00")) +
  ggtitle("Original vs. Predicted y with confidence bands")
```




```
ggplot() +
  geom_line(data=df, aes(x=x, y=y, color="Actual y")) +
    geom_line(data=df3, aes(x=x, y=y, color="Predicted y")) +
        scale_colour_manual("", breaks = c("Actual y", "Predicted y"),
                        values = c("#000000","#56B4E9")) +
  ggtitle("Original vs. Predicted y using gausspr")
```

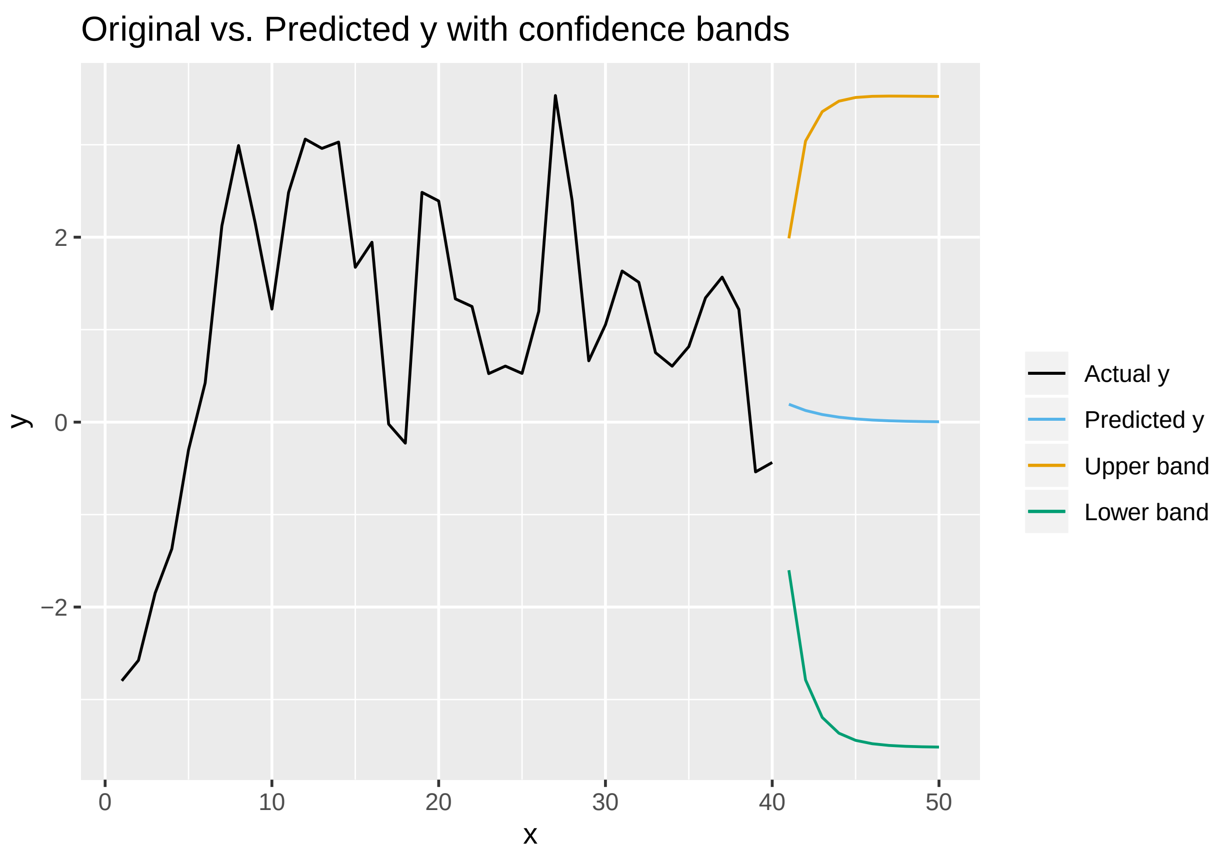


1.5 Prediction Band

Task: Generate 50 observations from ARMA(1, 1) process with $\phi = 0.7$, $\theta = 0.50$. Use first 40 values to fit an ARMA(1,1) model with $\mu = 0$. Plot the data, the 95% prediction band and plot also the true 10 values that you initially dropped. How many of them are outside the prediction band? How can this be interpreted?

```
x_t <- arima.sim(model = list(ar = c(0.7), ma=c(0.5)), n=50)
fit_x_t <- arima(x_t[1:40], order = c(1,0,1), include.mean = 0)

predicted_x_t <- predict(fit_x_t, n.ahead=10)
predicted_x_t_upper_band <- predicted_x_t$pred + 1.96 * predicted_x_t$se
predicted_x_t_lower_band <- predicted_x_t$pred - 1.96 * predicted_x_t$se

df <- data.frame(y = x_t[1:40], x=1:40)
df2 <- data.frame(y = predicted_x_t$pred,
                  upper_band=predicted_x_t_upper_band,
                  lower_band=predicted_x_t_lower_band,
                  x = 41:50)

ggplot() +
  geom_line(data=df, aes(x=x, y=y, color="Actual y")) +
    geom_line(data=df2, aes(x=x, y=y, color="Predicted y")) +
      geom_line(data=df2, aes(x=x, y=upper_band, color="Upper band")) +
        geom_line(data=df2, aes(x=x, y=lower_band, color="Lower band")) +
```

```
    scale_colour_manual("", breaks = c("Actual y", "Predicted y", "Upper band", "Lower band"),
                        values = c("#000000", "#009E73", "#56B4E9", "#E69F00")) +
  ggtitle("Original vs. Predicted y with confidence bands")
```



Analysis: All are inside the bands

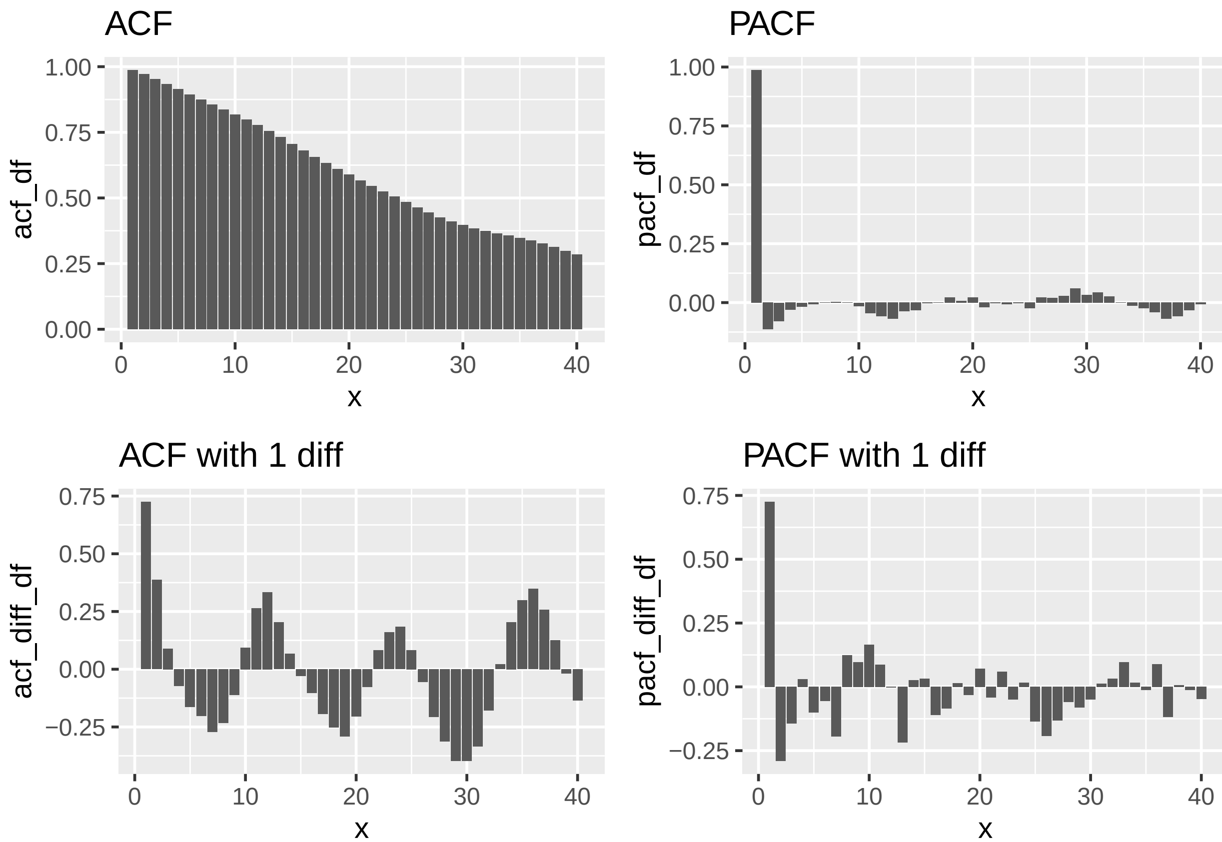
2 Assignment 2: ACF and PACF diagnostics

2.1 ARIMA Model Suggestion

Task: For data series `chicken` in package `astsa` (denote it by `x_t`) plot 4 following graphs up to 40 lags: ACF(x_t), PACF(x_t), ACF(∇x_t), PACF(∇x_t) (group them in one graph). Which ARIMA(p, d, q) or ARIMA$(p, d, q) \times (P, D, Q)_s$ models can be suggested based on this information only? Motivate your choice.

```
set.seed(12345)

plot_acf_pacf <- function(df){
acf_df <- acf(df, plot = FALSE, lag.max = 40)$acf
pacf_df <- pacf(df, plot = FALSE, lag.max = 40)$acf
acf_diff_df <- acf(diff(df), plot = FALSE, lag.max = 40)$acf
pacf_diff_df <- pacf(diff(df), plot = FALSE, lag.max = 40)$acf

df <- data.frame(acf_df=acf_df,
                 pacf_df=pacf_df,
```

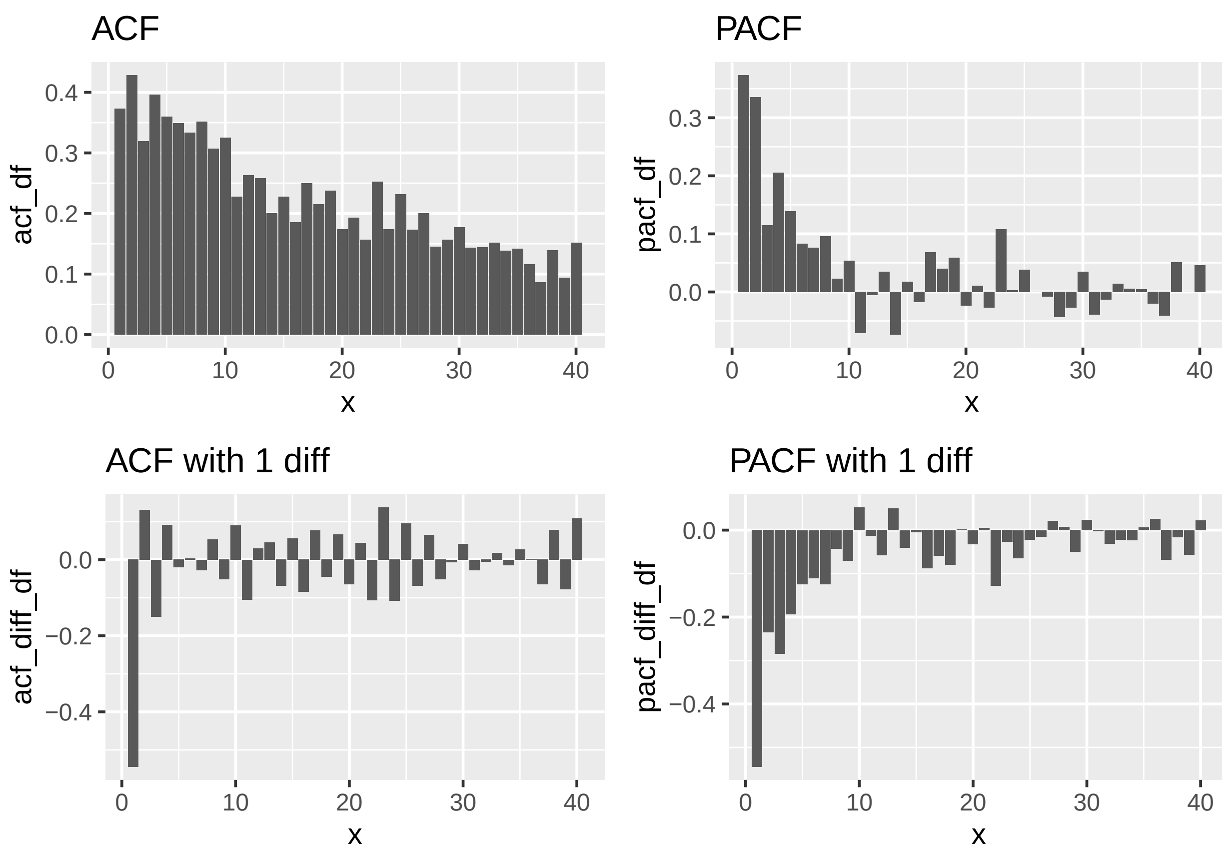
```r
                 acf_diff_df=acf_diff_df,
                 pacf_diff_df=pacf_diff_df,
                 x=1:length(pacf_diff_df))


plot1 <- ggplot(data=df, aes(x=x)) +
  geom_col(aes(y=acf_df)) +
  ggtitle("ACF")

plot2 <- ggplot(data=df, aes(x=x)) +
  geom_col(aes(y=pacf_df)) +
  ggtitle("PACF")

plot3 <- ggplot(data=df, aes(x=x)) +
  geom_col(aes(y=acf_diff_df)) +
  ggtitle("ACF with 1 diff")

plot4 <- ggplot(data=df, aes(x=x)) +
  geom_col(aes(y=pacf_diff_df)) +
  ggtitle("PACF with 1 diff")


return(grid.arrange(plot1, plot2, plot3, plot4,  nrow = 2,ncol = 2))
}

plot_acf_pacf(df=chicken)
```



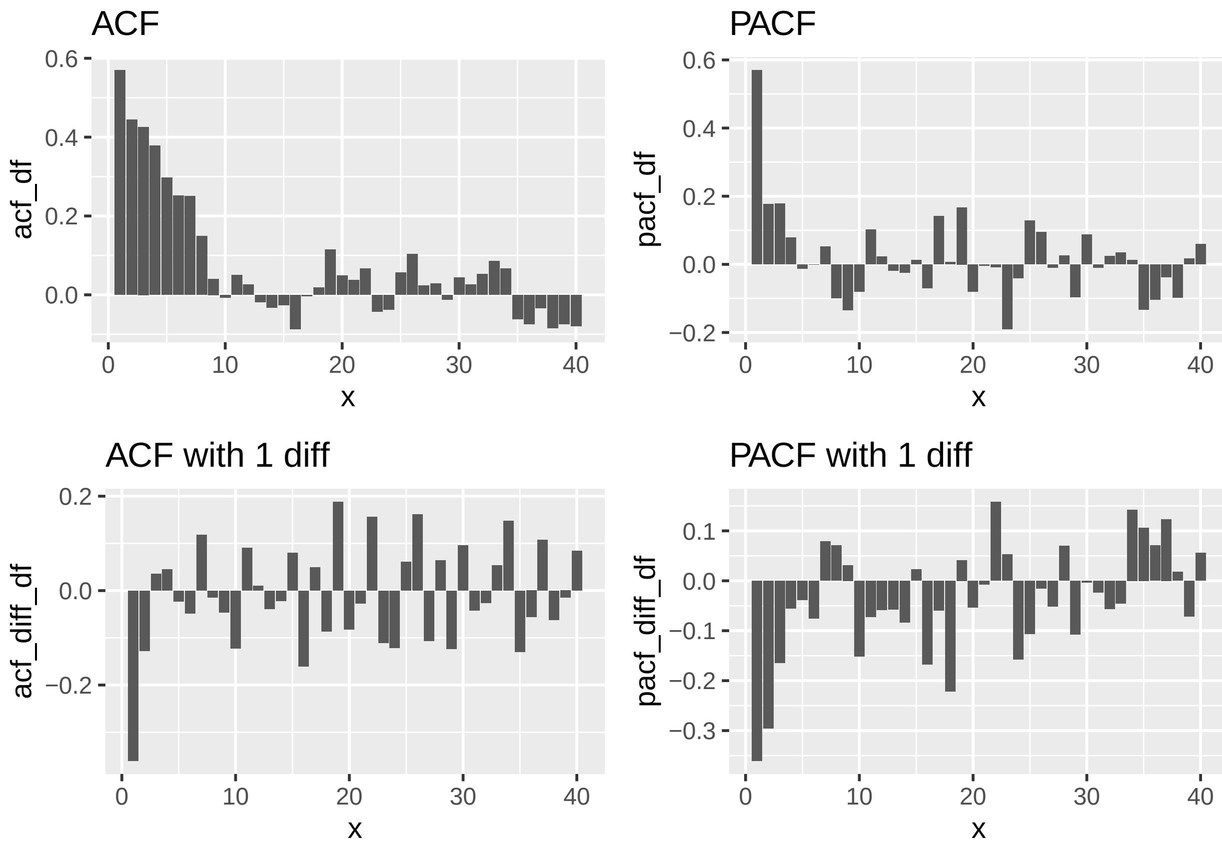
2.2 More Data sets

Task: Repeat step 1 for the following data sets: `so2`, `EQcount`, `HCT` in package `astsa`.

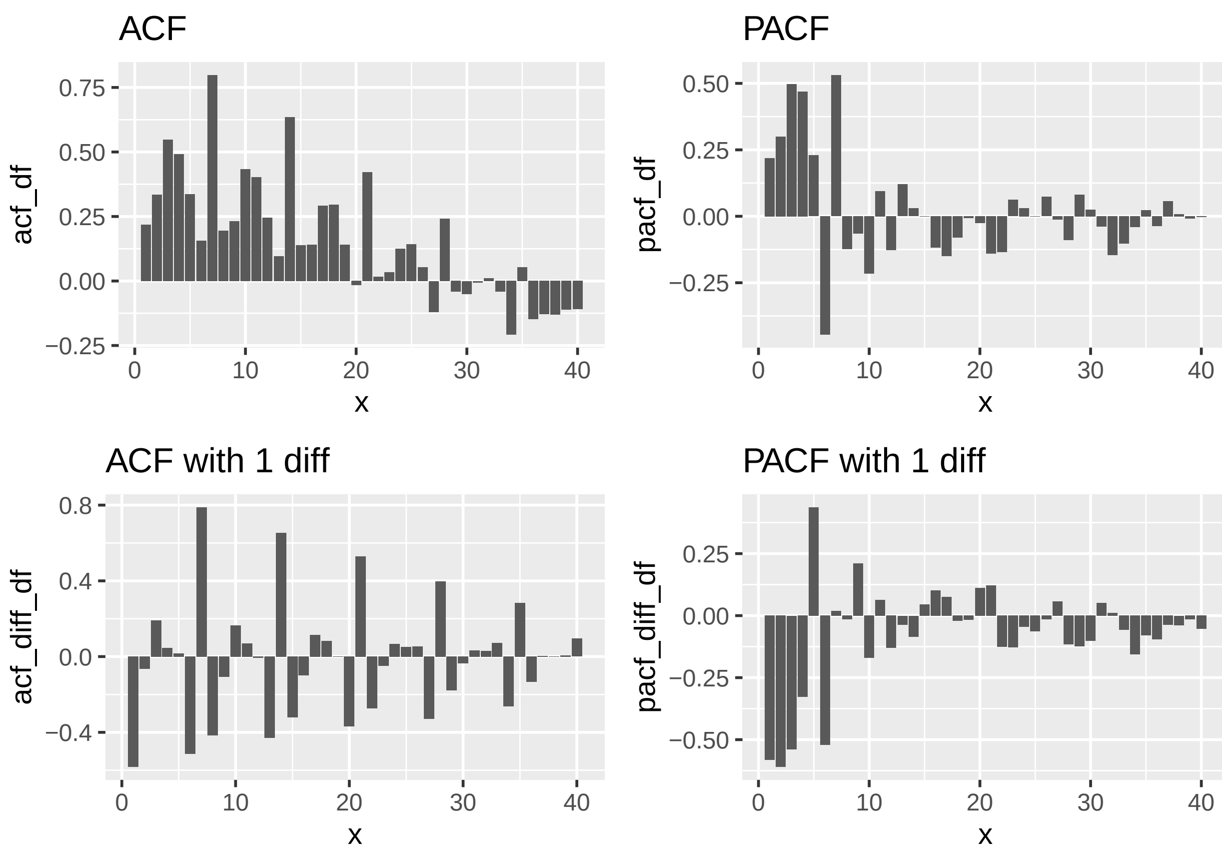
```
set.seed(12345)
plot_acf_pacf(df=so2)
```




```r
plot_acf_pacf(df=EQcount)
```




```
plot_acf_pacf(df=HCT)
```



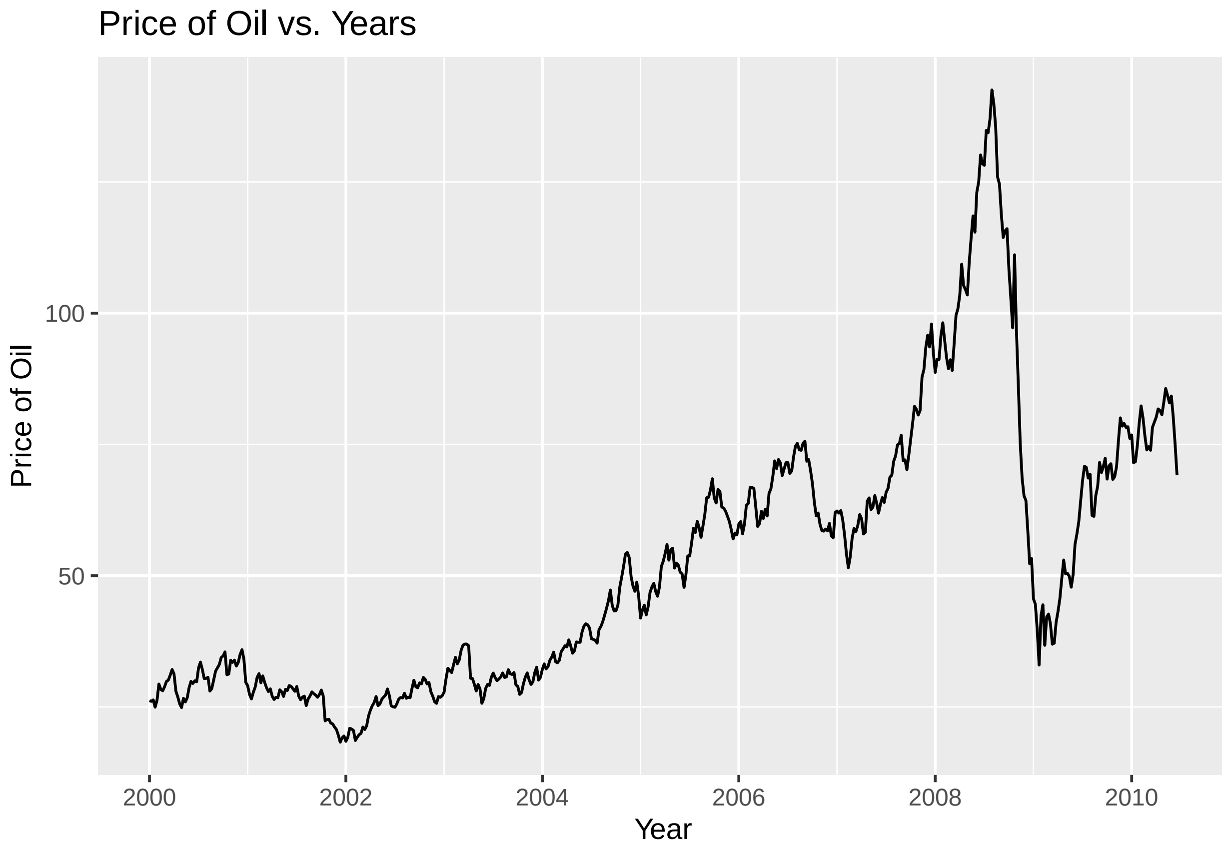
3 Assignment 3: ARIMA modeling cycle

In this assignment, you are assumed to apply a complete ARIMA modeling cycle starting from visualization and detrending and ending up with a forecasting.

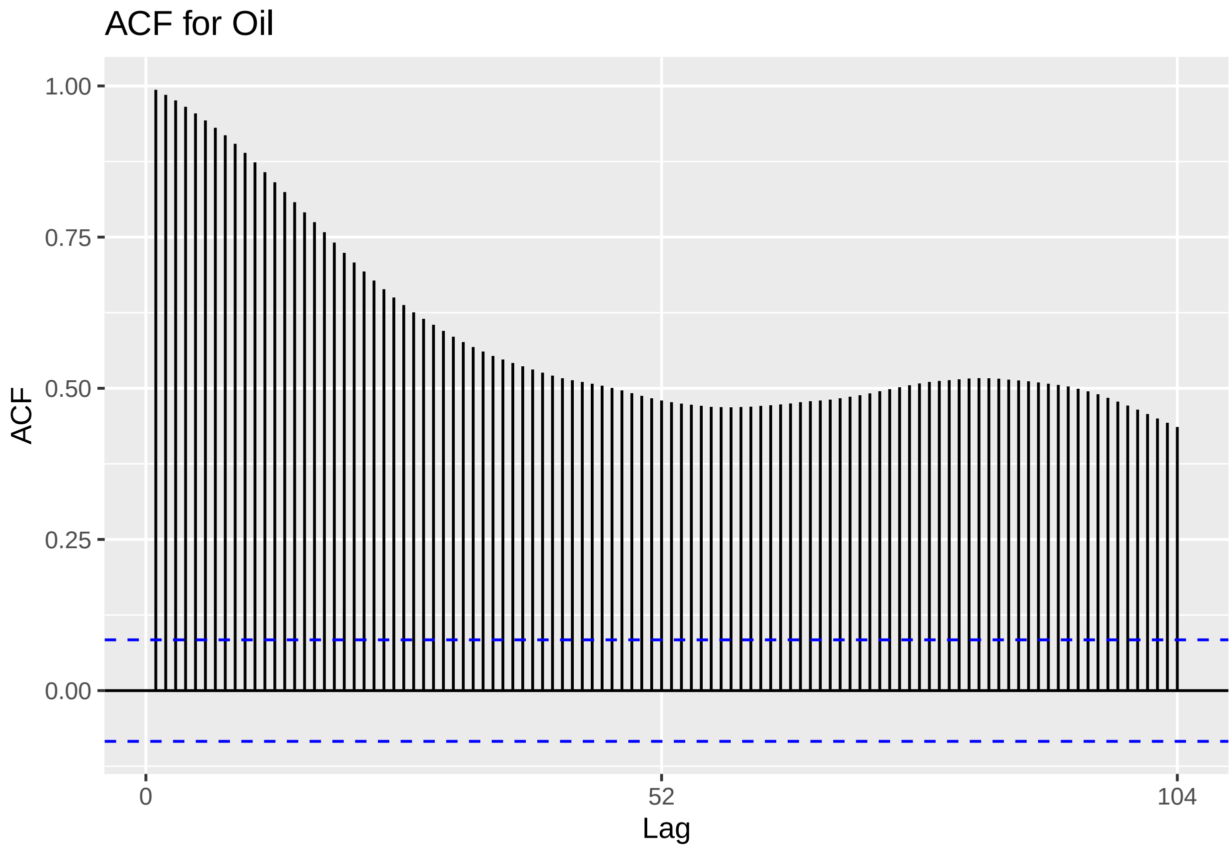
3.1 Finding a Suitable ARIMA Model (oil)

Task: Find a suitable ARIMA(p, d, q) model for the data set `oil` present in the library `astsa`. Your modeling should include the following steps in an appropriate order: visualization, unit root test, detrending by differencing (if necessary), transformations (if necessary), ACF and PACF plots when needed, EACF analysis, Q-Q plots, Box-Ljung test, ARIMA fit analysis, control of the parameter redundancy in the fitted model. When performing these steps, always have 2 tentative models at hand and select one of them in the end. Validate your choice by AIC and BIC and write down the equation of the selected model. Finally, perform forecasting of the model 20 observations ahead and provide a suitable plot showing the forecast and its uncertainty.

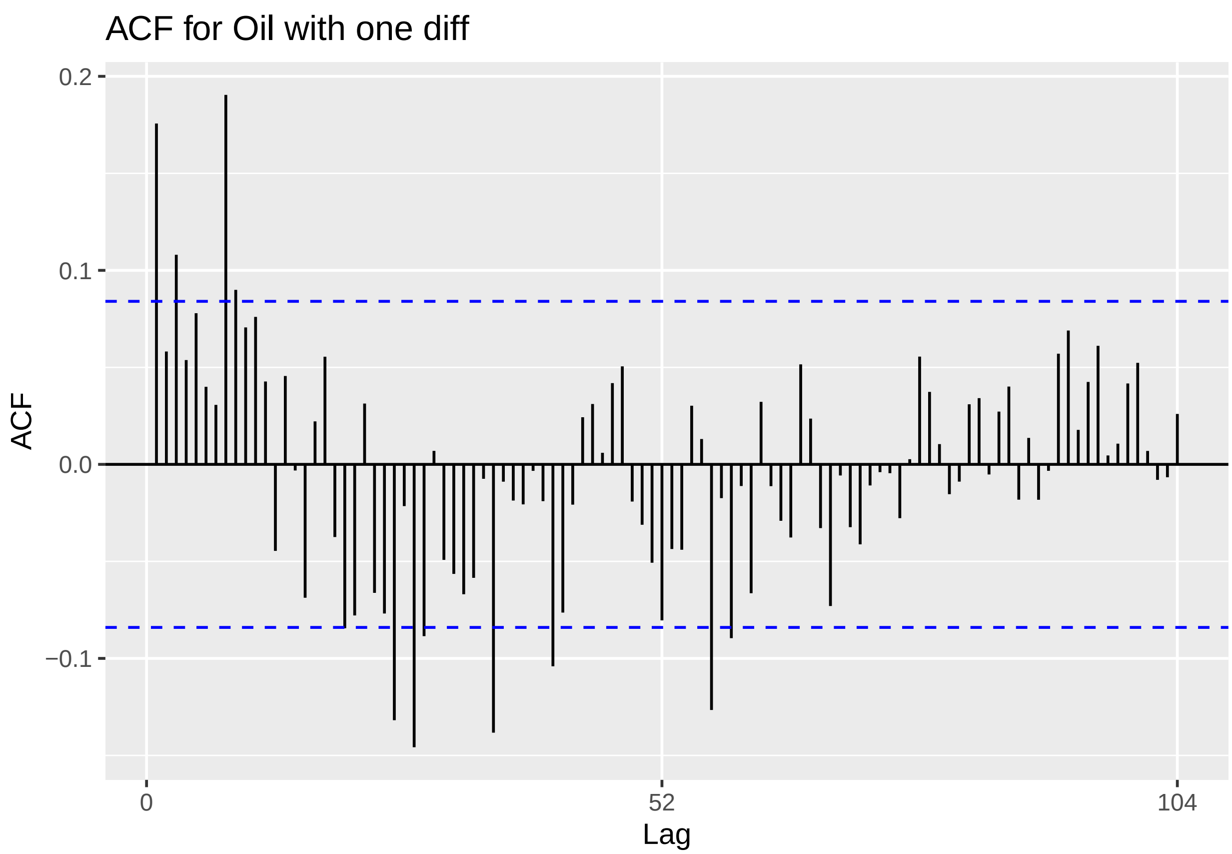
```r
set.seed(12345)

# visualization
autoplot(ts(oil, start = 2000, frequency = 52)) +
           ylab("Price of Oil") +xlab("Year") +
           ggtitle("Price of Oil vs. Years")
```



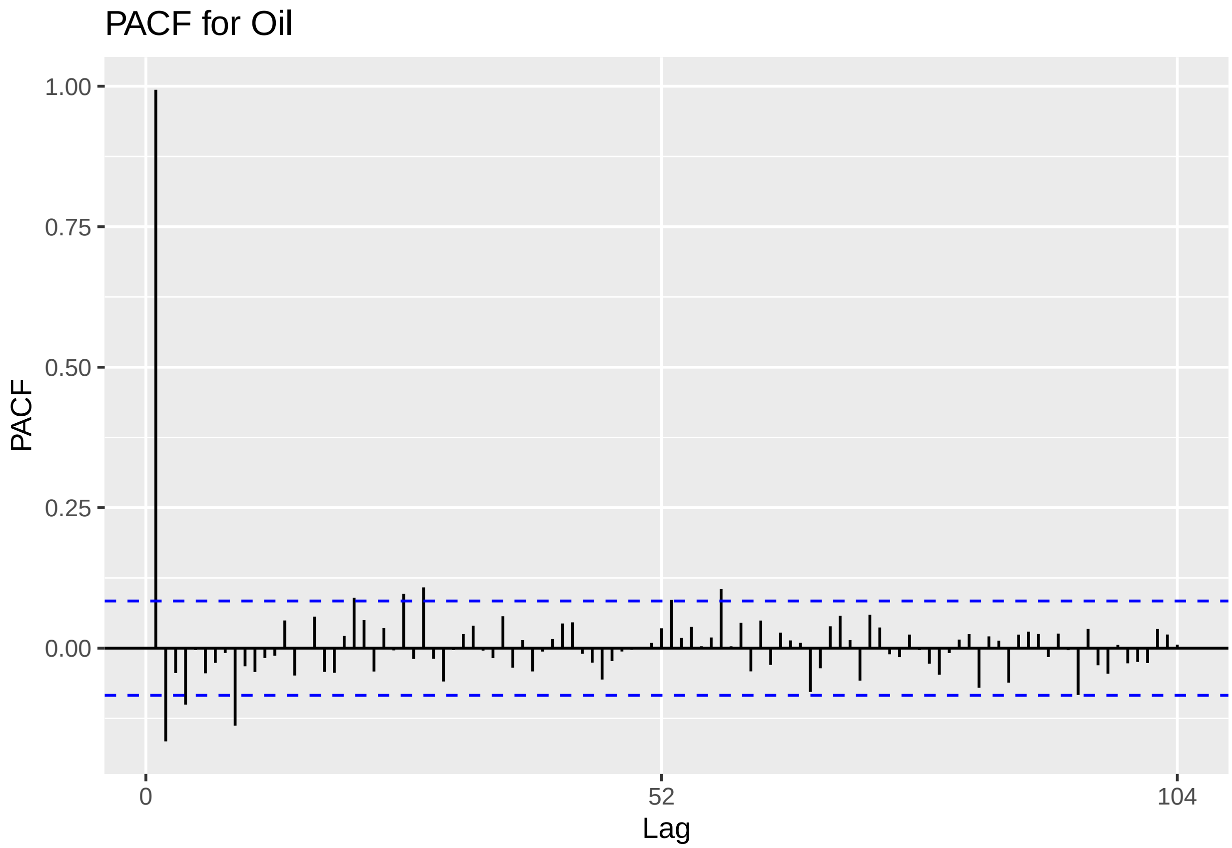

```r
ggAcf(oil) + ggtitle("ACF for Oil")
```



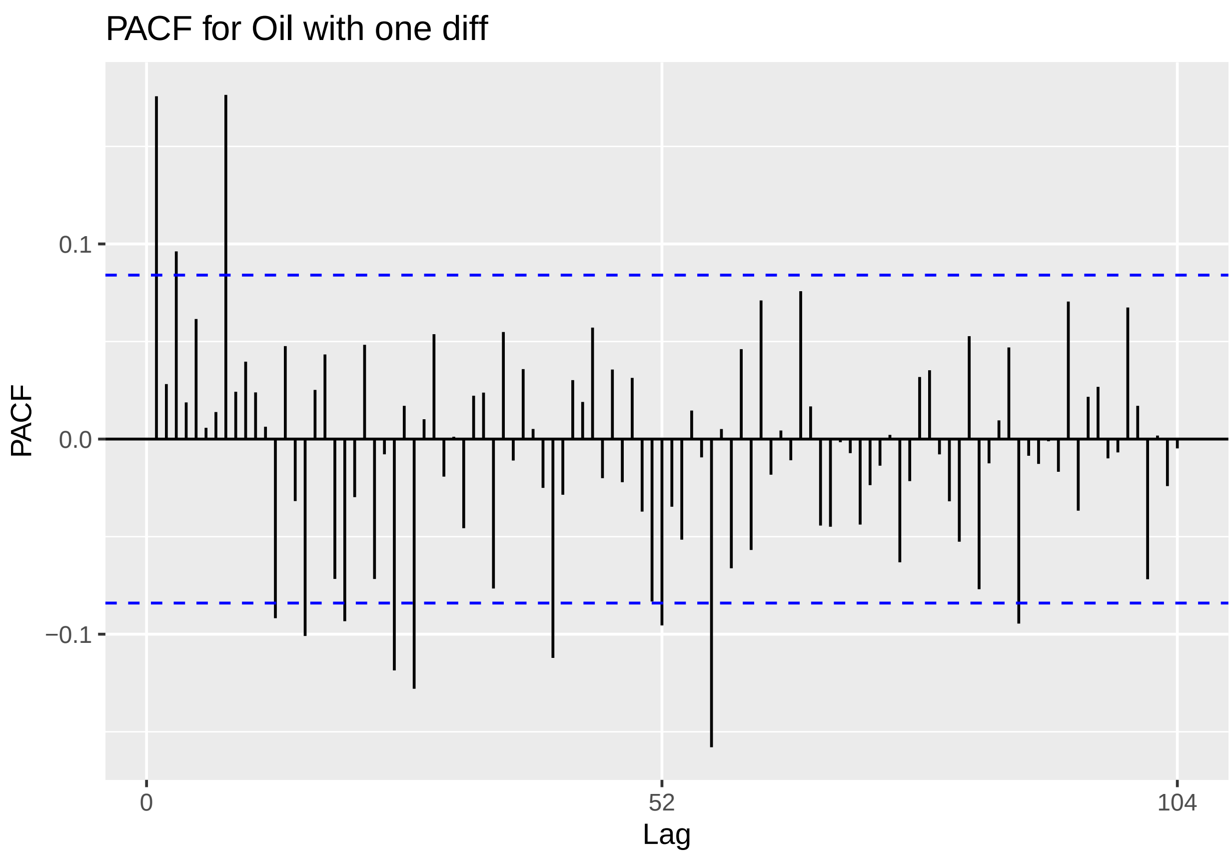

```r
ggAcf(diff(oil)) + ggtitle("ACF for Oil with one diff")
```



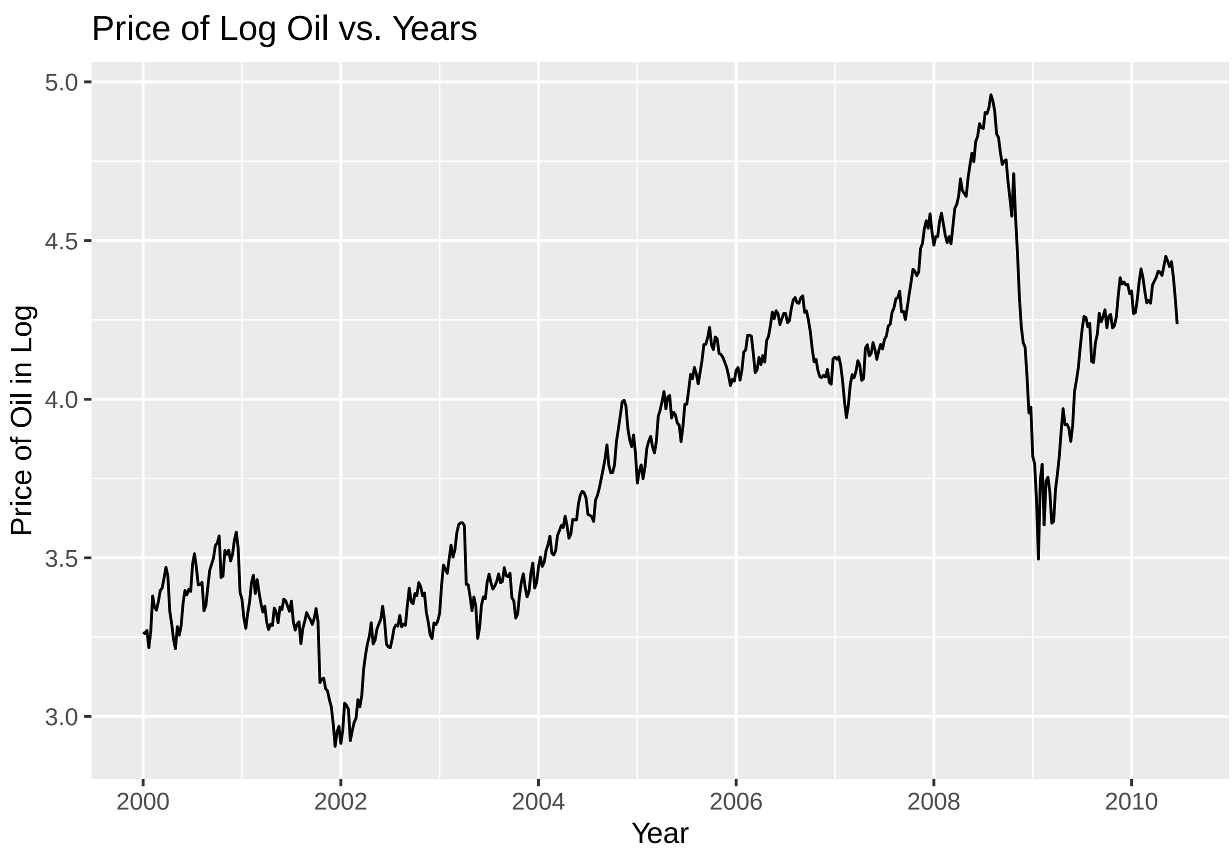

```r
ggPacf(oil) + ggtitle("PACF for Oil")
```



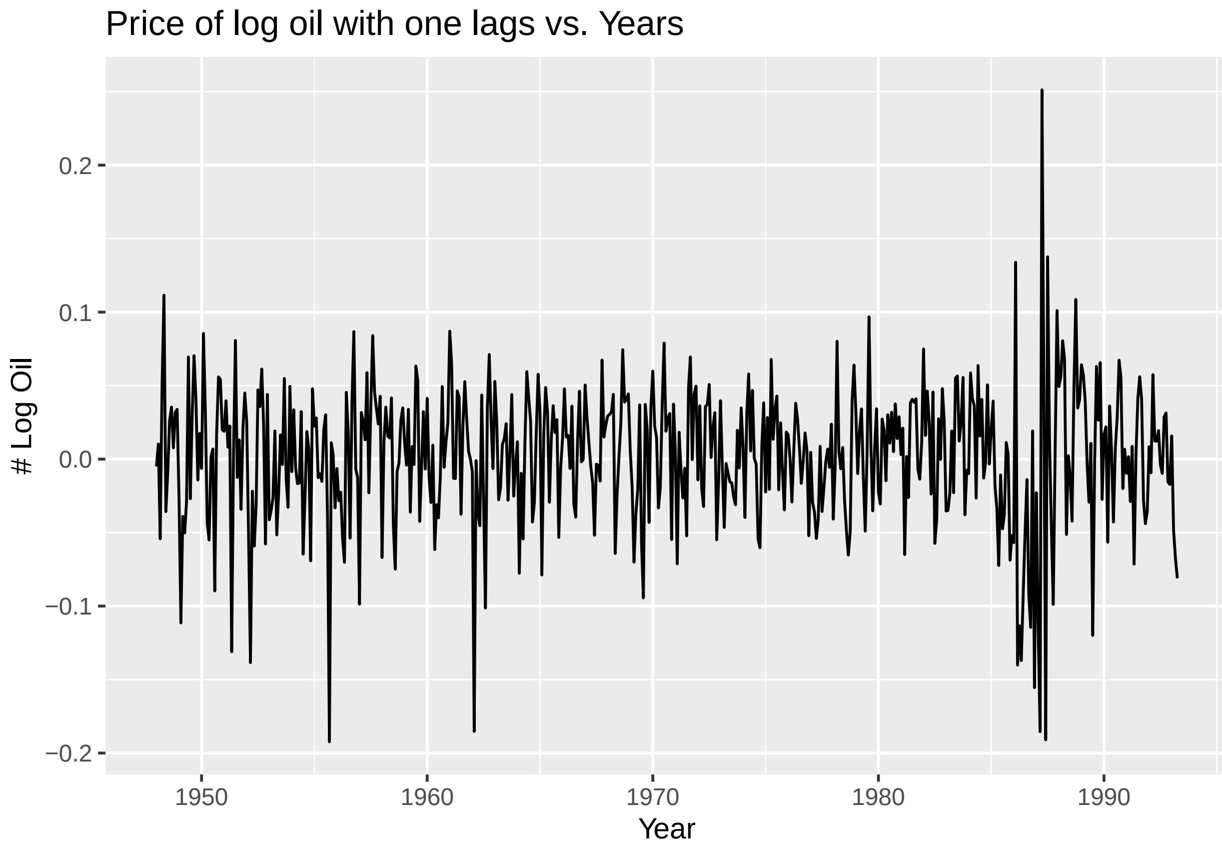

```r
ggPacf(diff(oil)) + ggtitle("PACF for Oil with one diff")
```



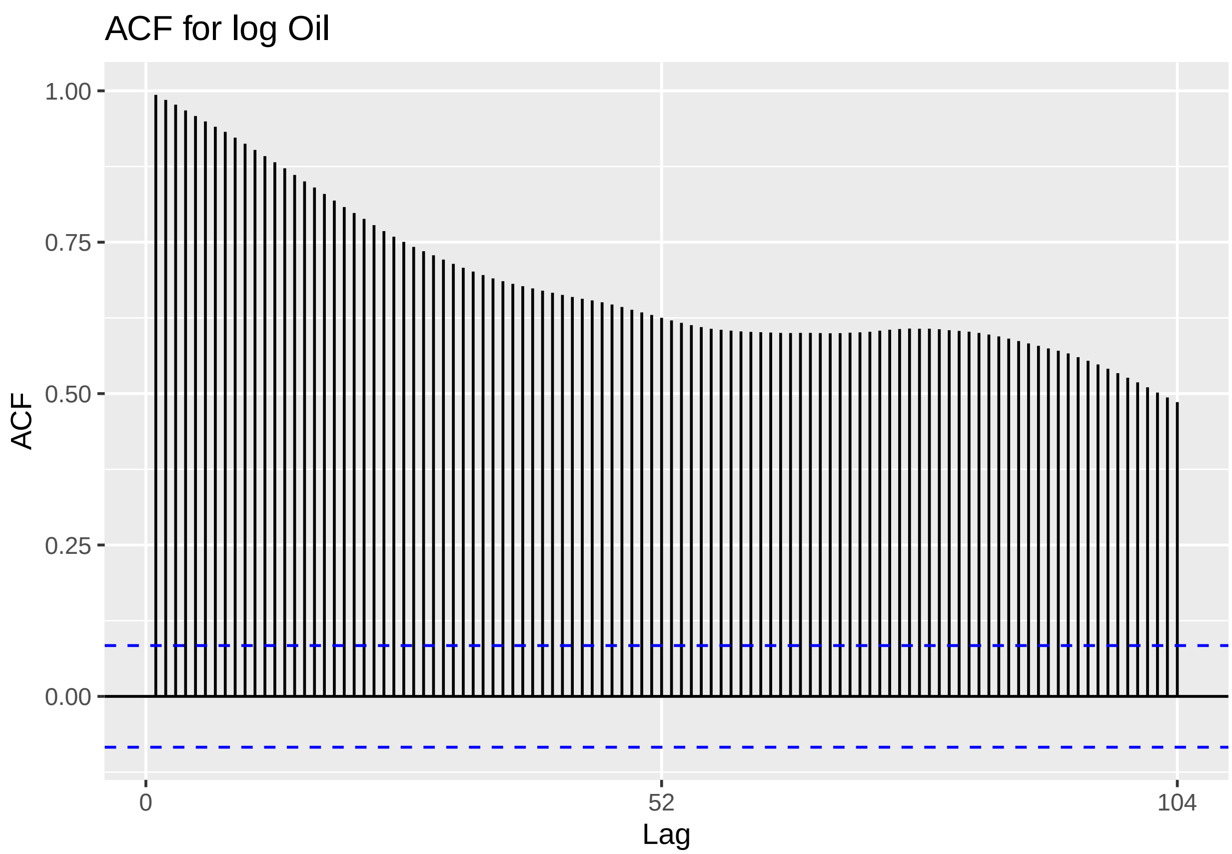

```r
# with log
autoplot(ts(log(oil), start = 2000, frequency = 52)) +
          ylab("Price of Oil in Log") +xlab("Year") +
          ggtitle("Price of Log Oil vs. Years")
```



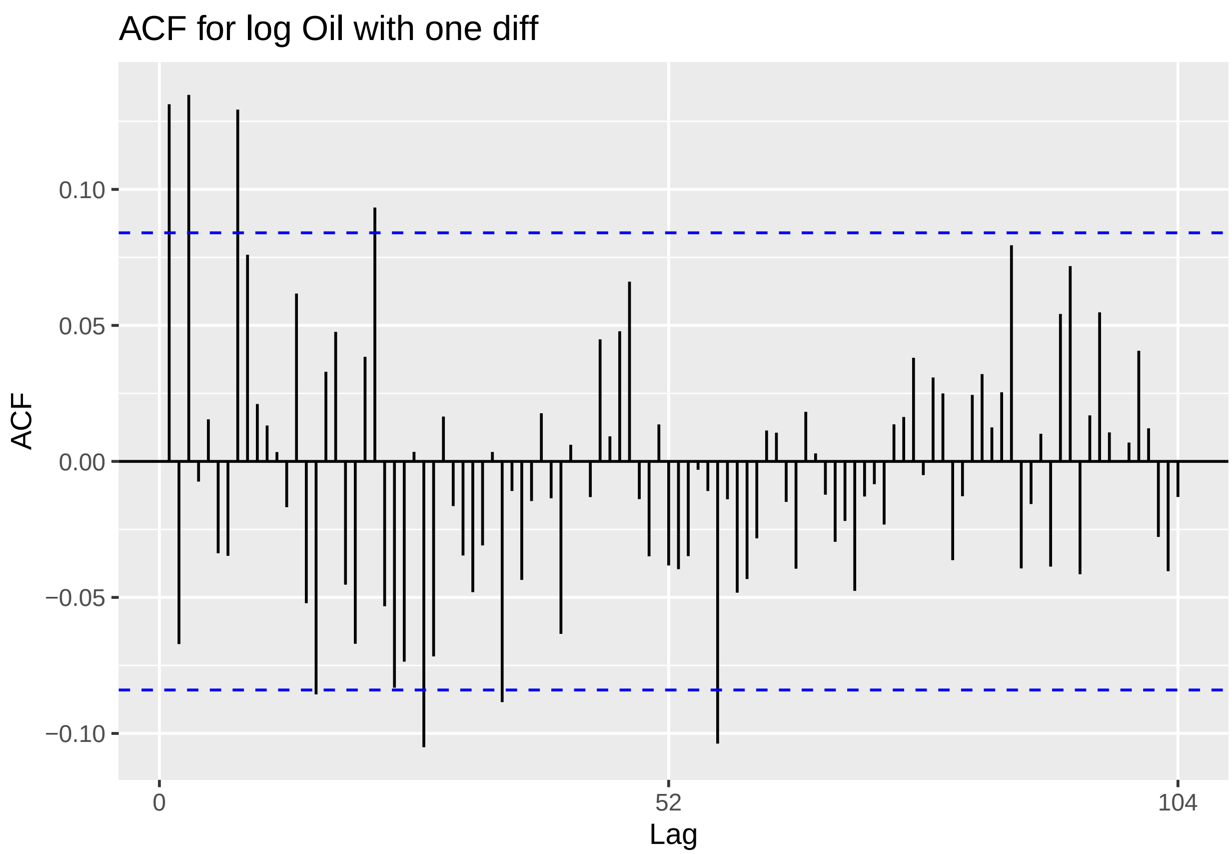

```
autoplot(ts(diff(log(oil), lag=1), start = 1948, frequency = 12)) +
          ylab("# Log Oil") +xlab("Year") +
          ggtitle("Price of log oil with one lags vs. Years")
```



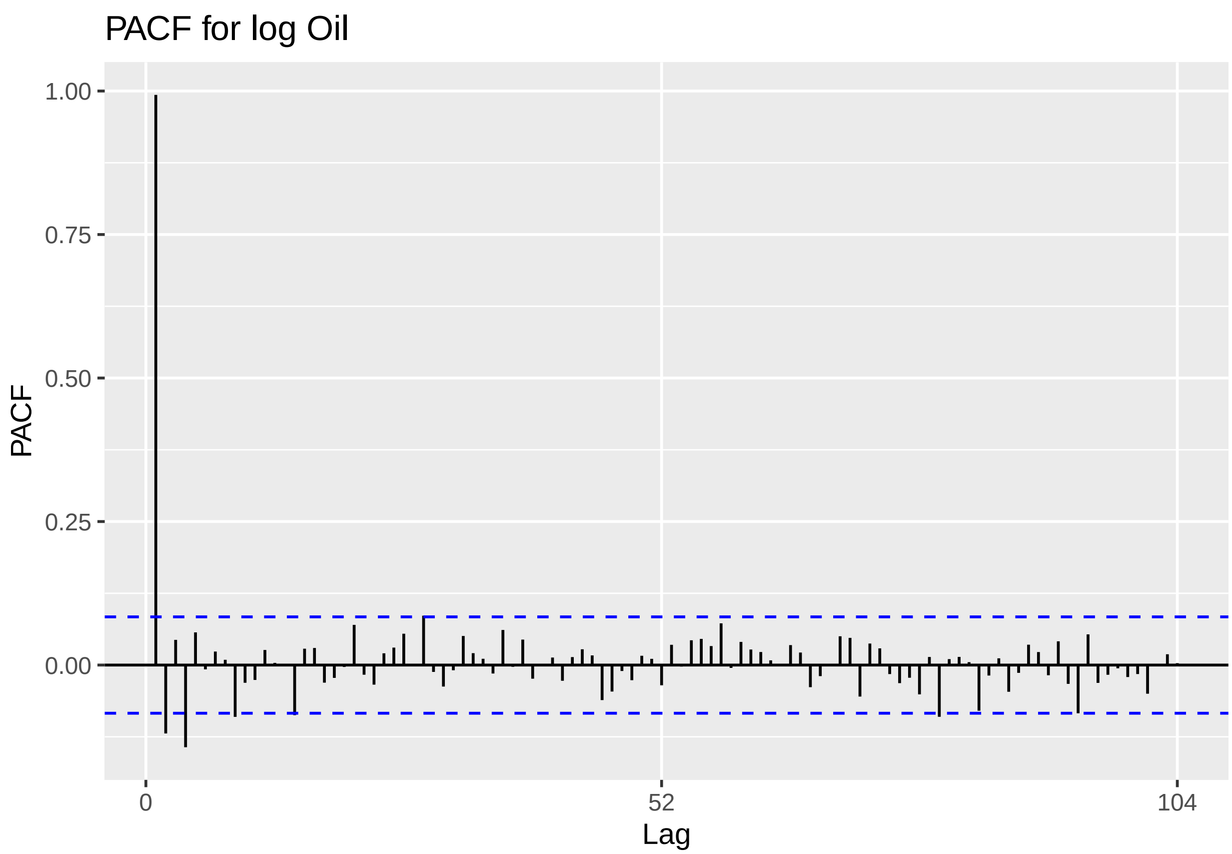

```r
ggAcf(log(oil)) + ggtitle("ACF for log Oil")
```



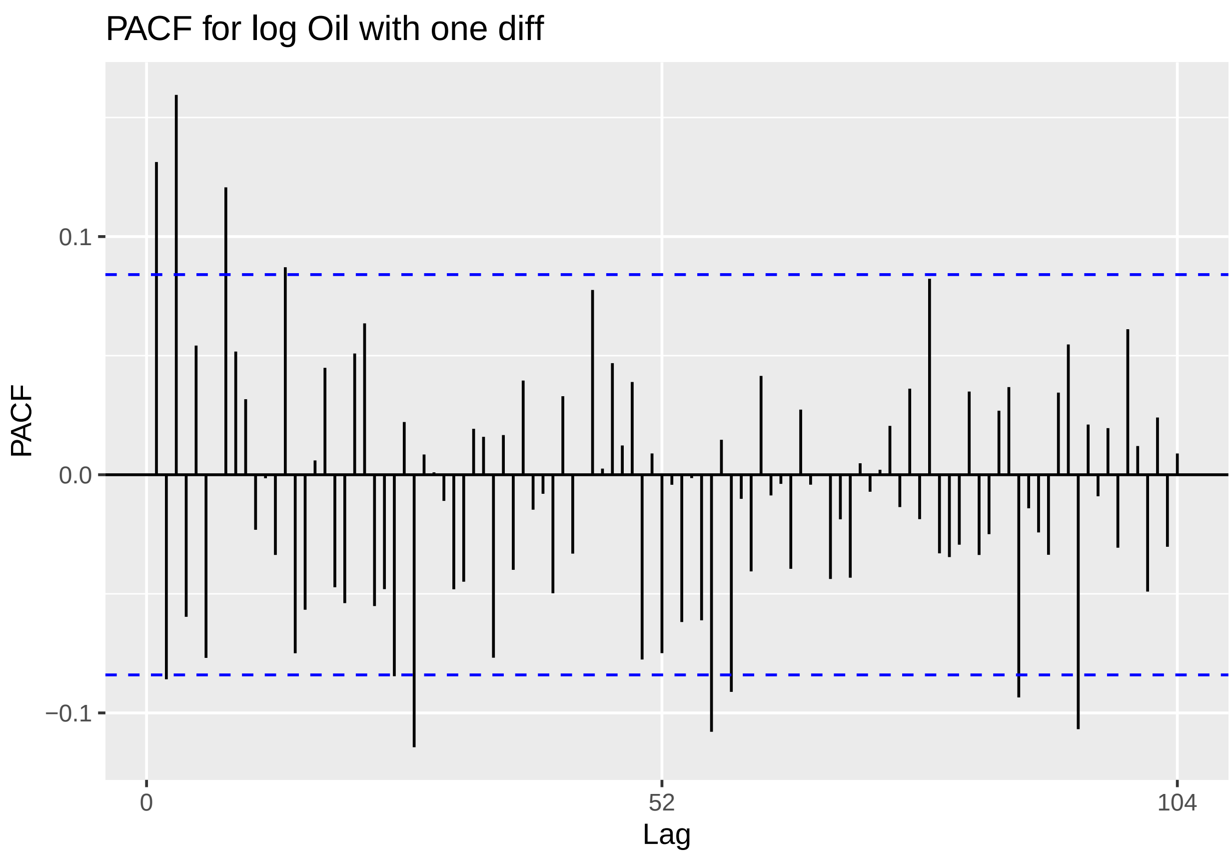

```r
ggAcf(diff(log(oil))) + ggtitle("ACF for log Oil with one diff")
```




```
ggPacf(log(oil)) + ggtitle("PACF for log Oil")
```




```r
ggPacf(diff(log(oil))) + ggtitle("PACF for log Oil with one diff")
```




```r
# EACF
eacf(diff(log(oil)))
```

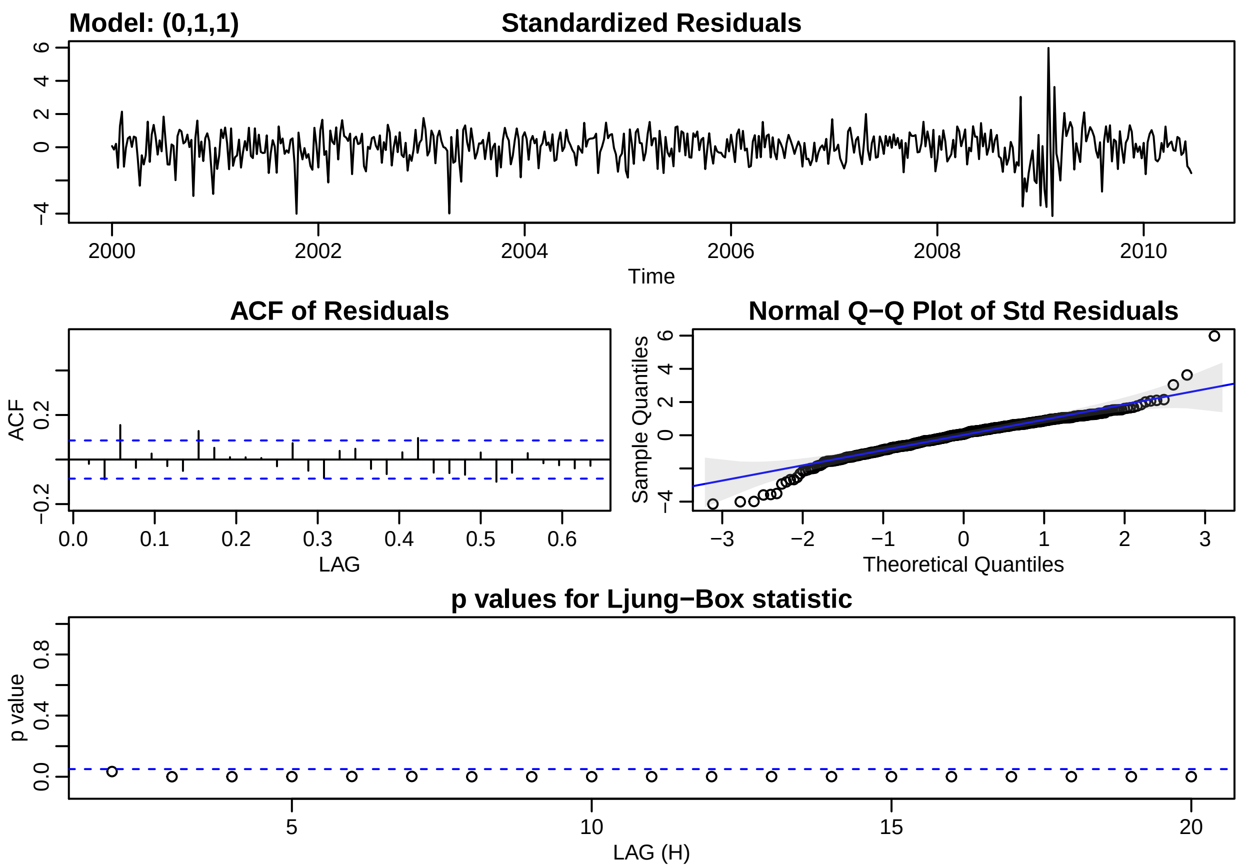
```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x o x o o o o x o o o  o  o  o
## 1 x o x o o o o x o o o  o  o  o
## 2 x x x o o o o x o o o  o  o  o
## 3 x x x o o o o x o o o  o  o  o
## 4 x o x o o o o x o o o  o  o  o
## 5 x x x o x o o x o o o  o  o  o
## 6 o x x o x x o x o o o  o  o  x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

Analysis: ARIMA(0,1,1) or ARIMA(1,1,1) or ARIMA(0,1,3) according to EACF

```r
#Suggested Models
modelA <- sarima(log(oil), 0,1,1)
```

```
## initial  value -3.058495
## iter   2 value -3.068906
## iter   3 value -3.069474
## iter   4 value -3.069476
## iter   4 value -3.069476
## iter   4 value -3.069476
```

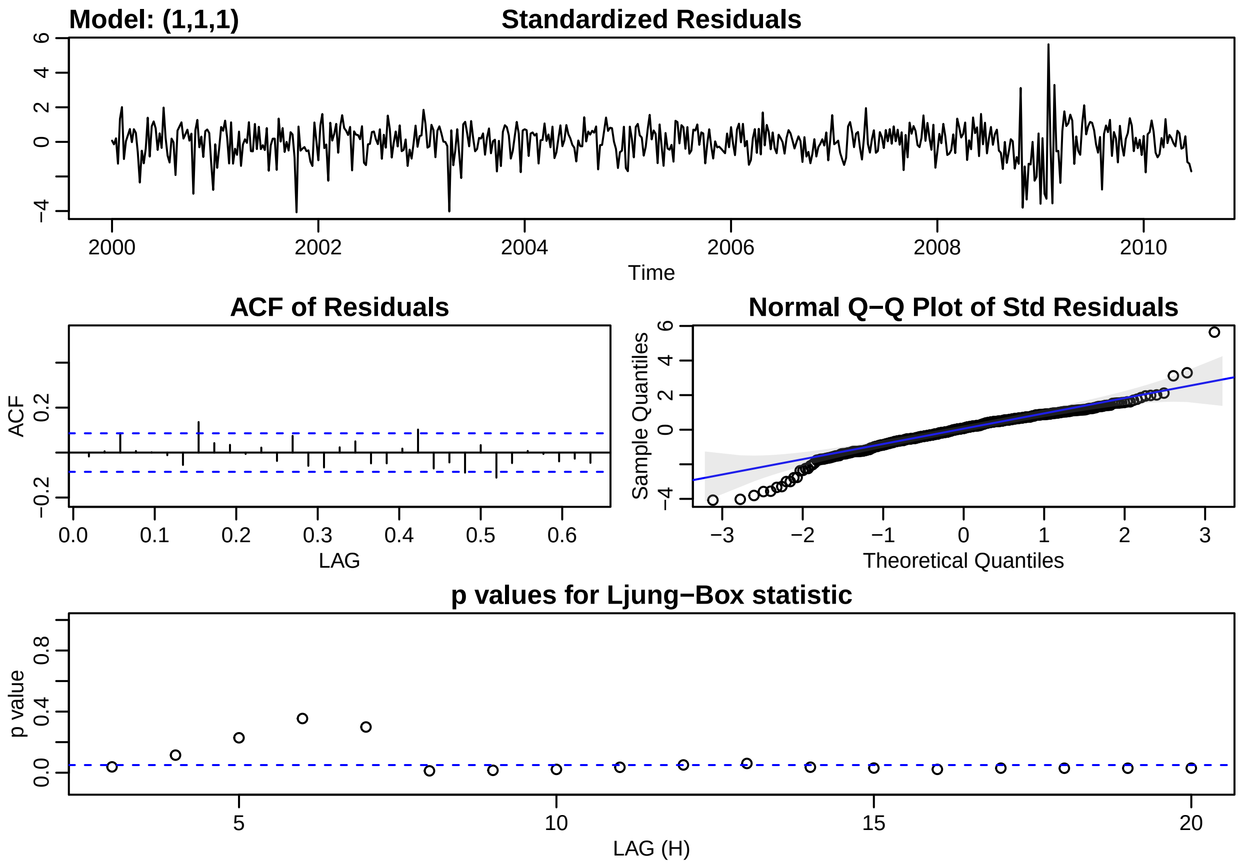
```
## final  value -3.069476
## converged
## initial  value -3.069450
## iter   2 value -3.069450
## iter   2 value -3.069450
## iter   2 value -3.069450
## final  value -3.069450
## converged
```




```r
modelB <- sarima(log(oil), 1,1,1)
```

```
## initial  value -3.057594
## iter   2 value -3.061420
## iter   3 value -3.067360
## iter   4 value -3.067479
## iter   5 value -3.071834
## iter   6 value -3.074359
## iter   7 value -3.074843
## iter   8 value -3.076656
## iter   9 value -3.080467
## iter  10 value -3.081546
## iter  11 value -3.081603
## iter  12 value -3.081615
## iter  13 value -3.081642
## iter  14 value -3.081643
```

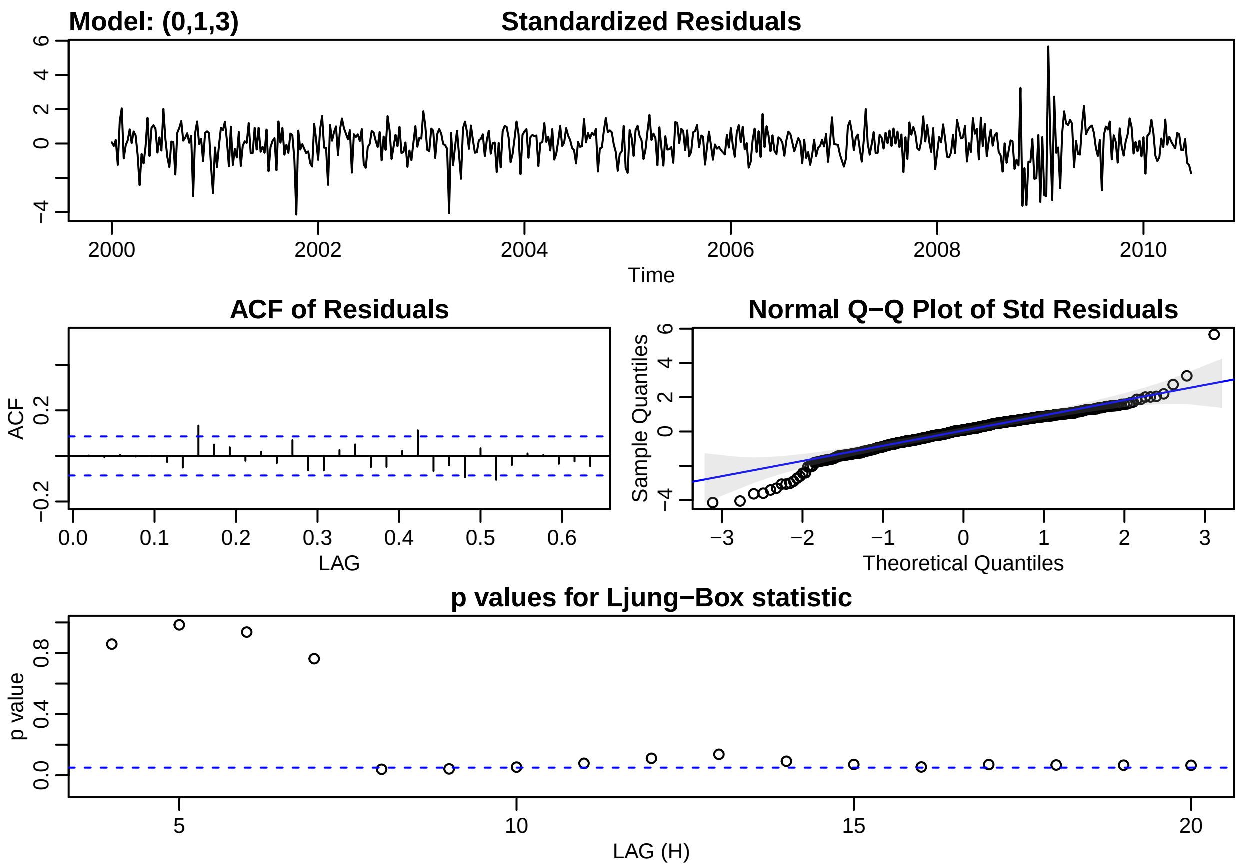
```
## iter  14 value -3.081643
## iter  14 value -3.081643
## final  value -3.081643
## converged
## initial  value -3.082345
## iter   2 value -3.082345
## iter   3 value -3.082346
## iter   4 value -3.082346
## iter   5 value -3.082346
## iter   5 value -3.082346
## iter   5 value -3.082346
## final  value -3.082346
## converged
```




```
modelC <- sarima(log(oil), 0,1,3)
```

```
## initial  value -3.058495
## iter   2 value -3.086110
## iter   3 value -3.086980
## iter   4 value -3.087501
## iter   5 value -3.087521
## iter   6 value -3.087521
## iter   7 value -3.087522
## iter   8 value -3.087522
## iter   9 value -3.087522
```

```
## iter   9 value -3.087522
## iter   9 value -3.087522
## final  value -3.087522
## converged
## initial  value -3.087448
## iter   2 value -3.087448
## iter   3 value -3.087449
## iter   3 value -3.087449
## iter   3 value -3.087449
## final  value -3.087449
## converged
```




```r
#ADF test
adf.test(modelA$fit$residuals)
```

```
## Warning in adf.test(modelA$fit$residuals): p-value smaller than printed p-
## value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  modelA$fit$residuals
## Dickey-Fuller = -6.5298, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```r
adf.test(modelB$fit$residuals)
```

```
## Warning in adf.test(modelB$fit$residuals): p-value smaller than printed p-
## value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  modelB$fit$residuals
## Dickey-Fuller = -6.461, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```r
adf.test(modelC$fit$residuals)
```

```
## Warning in adf.test(modelC$fit$residuals): p-value smaller than printed p-
## value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  modelC$fit$residuals
## Dickey-Fuller = -6.7187, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```r
#Redundancy check
summary(modelA$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
##       0.1701    0.0018
## s.e.  0.0499    0.0023
##
## sigma^2 estimated as 0.002157:  log likelihood = 897.88,  aic = -1789.76
##
## Training set error measures:
```

```
## Warning in trainingaccuracy(f, test, d, D): test elements must be within
## sample
```

```
##               ME RMSE MAE MPE MAPE
## Training set NaN  NaN NaN NaN  NaN
```

```r
summary(modelB$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     ma1  constant
##       -0.5264  0.7146    0.0018
## s.e.   0.0871  0.0683    0.0022
##
## sigma^2 estimated as 0.002102:  log likelihood = 904.89,  aic = -1801.79
##
## Training set error measures:

## Warning in trainingaccuracy(f, test, d, D): test elements must be within
## sample

##                ME RMSE MAE MPE MAPE
## Training set NaN  NaN NaN NaN  NaN
```

```
summary(modelC$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  constant
##       0.1688  -0.0900  0.1447    0.0017
## s.e.  0.0424   0.0425  0.0430    0.0024
##
## sigma^2 estimated as 0.00208:  log likelihood = 907.67,  aic = -1805.34
##
## Training set error measures:

## Warning in trainingaccuracy(f, test, d, D): test elements must be within
## sample

##                ME RMSE MAE MPE MAPE
## Training set NaN  NaN NaN NaN  NaN
```

```
#BIC
BIC(modelA$fit)
```

```
## [1] -1776.859
```

```
BIC(modelB$fit)
```

```
## [1] -1784.592
```

```r
BIC(modelC$fit)
```

```
## [1] -1783.844
```

```r
#AIC
AIC(modelA$fit)
```

```
## [1] -1789.756
```

```r
AIC(modelB$fit)
```

```
## [1] -1801.787
```

```r
AIC(modelC$fit)
```

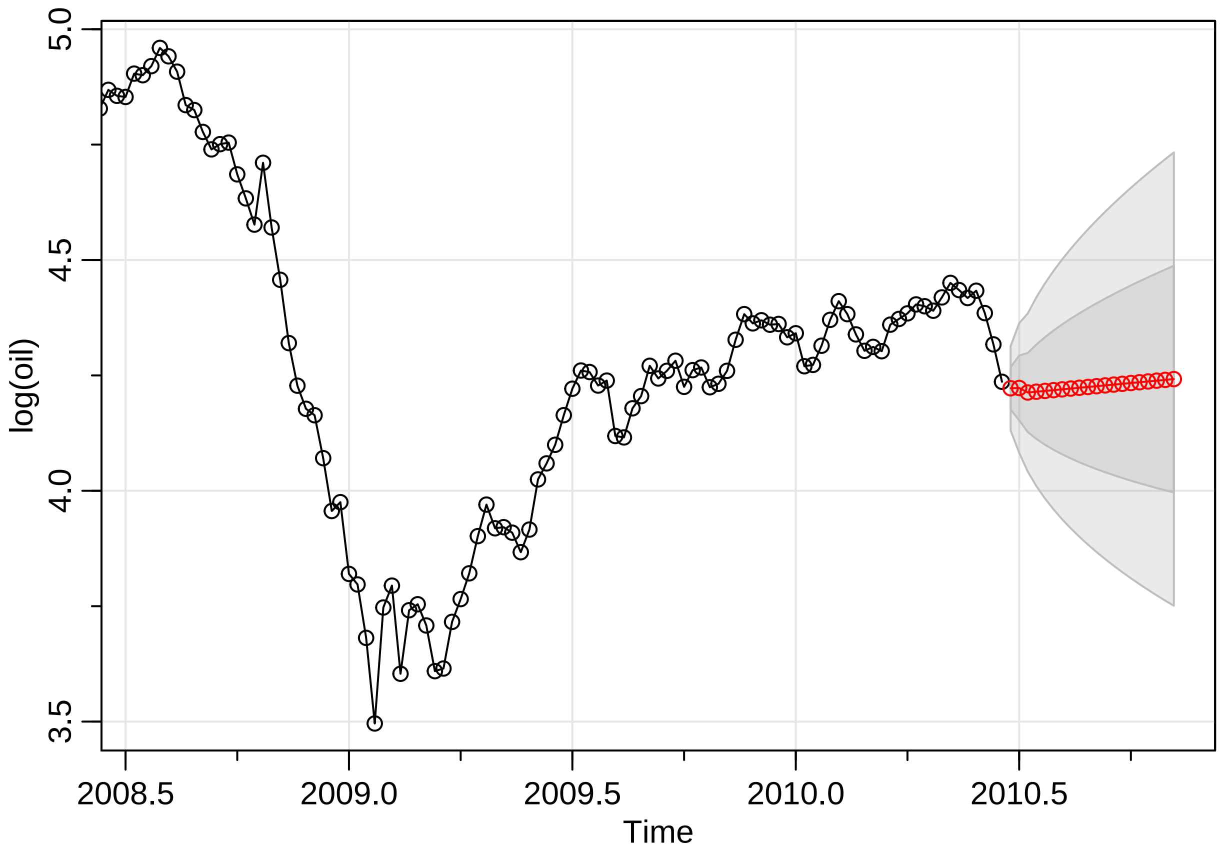
```
## [1] -1805.339
```

```r
#Model C is the best
```

According to AIC and BIC the ModelC (ARIMA 0,1,3) is the best

Model equation is $\Delta x_t = w_t + 0.1688w_{t-1} - 0.0900w_{t-2} + 0.1447w_{t-3}$

```r
#Forecasting
sarima.for(log(oil), 0,1,3, n.ahead = 20)
```

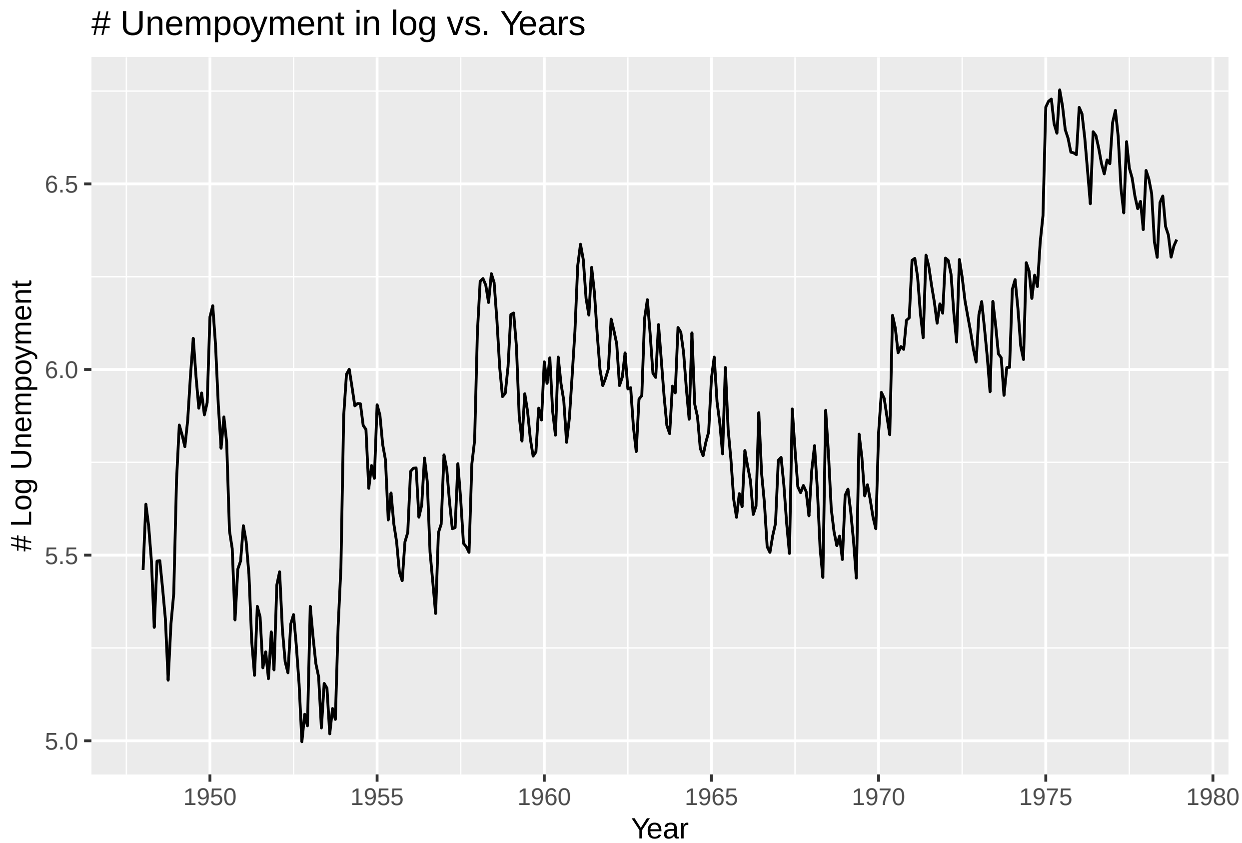


```
## $pred
## Time Series:
## Start = c(2010, 26)
## End = c(2010, 45)
## Frequency = 52
##  [1] 4.222141 4.222731 4.212938 4.214647 4.216356 4.218066 4.219775
##  [8] 4.221485 4.223194 4.224904 4.226613 4.228323 4.230032 4.231741
## [15] 4.233451 4.235160 4.236870 4.238579 4.240289 4.241998
##
## $se
## Time Series:
## Start = c(2010, 26)
## End = c(2010, 45)
## Frequency = 52
##  [1] 0.04561249 0.07016150 0.08569792 0.10226755 0.11650396 0.12918085
##  [7] 0.14072033 0.15138273 0.16134203 0.17072132 0.17961149 0.18808192
## [13] 0.19618697 0.20397021 0.21146718 0.21870731 0.22571532 0.23251220
## [19] 0.23911597 0.24554218
```

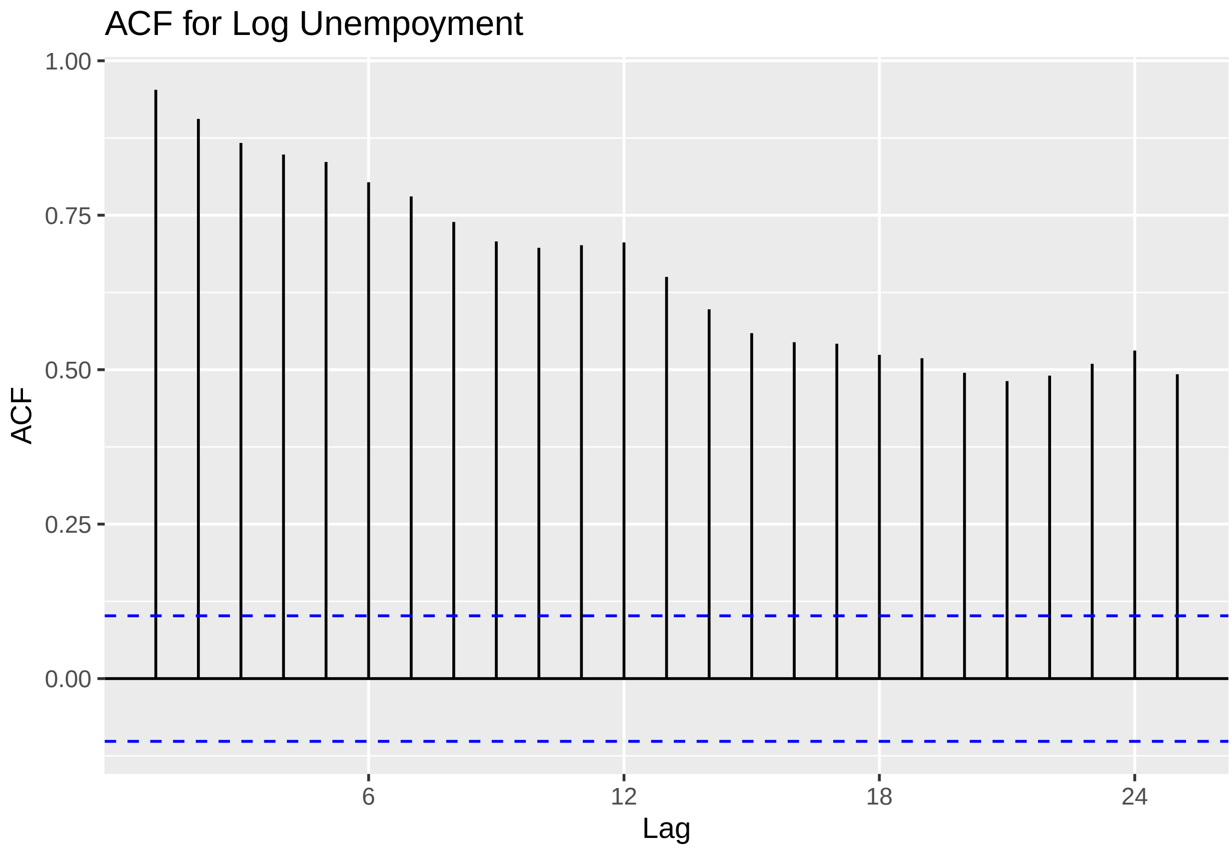
3.2 Finding a Suitable ARIMA Model (unemp)

Task: Find a suitable ARIMA$(p, d, q) \times (P, D, Q)_s$ model for the data set unemp present in the library `astsa`. Your modeling should include the following steps in an appropriate order: visualization, detrending by differencing (if necessary), transformations (if necessary), ACF and PACF plots when needed, EACF analysis, Q-Q plots, Box-Ljung test, ARIMA fit analysis, control of the parameter redundancy in the fitted model. When performing these steps, always have 2 tentative models at hand and select one of them in the end. Validate your choice by AIC and BIC and write down the equation of the selected model (write in the back-shift operator notation without expanding the brackets). Finally, perform forecasting of the model 20 observations ahead and provide a suitable plot showing the forecast and its uncertainty.

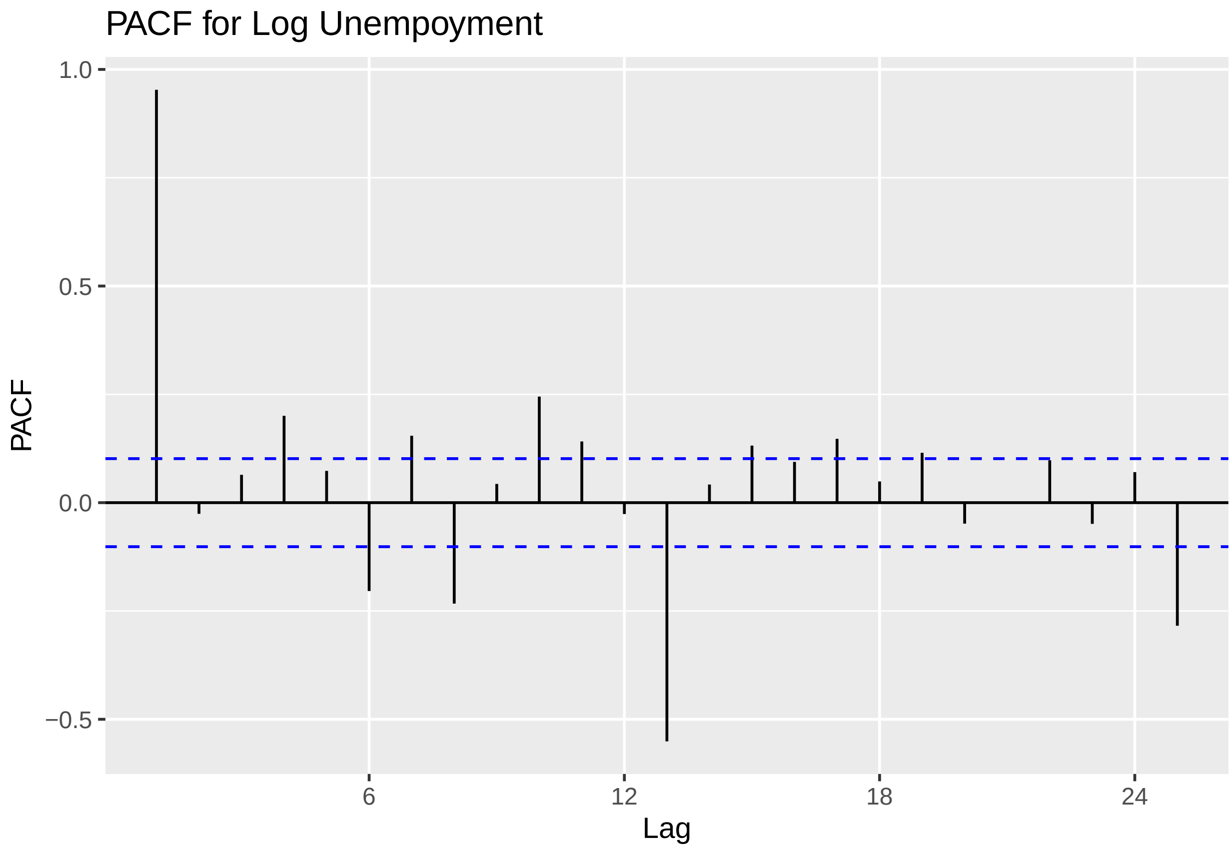
```r
set.seed(12345)

# visualization with log
autoplot(ts(log(unemp), start = 1948, frequency = 12)) +
          ylab("# Log Unempoyment") +xlab("Year") +
          ggtitle("# Unempoyment in log vs. Years")
```



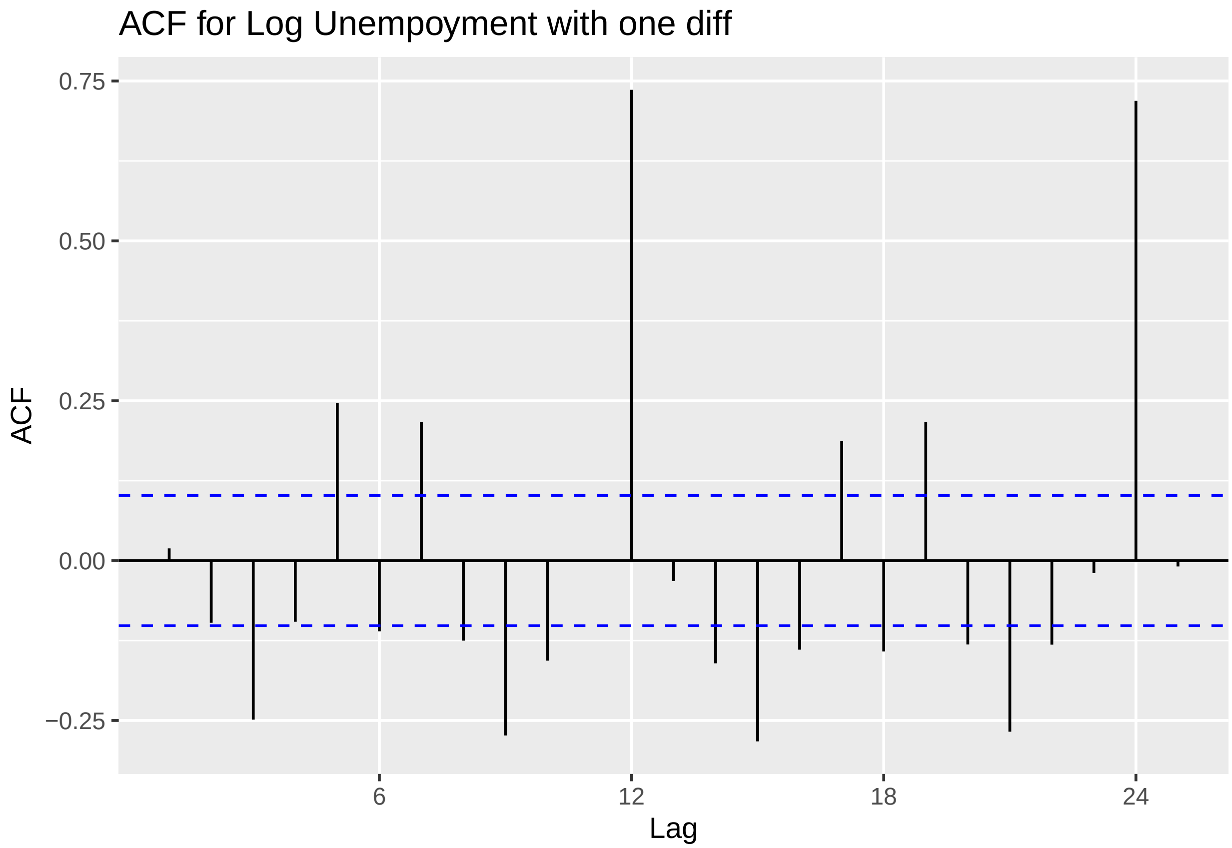

```r
ggAcf(log(unemp)) + ggtitle("ACF for Log Unempoyment")
```



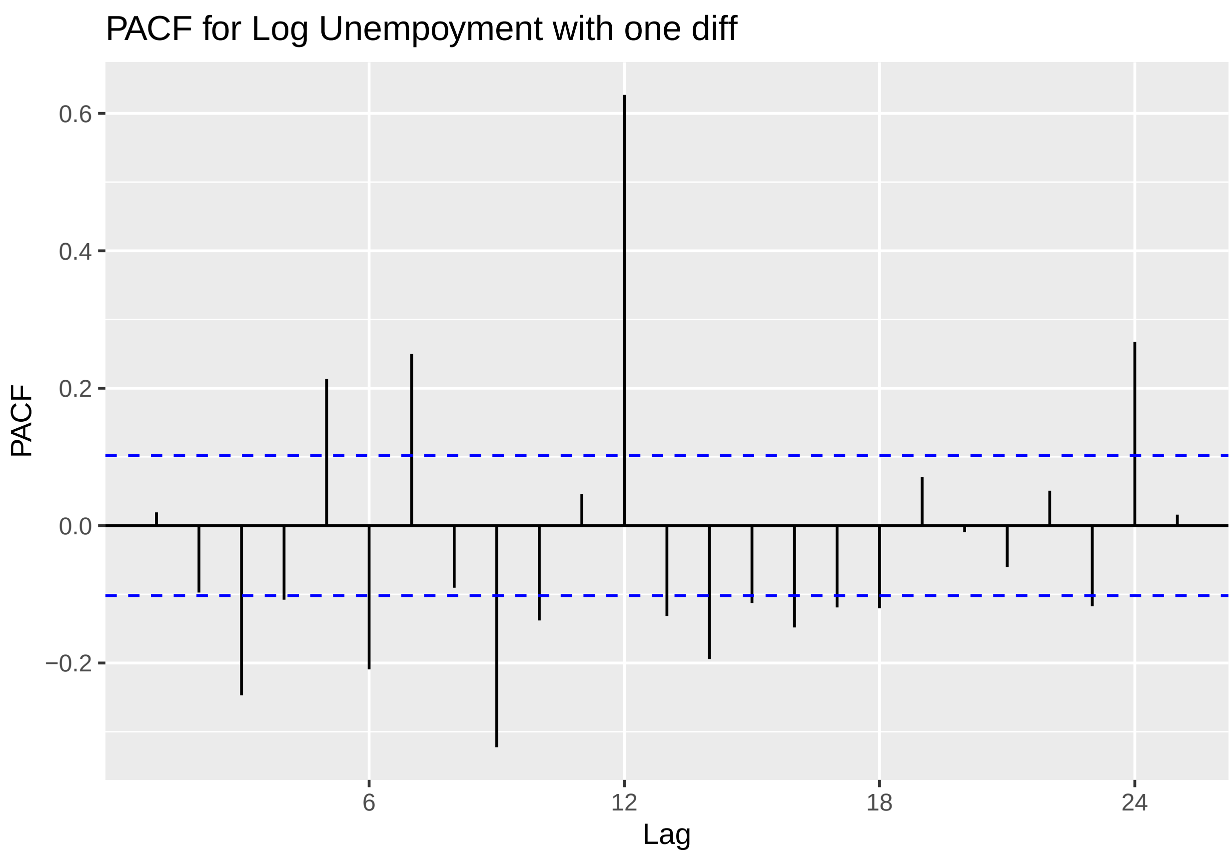

```r
ggPacf(log(unemp)) + ggtitle("PACF for Log Unempoyment")
```




```r
ggAcf(diff(log(unemp))) + ggtitle("ACF for Log Unempoyment with one diff")
```

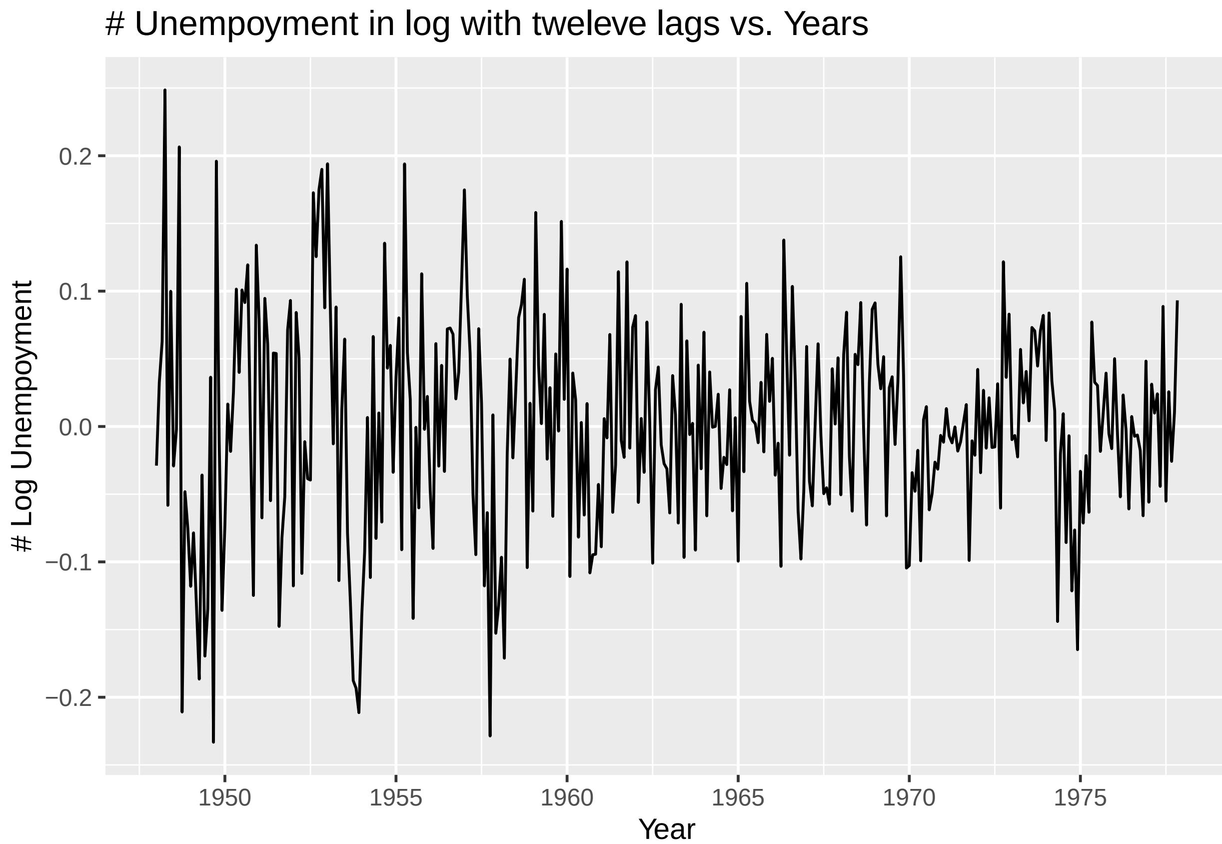



```r
ggPacf(diff(log(unemp))) + ggtitle("PACF for Log Unempoyment with one diff")
```

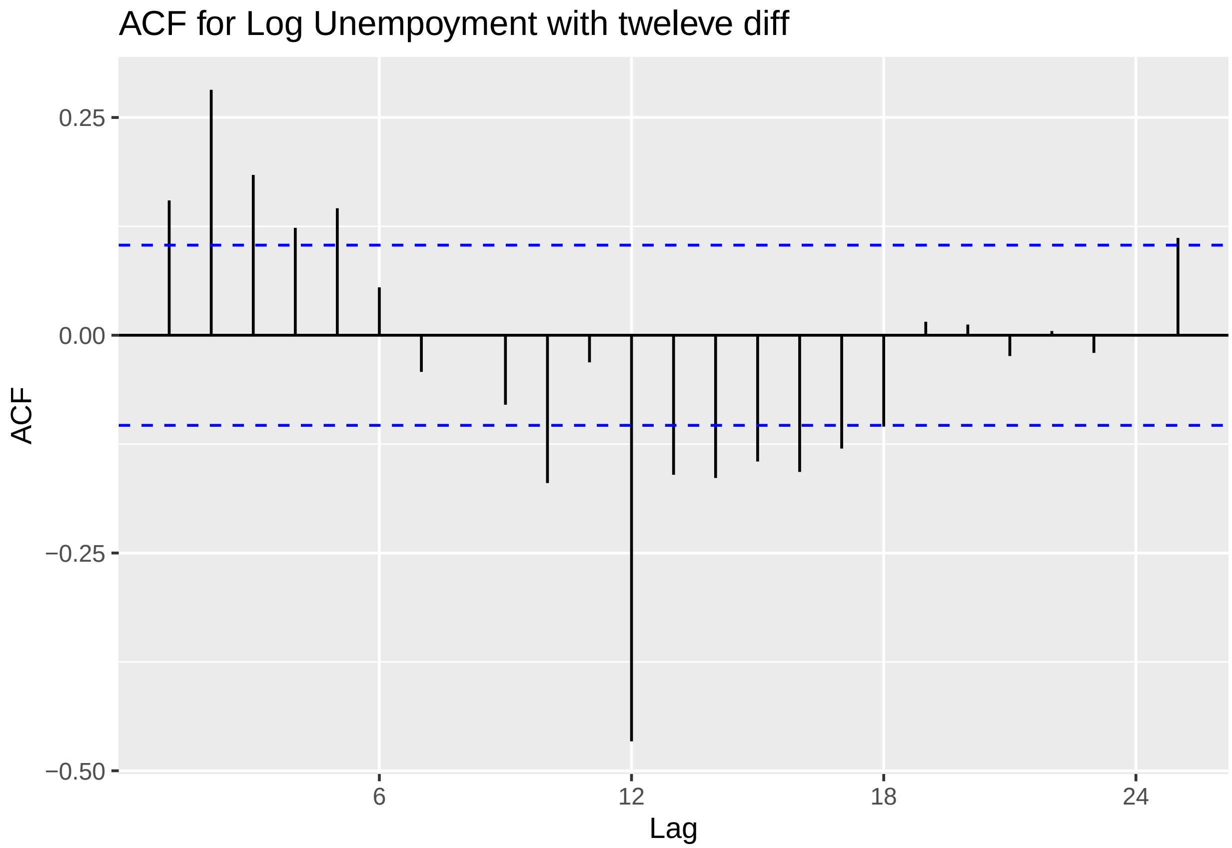


Analysis: ACF with one lag shows that there are seasonal components at 12,24…

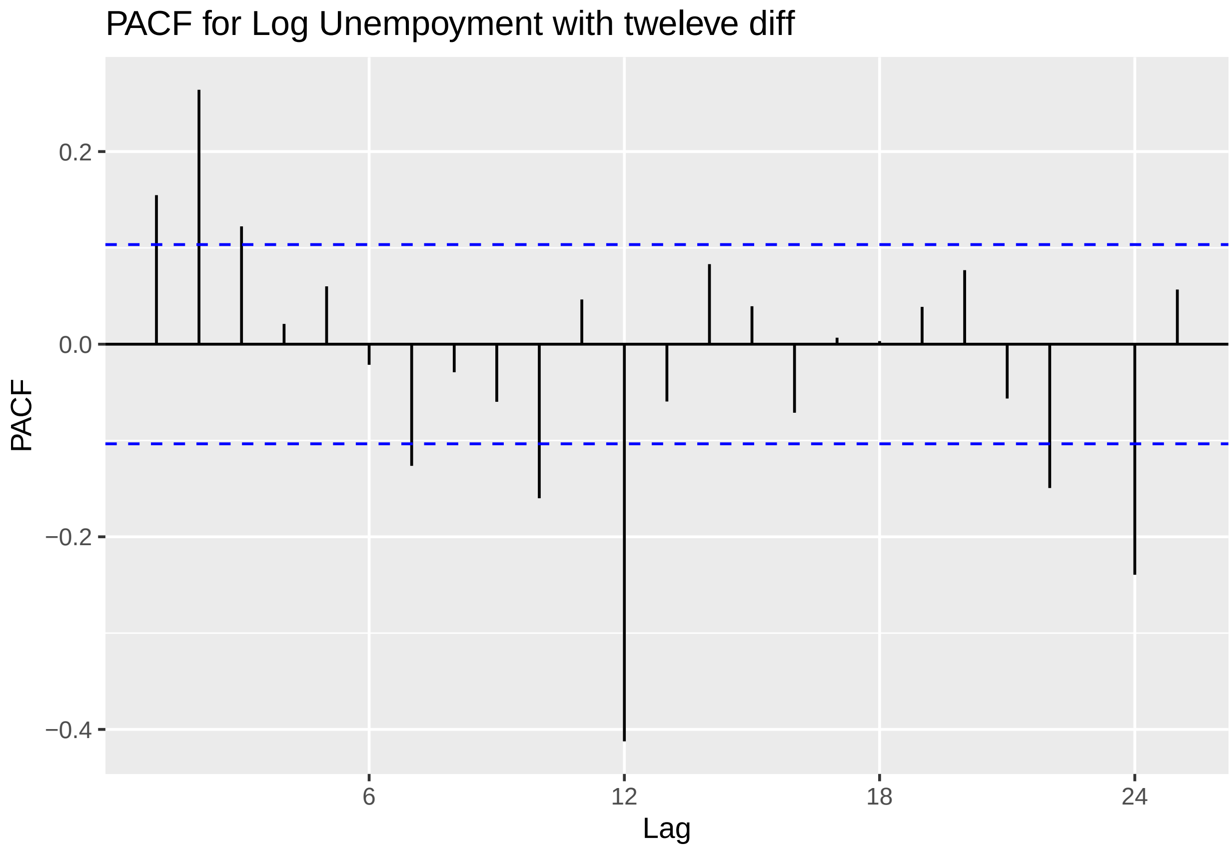
```r
# visualization with log
autoplot(ts(diff(diff(log(unemp)), lag=12), start = 1948, frequency = 12)) +
           ylab("# Log Unempoyment") +xlab("Year") +
           ggtitle("# Unempoyment in log with tweleve lags vs. Years")
```




```r
ggAcf(diff(diff(log(unemp)), lag=12)) + ggtitle("ACF for Log Unempoyment with tweleve diff")
```




```
ggPacf(diff(diff(log(unemp)), lag=12)) + ggtitle("PACF for Log Unempoyment with tweleve diff")
```



Analysis: From the plot of time series we can say that its much more stationary than single lag. Thus clearly two lags are needed.

4 EACF

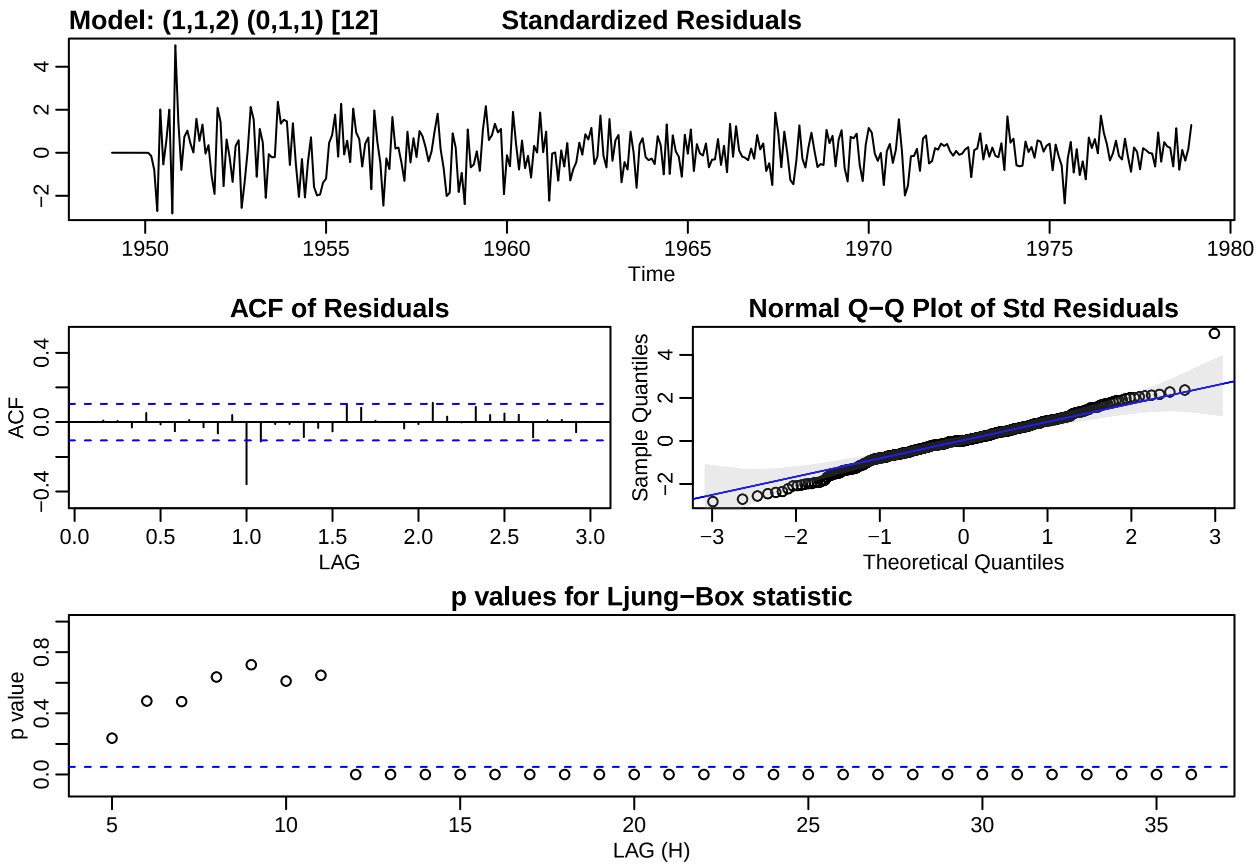
```r
eacf(diff(diff(log(unemp)), lag=12))
```

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

Analysis: The best ARIMA model is $ARIMA(1, 1, 2)(0, 1, 1)_{12}$ and $ARIMA(1, 1, 3)(0, 1, 1)_{12}$

```r
#Suggested Models
modelA <- sarima(diff(diff(log(unemp)), lag=12), 1,1,2,0,1,1,12)
```

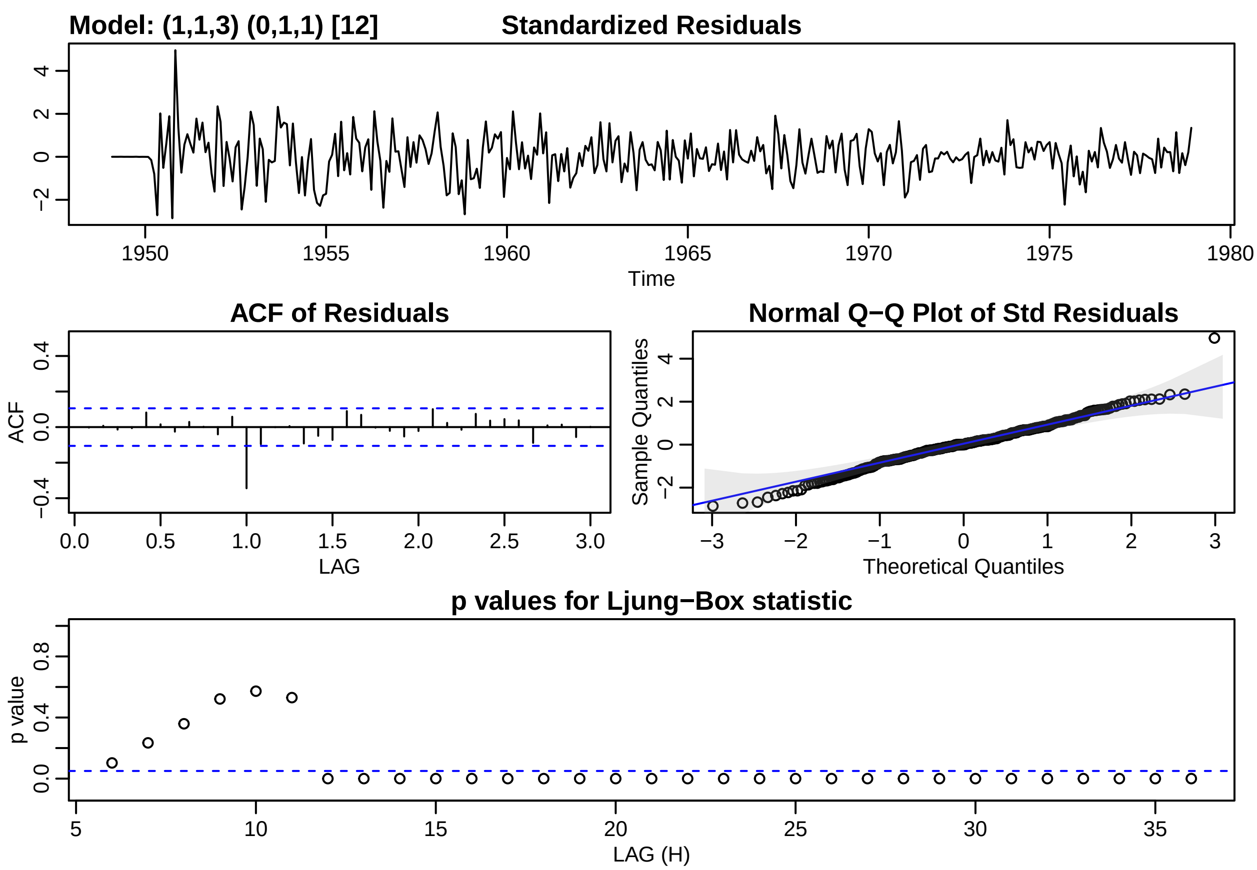
```
## initial  value -1.769018
## iter   2 value -2.240282
## iter   3 value -2.298505
## iter   4 value -2.337902
## iter   5 value -2.369419
## iter   6 value -2.374392
## iter   7 value -2.374804
## iter   8 value -2.375212
## iter   9 value -2.375230
## iter  10 value -2.375240
## iter  11 value -2.375245
## iter  12 value -2.375265
## iter  13 value -2.375352
## iter  14 value -2.375381
## iter  15 value -2.375382
## iter  15 value -2.375382
## iter  15 value -2.375382
## final  value -2.375382
## converged
## initial  value -2.433389
## iter   2 value -2.447200
## iter   3 value -2.484487
## iter   4 value -2.487461
## iter   5 value -2.487467
## iter   6 value -2.487482
## iter   7 value -2.487490
## iter   8 value -2.487537
## iter   9 value -2.487597
## iter  10 value -2.487660
## iter  11 value -2.487684
## iter  12 value -2.487686
## iter  13 value -2.487694
## iter  14 value -2.487697
## iter  15 value -2.487697
## iter  16 value -2.487698
## iter  17 value -2.487700
## iter  18 value -2.487703
## iter  19 value -2.487706
## iter  20 value -2.487706
## iter  21 value -2.487706
## iter  21 value -2.487706
## iter  21 value -2.487706
## final  value -2.487706
## converged
```




```r
modelB <- sarima(diff(diff(log(unemp)), lag=12), 1,1,3,0,1,1,12)
```

```
## initial  value -1.769018
## iter   2 value -2.244155
## iter   3 value -2.302292
## iter   4 value -2.340742
## iter   5 value -2.369400
## iter   6 value -2.374799
## iter   7 value -2.375178
## iter   8 value -2.375324
## iter   9 value -2.375353
## iter  10 value -2.375368
## iter  11 value -2.375448
## iter  12 value -2.375595
## iter  13 value -2.375646
## iter  14 value -2.375654
## iter  15 value -2.375709
## iter  16 value -2.375717
## iter  17 value -2.375735
## iter  18 value -2.375768
## iter  19 value -2.375866
## iter  20 value -2.376046
## iter  21 value -2.376347
## iter  22 value -2.376802
## iter  23 value -2.376958
```

```
## iter  24 value -2.376964
## iter  25 value -2.376966
## iter  26 value -2.376967
## iter  27 value -2.376969
## iter  28 value -2.376970
## iter  29 value -2.376974
## iter  30 value -2.376979
## iter  31 value -2.376990
## iter  32 value -2.377000
## iter  33 value -2.377004
## iter  34 value -2.377004
## iter  34 value -2.377004
## final  value -2.377004
## converged
## initial  value -2.434525
## iter   2 value -2.459257
## iter   3 value -2.468179
## iter   4 value -2.494510
## iter   5 value -2.503878
## iter   6 value -2.508202
## iter   7 value -2.508926
## iter   8 value -2.509454
## iter   9 value -2.509957
## iter  10 value -2.510582
## iter  11 value -2.512562
## iter  12 value -2.514211
## iter  13 value -2.516783
## iter  14 value -2.518564
## iter  15 value -2.518743
## iter  16 value -2.519428
## iter  17 value -2.519788
## iter  18 value -2.520018
## iter  19 value -2.520072
## iter  20 value -2.520124
## iter  21 value -2.520138
## iter  22 value -2.520181
## iter  23 value -2.520200
## iter  24 value -2.520200
## iter  25 value -2.520202
## iter  26 value -2.520206
## iter  27 value -2.520206
## iter  28 value -2.520206
## iter  28 value -2.520206
## iter  28 value -2.520206
## final  value -2.520206
## converged
```




```r
#ADF test
adf.test(modelA$fit$residuals)
```

```
## Warning in adf.test(modelA$fit$residuals): p-value smaller than printed p-
## value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  modelA$fit$residuals
## Dickey-Fuller = -6.7067, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
adf.test(modelB$fit$residuals)
```

```
## Warning in adf.test(modelB$fit$residuals): p-value smaller than printed p-
## value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  modelB$fit$residuals
## Dickey-Fuller = -6.1312, Lag order = 7, p-value = 0.01
## alternative hypothesis: stationary
```

```r
#Redundancy check
summary(modelA$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      ma2     sma1
##       -0.3336  -0.5012  -0.1053  -1.0000
## s.e.   0.2761   0.2810   0.2206   0.0269
##
## sigma^2 estimated as 0.006124:  log likelihood = 369.79,  aic = -729.59
##
## Training set error measures:
```

```
## Warning in trainingaccuracy(f, test, d, D): test elements must be within
## sample
```

```
##               ME RMSE MAE MPE MAPE
## Training set NaN  NaN NaN NaN  NaN
```

```r
summary(modelB$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     ma2      ma3     sma1
##       0.7351  -1.6578  0.8376  -0.1798  -1.0000
## s.e.  0.0862   0.0990  0.1158   0.0527   0.0277
##
## sigma^2 estimated as 0.005603:  log likelihood = 381.04,  aic = -750.08
##
## Training set error measures:
```

```
## Warning in trainingaccuracy(f, test, d, D): test elements must be within
## sample
```

```
##               ME RMSE MAE MPE MAPE
## Training set NaN  NaN NaN NaN  NaN
```

```r
#BIC & AIC
BIC(modelA$fit)
```

```
## [1] -710.3552
```

```r
BIC(modelB$fit)
```

```
## [1] -726.9988
```

```r
AIC(modelA$fit)
```

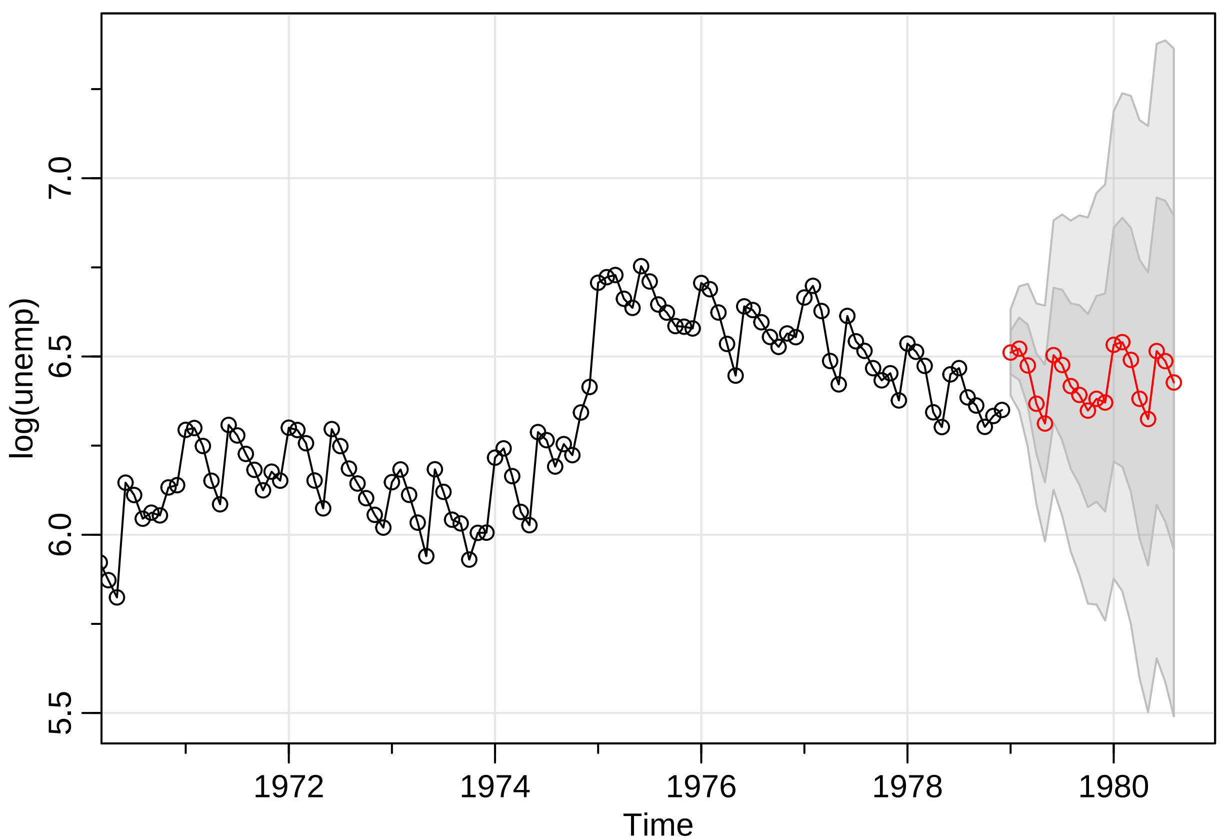
```
## [1] -729.5874
```

```r
AIC(modelB$fit)
```

```
## [1] -750.0774
```

Analysis: Model B is better, that is $ARIMA(1,1,3)(0,1,1)_{12}$ thus model equation is $x_t(1-0.7351B) = w_t(1-1.6578B)(1+0.8376B^2)(1-0.1798B^3)(1-B^{12})$

```r
#Forecasting
sarima.for(log(unemp), 1,1,2,0,1,1,12, n.ahead = 20)
```




```
## $pred
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 1979 6.511037 6.521885 6.474852 6.367511 6.312002 6.503737 6.476216
## 1980 6.532803 6.540151 6.490560 6.381349 6.324472 6.515208 6.486955
##           Aug      Sep      Oct      Nov      Dec
```

```
## 1979 6.416888 6.392187 6.348212 6.381297 6.370862
## 1980 6.427093
##
## $se
##             Jan        Feb        Mar        Apr        May        Jun
## 1979 0.06026437 0.08744082 0.11445234 0.14051447 0.16533249 0.18882900
## 1980 0.32785034 0.34907398 0.37016138 0.39087701 0.41109021 0.43073737
##             Jul        Aug        Sep        Oct        Nov        Dec
## 1979 0.21102999 0.23200981 0.25186274 0.27068806 0.28858231 0.30563552
## 1980 0.44979658 0.46827113
```

5 Source Code

```r
knitr::opts_chunk$set(echo = TRUE)
set.seed(12345)
options(scipen = 999)
options(tinytex.verbose = TRUE)

library("tidyverse") #ggplot and dplyr
library("gridExtra") # combine plots
library("knitr") # for pdf
library("fpp2") #timeseries with autoplot and stuff
library("reshape2") #reshape the data
library("forecast") # for forecasting time series
library("kernlab") #gausspr function
library("astsa") #oil and gas dataset
library("TSA") #Q3
library("tseries")

# The palette with black:
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73",
                "#F0E442", "#0072B2", "#D55E00", "#CC79A7")

set.seed(12345)
x_t <- arima.sim(model = list(ar = c(0.8,-0.2,0.1)), n=1000)
actual_pacf_value <- pacf(x_t, plot = FALSE)$acf[3]
df <- data.frame(x_t = as.vector(x_t))
df$x_t_lag_1 <- lag(df$x_t,1)
df$x_t_lag_2 <- lag(df$x_t,2)
df$x_t_lag_3 <- lag(df$x_t,3)
df <- na.omit(df)

# building models and getting their residuals
model_1_res <- lm(x_t ~ x_t_lag_1 + x_t_lag_2, data = df)$residuals
model_2_res <- lm(x_t_lag_3 ~ x_t_lag_1 + x_t_lag_2, data = df)$residuals

# theortical pacf values
theotical_pacf_value <- cor(x = model_1_res, y = model_2_res, use = "na.or.complete")

cat("The theoretical and actual value of PACF are: ", theotical_pacf_value, actual_pacf_value)
```

```r
set.seed(12345)
x_t <- arima.sim(model = list(ar = c(0.8,0.1)), n=100)

method_yule_walker <- ar(x_t, order = 2, method = "yule-walker", aic = FALSE)$ar
method_cls <- ar(x_t, order = 2, method = "ols", aic = FALSE)$ar
method_mle <- ar(x_t, order = 2, method = "mle", aic = FALSE)$ar

df <- data.frame(rbind(method_yule_walker, method_cls,method_mle))

kable(df, caption = "Comparison of parameters using different methods")

# Since varience is not given by ar we use arima function
ML_Model_CI = arima(x_t, order = c(2,0,0), method = "ML")
sigma = ML_Model_CI$var.coef[2, 2]
phi_2 = ML_Model_CI$coef[2]
CI = c(phi_2 - 1.96 * sigma, phi_2 + 1.96 * sigma)
CI
set.seed(12345)
x_t <- arima.sim(model = list(ma = c(0.3,rep(0,10),0.6,0.18)), n=200)

df <- data.frame(sample_acf = acf(x_t, plot = FALSE, lag.max = 14)$acf,
                 sample_pacf = pacf(x_t, plot = FALSE, lag.max = 14)$acf,
                 theortical_acf = ARMAacf(ma = c(0.3,rep(0,10),0.6,0.18), pacf = FALSE, lag.max = 13),
                 theortical_pacf = ARMAacf(ma = c(0.3,rep(0,10),0.6,0.18), pacf = TRUE, lag.max = 14))

df$index <- rownames(df)

plot1 <- ggplot(data=df, aes(x=index)) +
  geom_col(aes(y=sample_acf)) +
  ggtitle("Sample ACF")

plot2 <- ggplot(data=df, aes(x=index)) +
  geom_col(aes(y=theortical_acf)) +
  ggtitle("Theoretical ACF")

grid.arrange(plot1, plot2, ncol = 1)

plot3 <- ggplot(data=df, aes(x=index)) +
  geom_col(aes(y=sample_pacf)) +
  ggtitle("Sample PACF")

plot4 <- ggplot(data=df, aes(x=index)) +
  geom_col(aes(y=theortical_pacf)) +
  ggtitle("Theoretical PACF")

grid.arrange(plot3, plot4, ncol = 1)

set.seed(12345)
x_t <- arima.sim(model = list(ma = c(0.3,rep(0,10),0.6,0.18)), n=200)
fit_x_t <- arima(x_t, order = c(0,0,1), seasonal = list(order = c(0,0,1),period = 12))
predicted_x_t <- predict(fit_x_t, n.ahead=30)
predicted_x_t_upper_band <- predicted_x_t$pred + 1.96 * predicted_x_t$se
predicted_x_t_lower_band <- predicted_x_t$pred - 1.96 * predicted_x_t$se
```

```r
#kernlab

df <- data.frame(y = x_t)
df$x <- as.numeric(rownames(df))
gausspr_model <- gausspr(x=df$x, y=df$y)
predicted_x_t_kernlab <- predict(gausspr_model, newdata=data.frame(x=201:230))

df3 <- data.frame(y = predicted_x_t_kernlab, x=201:230)


df2 <- data.frame(predicted_x_t = predicted_x_t$pred,
                  predicted_x_t_upper = predicted_x_t_upper_band,
                  predicted_x_t_lower = predicted_x_t_lower_band,
                  x = 201:230)

ggplot() +
  geom_line(data=df, aes(x=x, y=y, color="Actual y")) +
    geom_line(data=df2, aes(x=x, y=predicted_x_t, color="Predicted y")) +
      geom_line(data=df2, aes(x=x, y=predicted_x_t_upper, color="Upper band")) +
        geom_line(data=df2, aes(x=x, y=predicted_x_t_lower, color="Lower band")) +
      scale_colour_manual("", breaks = c("Actual y", "Predicted y", "Upper band", "Lower band"),
                        values = c("#000000", "#009E73", "#56B4E9", "#E69F00")) +
  ggtitle("Original vs. Predicted y with confidence bands")


ggplot() +
  geom_line(data=df, aes(x=x, y=y, color="Actual y")) +
    geom_line(data=df3, aes(x=x, y=y, color="Predicted y")) +
        scale_colour_manual("", breaks = c("Actual y", "Predicted y"),
                        values = c("#000000","#56B4E9")) +
  ggtitle("Original vs. Predicted y using gausspr")


x_t <- arima.sim(model = list(ar = c(0.7), ma=c(0.5)), n=50)
fit_x_t <- arima(x_t[1:40], order = c(1,0,1), include.mean = 0)

predicted_x_t <- predict(fit_x_t, n.ahead=10)
predicted_x_t_upper_band <- predicted_x_t$pred + 1.96 * predicted_x_t$se
predicted_x_t_lower_band <- predicted_x_t$pred - 1.96 * predicted_x_t$se

df <- data.frame(y = x_t[1:40], x=1:40)
df2 <- data.frame(y = predicted_x_t$pred,
                  upper_band=predicted_x_t_upper_band,
                  lower_band=predicted_x_t_lower_band,
                  x = 41:50)

ggplot() +
  geom_line(data=df, aes(x=x, y=y, color="Actual y")) +
    geom_line(data=df2, aes(x=x, y=y, color="Predicted y")) +
      geom_line(data=df2, aes(x=x, y=upper_band, color="Upper band")) +
        geom_line(data=df2, aes(x=x, y=lower_band, color="Lower band")) +
      scale_colour_manual("", breaks = c("Actual y", "Predicted y", "Upper band", "Lower band"),
                        values = c("#000000", "#009E73", "#56B4E9", "#E69F00")) +
```

```r
  ggtitle("Original vs. Predicted y with confidence bands")
set.seed(12345)

plot_acf_pacf <- function(df){
acf_df <- acf(df, plot = FALSE, lag.max = 40)$acf
pacf_df <- pacf(df, plot = FALSE, lag.max = 40)$acf
acf_diff_df <- acf(diff(df), plot = FALSE, lag.max = 40)$acf
pacf_diff_df <- pacf(diff(df), plot = FALSE, lag.max = 40)$acf

df <- data.frame(acf_df=acf_df,
                 pacf_df=pacf_df,
                 acf_diff_df=acf_diff_df,
                 pacf_diff_df=pacf_diff_df,
                 x=1:length(pacf_diff_df))


plot1 <- ggplot(data=df, aes(x=x)) +
  geom_col(aes(y=acf_df)) +
  ggtitle("ACF")

plot2 <- ggplot(data=df, aes(x=x)) +
  geom_col(aes(y=pacf_df)) +
  ggtitle("PACF")

plot3 <- ggplot(data=df, aes(x=x)) +
  geom_col(aes(y=acf_diff_df)) +
  ggtitle("ACF with 1 diff")

plot4 <- ggplot(data=df, aes(x=x)) +
  geom_col(aes(y=pacf_diff_df)) +
  ggtitle("PACF with 1 diff")


return(grid.arrange(plot1, plot2, plot3, plot4,  nrow = 2,ncol = 2))
}

plot_acf_pacf(df=chicken)

set.seed(12345)
plot_acf_pacf(df=so2)
plot_acf_pacf(df=EQcount)
plot_acf_pacf(df=HCT)
set.seed(12345)

# visualization
autoplot(ts(oil, start = 2000, frequency = 52)) +
           ylab("Price of Oil") +xlab("Year") +
           ggtitle("Price of Oil vs. Years")

ggAcf(oil) + ggtitle("ACF for Oil")
ggAcf(diff(oil)) + ggtitle("ACF for Oil with one diff")
ggPacf(oil) + ggtitle("PACF for Oil")
ggPacf(diff(oil)) + ggtitle("PACF for Oil with one diff")
```

```r
# with log
autoplot(ts(log(oil), start = 2000, frequency = 52)) +
          ylab("Price of Oil in Log") +xlab("Year") +
          ggtitle("Price of Log Oil vs. Years")


autoplot(ts(diff(log(oil), lag=1), start = 1948, frequency = 12)) +
          ylab("# Log Oil") +xlab("Year") +
          ggtitle("Price of log oil with one lags vs. Years")

ggAcf(log(oil)) + ggtitle("ACF for log Oil")
ggAcf(diff(log(oil))) + ggtitle("ACF for log Oil with one diff")
ggPacf(log(oil)) + ggtitle("PACF for log Oil")
ggPacf(diff(log(oil))) + ggtitle("PACF for log Oil with one diff")


# EACF
eacf(diff(log(oil)))


#Suggested Models
modelA <- sarima(log(oil), 0,1,1)
modelB <- sarima(log(oil), 1,1,1)
modelC <- sarima(log(oil), 0,1,3)

#ADF test
adf.test(modelA$fit$residuals)
adf.test(modelB$fit$residuals)
adf.test(modelC$fit$residuals)

#Redundancy check
summary(modelA$fit)
summary(modelB$fit)
summary(modelC$fit)

#BIC
BIC(modelA$fit)
BIC(modelB$fit)
BIC(modelC$fit)

#AIC
AIC(modelA$fit)
AIC(modelB$fit)
AIC(modelC$fit)

#Model C is the best

#Forecasting
sarima.for(log(oil), 0,1,3, n.ahead = 20)


set.seed(12345)
```

```r
# visualization with log
autoplot(ts(log(unemp), start = 1948, frequency = 12)) +
           ylab("# Log Unempoyment") +xlab("Year") +
           ggtitle("# Unempoyment in log vs. Years")

ggAcf(log(unemp)) + ggtitle("ACF for Log Unempoyment")
ggPacf(log(unemp)) + ggtitle("PACF for Log Unempoyment")


ggAcf(diff(log(unemp))) + ggtitle("ACF for Log Unempoyment with one diff")
ggPacf(diff(log(unemp))) + ggtitle("PACF for Log Unempoyment with one diff")

# visualization with log
autoplot(ts(diff(diff(log(unemp)), lag=12), start = 1948, frequency = 12)) +
           ylab("# Log Unempoyment") +xlab("Year") +
           ggtitle("# Unempoyment in log with tweleve lags vs. Years")

ggAcf(diff(diff(log(unemp)), lag=12)) + ggtitle("ACF for Log Unempoyment with tweleve diff")
ggPacf(diff(diff(log(unemp)), lag=12)) + ggtitle("PACF for Log Unempoyment with tweleve diff")

eacf(diff(diff(log(unemp)), lag=12))

#Suggested Models
modelA <- sarima(diff(diff(log(unemp)), lag=12), 1,1,2,0,1,1,12)
modelB <- sarima(diff(diff(log(unemp)), lag=12), 1,1,3,0,1,1,12)


#ADF test
adf.test(modelA$fit$residuals)
adf.test(modelB$fit$residuals)

#Redundancy check
summary(modelA$fit)
summary(modelB$fit)

#BIC & AIC
BIC(modelA$fit)
BIC(modelB$fit)
AIC(modelA$fit)
AIC(modelB$fit)

#Forecasting
sarima.for(log(unemp), 1,1,2,0,1,1,12, n.ahead = 20)
```