Time Series Final Project: Total Nonfarm Job Openings (U.S., Monthly)

Due: August 12, 2024 @ 5:59pm Central

Section: 1

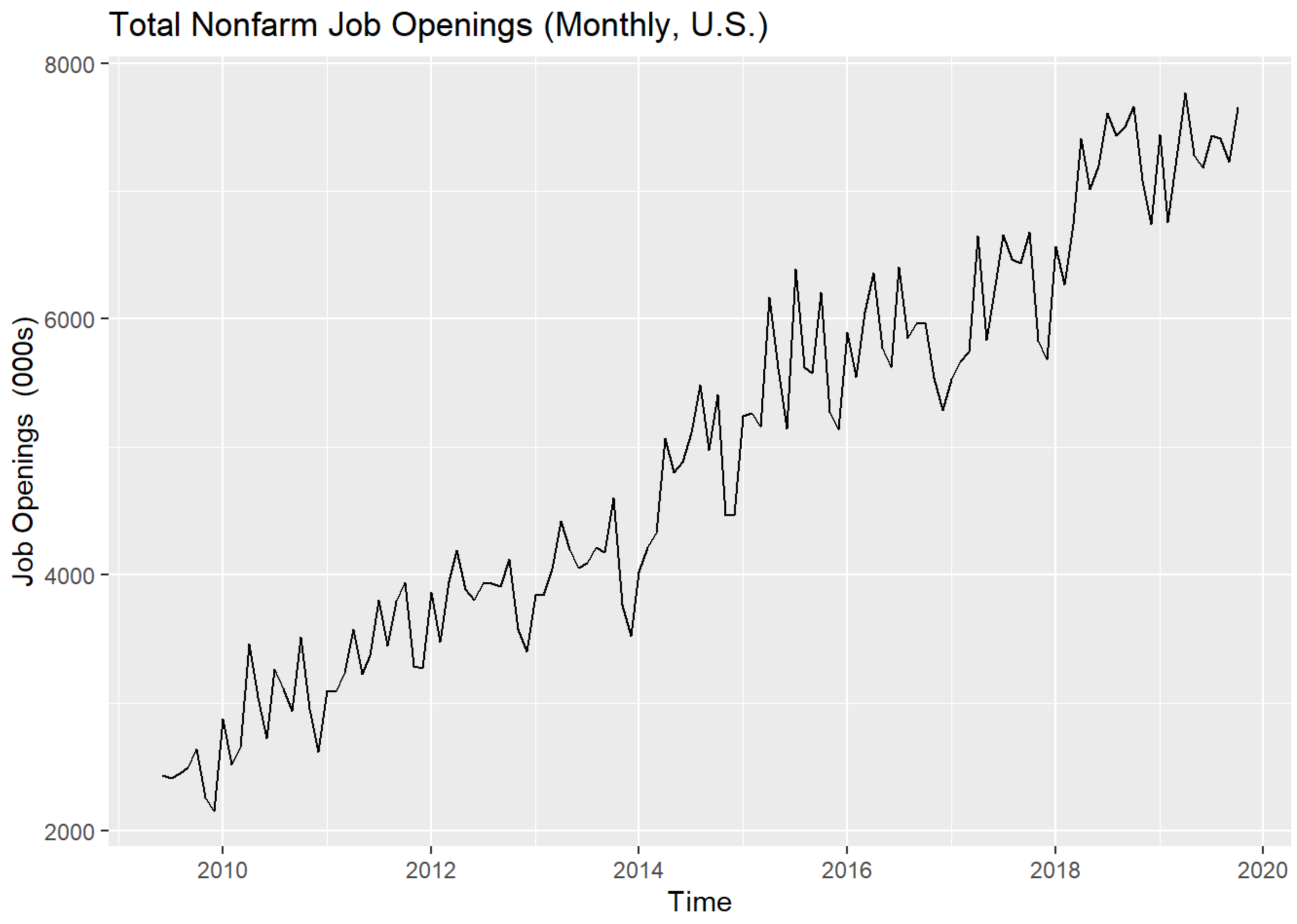
Students: Alex Foster, Natalie Kim, Mathew Spencer, Danae Vassiliadis

Data Sources

- job openings (in thousands): https://fred.stlouisfed.org/series/JTUJOL (https://fred.stlouisfed.org/series/JTUJOL)
- advance retail sales (in millions of USD): https://fred.stlouisfed.org/series/RSXFSN (https://fred.stlouisfed.org/series/RSXFSN)
- consumer sentiment from University of Michican Survey: https://fred.stlouisfed.org/series/UMCSENT (https://fred.stlouisfed.org/series/UMCSENT)

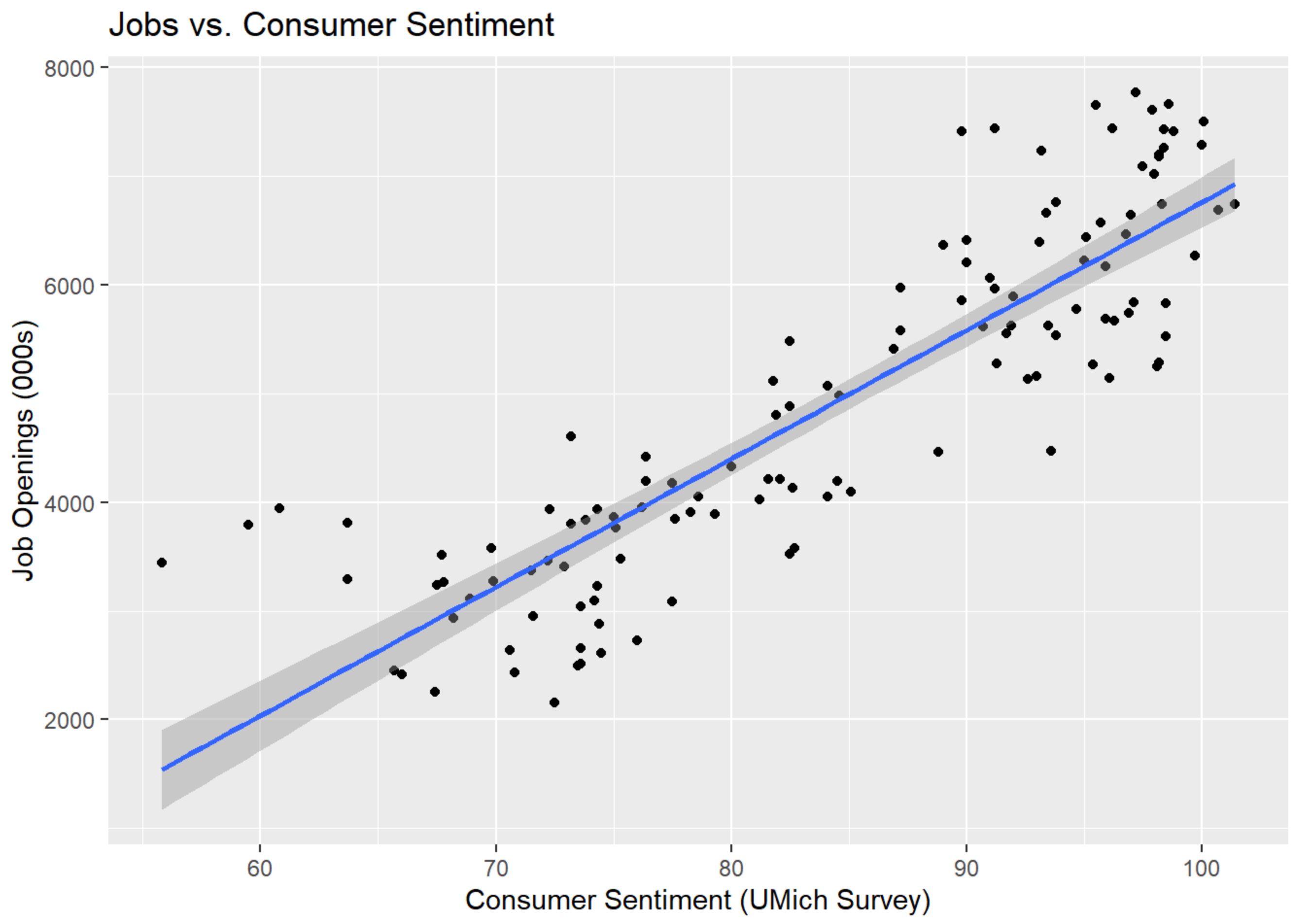
Step 1: load and plot data

```
# load data
data <- read_csv("data.csv", show_col_types = FALSE)

# convert data to time series objects
jobs <- ts(data$jobs, start=c(2000,12), frequency=12)
sentiment <- ts(data$sentiment, start=c(2000,12), frequency=12)
retail <- ts(data$retail, start=c(2000,12), frequency=12)

# merge into single time series object
data <- ts(cbind(jobs, sentiment, retail), start=c(2000,12), frequency=12)

# focus on time between recessions (June 2009 through October 2019)
data <- window(data, start=c(2009,6), end=c(2019,10))

# plot jobs
autoplot(data[,'jobs']) +
  ggtitle('Total Nonfarm Job Openings (Monthly, U.S.)') +
  ylab('Job Openings  (000s)')
```



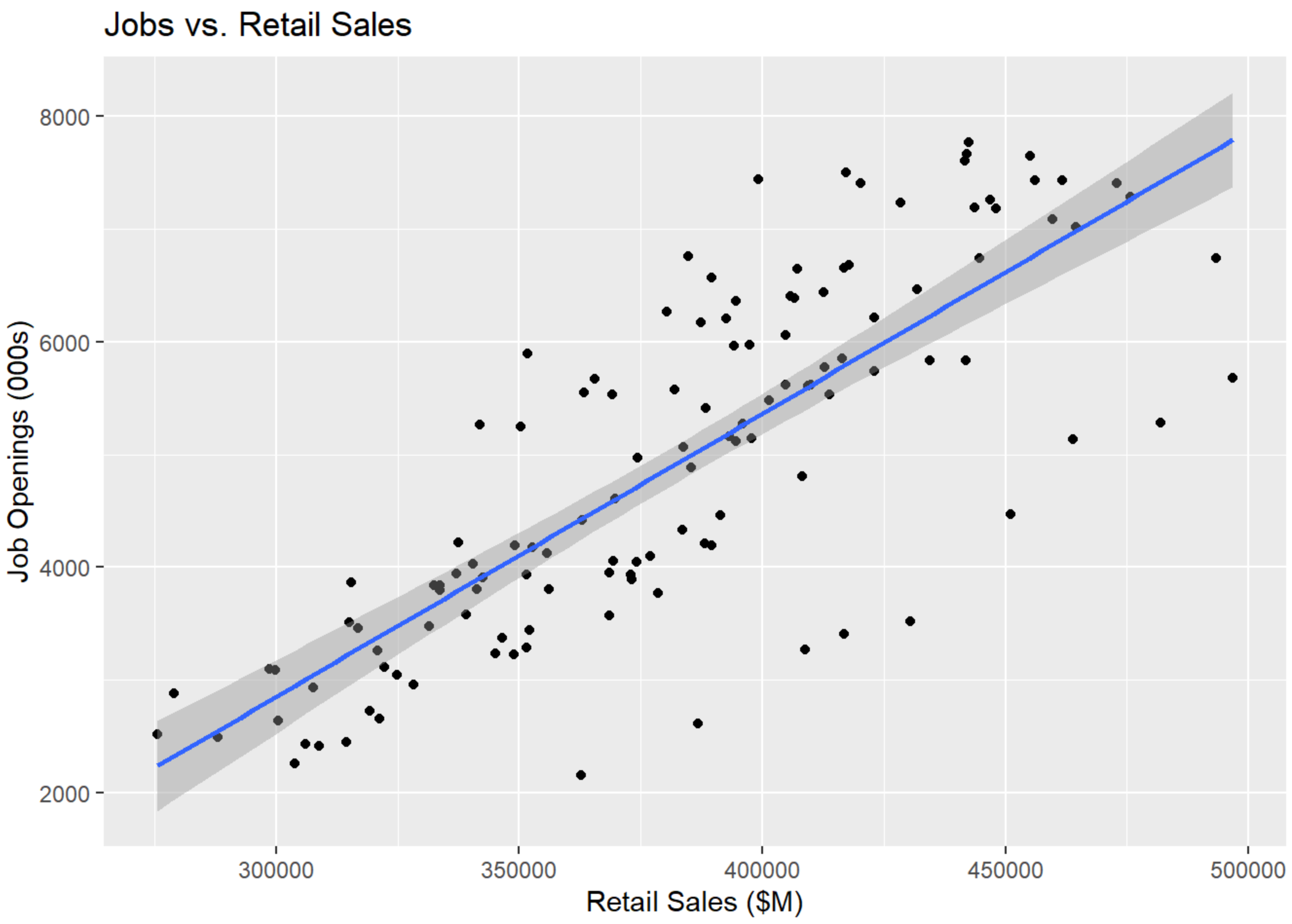
```
# plot jobs vs. potential external regressors
ggplot(as.data.frame(data), aes(x=sentiment, y=jobs))+
  geom_point()+
  geom_smooth(method='lm')+ # add linear trendline
  ggtitle('Jobs vs. Consumer Sentiment')+
  xlab('Consumer Sentiment (UMich Survey)')+
  ylab('Job Openings (000s)')
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



```
ggplot(as.data.frame(data), aes(x=retail, y=jobs))+
  geom_point()+
  geom_smooth(method='lm')+ # add linear trendline
  ggtitle('Jobs vs. Retail Sales')+
  xlab('Retail Sales ($M)')+
  ylab('Job Openings (000s)')
```

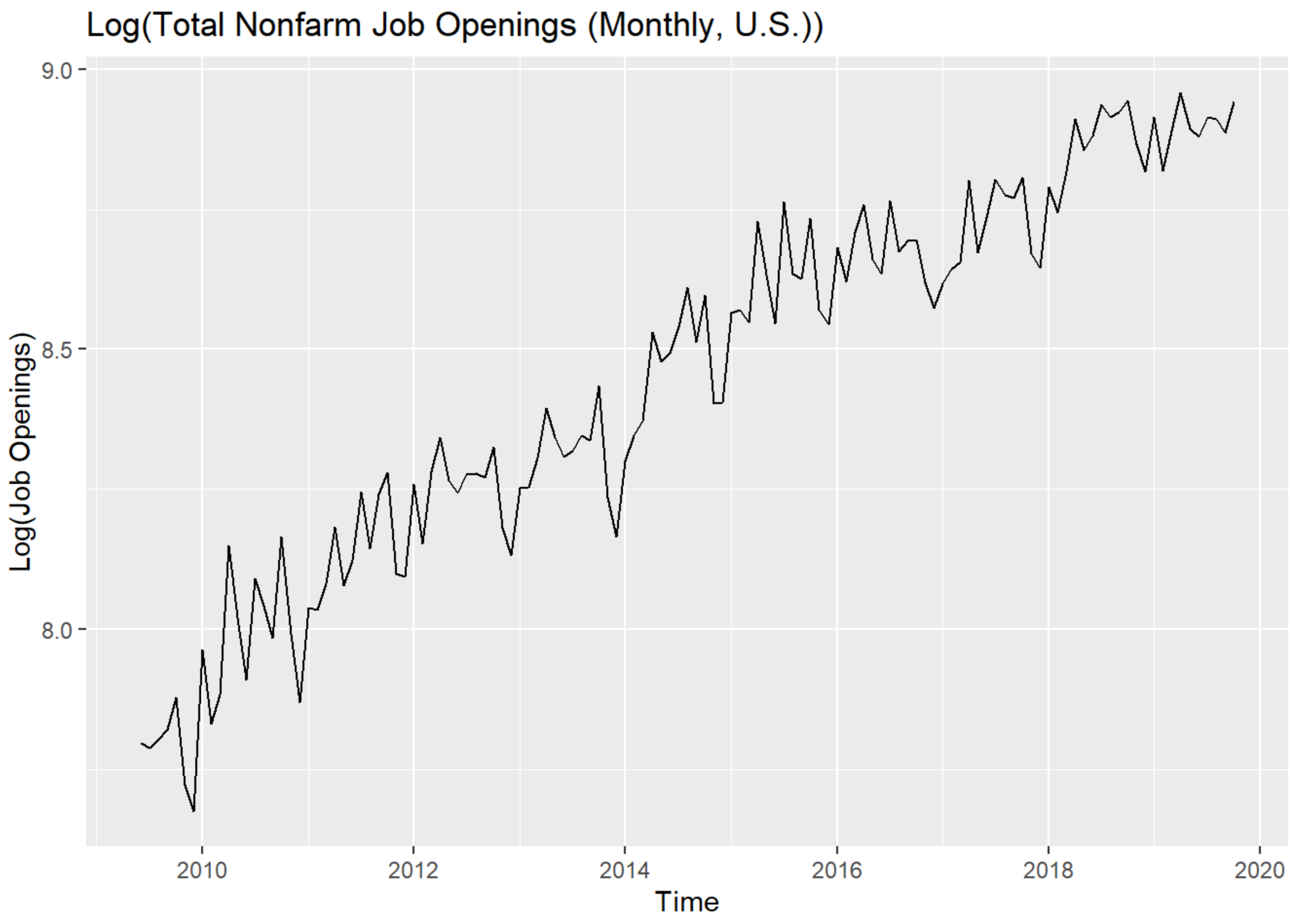
```
## `geom_smooth()` using formula = 'y ~ x'
```



Comment: The plots above show that our time series has seasonality and generally has a positive trend. Specifically, we tend to see a dip in job openings at the end of the year, with higher levels in the middle of the year.

Step 2: transform and difference the data

```
# box-cox transform to stabilize variance
lambda <- BoxCox.lambda(data) # lambda = 2.00 --> implies data is already fairly close t
o stable
lambda <- 0 # will opt for log transform since variance APPEARS to be multiplicative
data_trans <- BoxCox(data, lambda=0)

autoplot(data_trans[,'jobs']) +
  ggtitle('Log(Total Nonfarm Job Openings (Monthly, U.S.))') +
  ylab('Log(Job Openings)')
```



```
# check for stationarity
# for KPSS test: small p-value -> reject Ho -> data is non-stationary
# for ADF test: small p-value -> reject Ho -> data is stationary

# raw data
kpss_raw <- kpss.test(data_trans[,'jobs'])
```

```
## Warning in kpss.test(data_trans[, "jobs"]): p-value smaller than printed
## p-value
```

```
adf_raw <- adf.test(data_trans[,'jobs'])
print('********************************')
```

```
## [1] "********************************"
```

```
print('***KPSS Test for raw data***')
```

```
## [1] "***KPSS Test for raw data***"
```

```
print(kpss_raw) # small p-value -> reject Ho -> data is non-stationary
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  data_trans[, "jobs"]
## KPSS Level = 2.512, Truncation lag parameter = 4, p-value = 0.01
```

```
print('********************************')
```

```
## [1] "********************************"
```

```
print('***ADF Test for raw data***')
```

```
## [1] "***ADF Test for raw data***"
```

```
print(adf_raw) # large p-value -> fail to reject Ho -> data is non-stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data_trans[, "jobs"]
## Dickey-Fuller = -3.7603, Lag order = 4, p-value = 0.02321
## alternative hypothesis: stationary
```

```
# apply seasonal differencing
data_diff_s <- diff(data_trans[,'jobs'], lag=12)
kpss_diff_s <- kpss.test(data_diff_s)
adf_diff_s <- adf.test(data_diff_s)
print('********************************')
```

```
## [1] "********************************"
```

```
print('***KPSS Test for Seasonal Diff. Data***')
```

```
## [1] "***KPSS Test for Seasonal Diff. Data***"
```

```
print(kpss_diff_s) # large p-value -> accept Ho -> data is stationary
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  data_diff_s
## KPSS Level = 0.48993, Truncation lag parameter = 4, p-value = 0.04393
```

```
print('********************************')
```

```
## [1] "********************************"
```

```
print('***ADF Test for seasonal diff. data***')
```

```
## [1] "***ADF Test for seasonal diff. data***"
```

```
print(adf_diff_s) # small p-value -> reject Ho -> data is stationary
```

```
## 
##  Augmented Dickey-Fuller Test
## 
## data:  data_diff_s
## Dickey-Fuller = -2.7635, Lag order = 4, p-value = 0.2598
## alternative hypothesis: stationary
```

```
# ADF and KPSS tests suggest we only need seasonal differencing
# but the ACF plots don't pass the eye test, so we'll apply first-order differencing als
o
# apply first-order differencing
data_diff2 <- diff(data_diff_s)
kpss_diff2 <- kpss.test(data_diff2)
```

```
## Warning in kpss.test(data_diff2): p-value greater than printed p-value
```

```
adf_diff2 <- adf.test(data_diff2)
```

```
## Warning in adf.test(data_diff2): p-value smaller than printed p-value
```

```
print('********************************')
```

```
## [1] "********************************"
```

```
print('***KPS Test for 2nd diff. data**')
```

```
## [1] "***KPS Test for 2nd diff. data**"
```

```
print(kpss_diff2)
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  data_diff2
## KPSS Level = 0.044917, Truncation lag parameter = 4, p-value = 0.1
```

```
print('********************************')
```

```
## [1] "********************************"
```

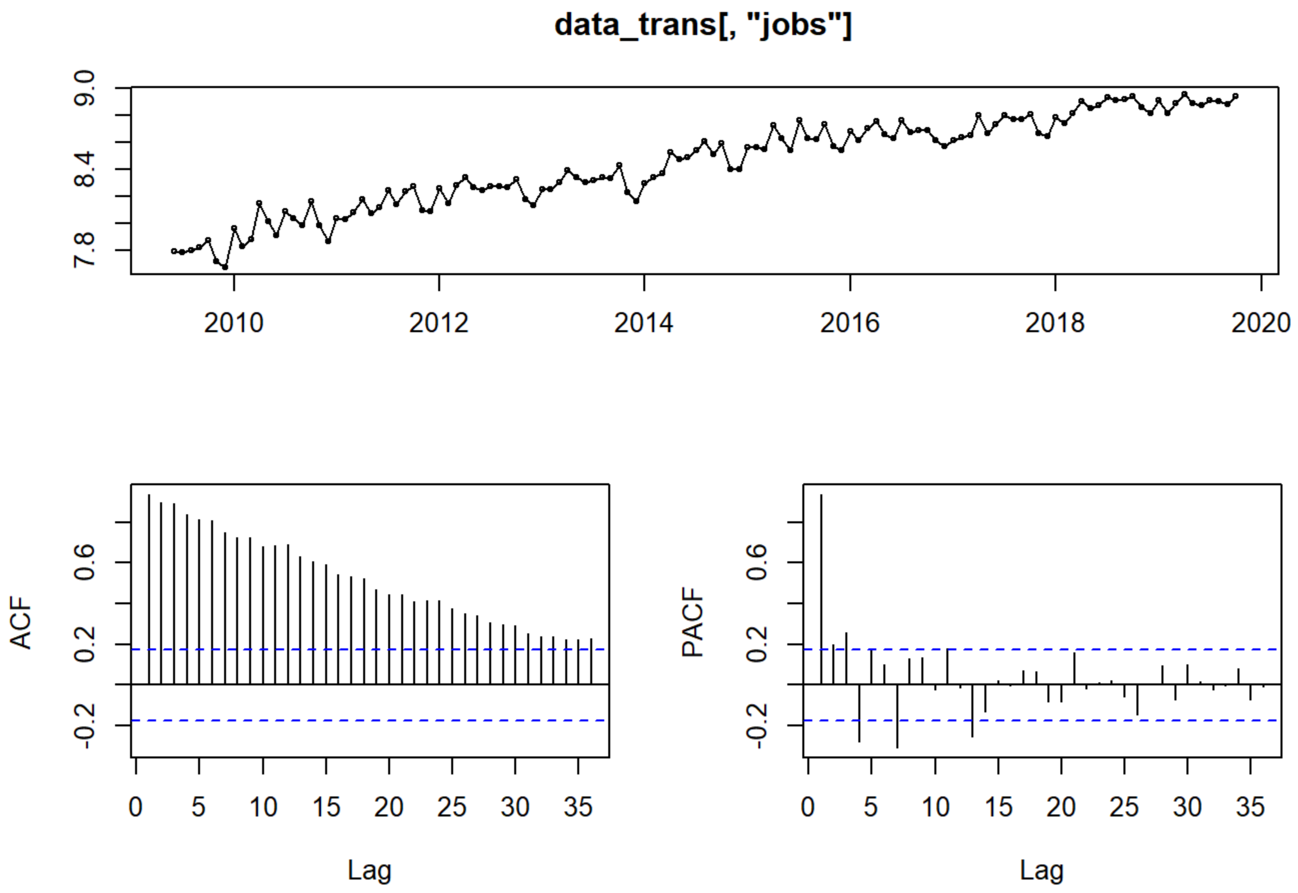
```
print('***ADF Test for 2nd diff. data**')
```

```
## [1] "***ADF Test for 2nd diff. data**"
```

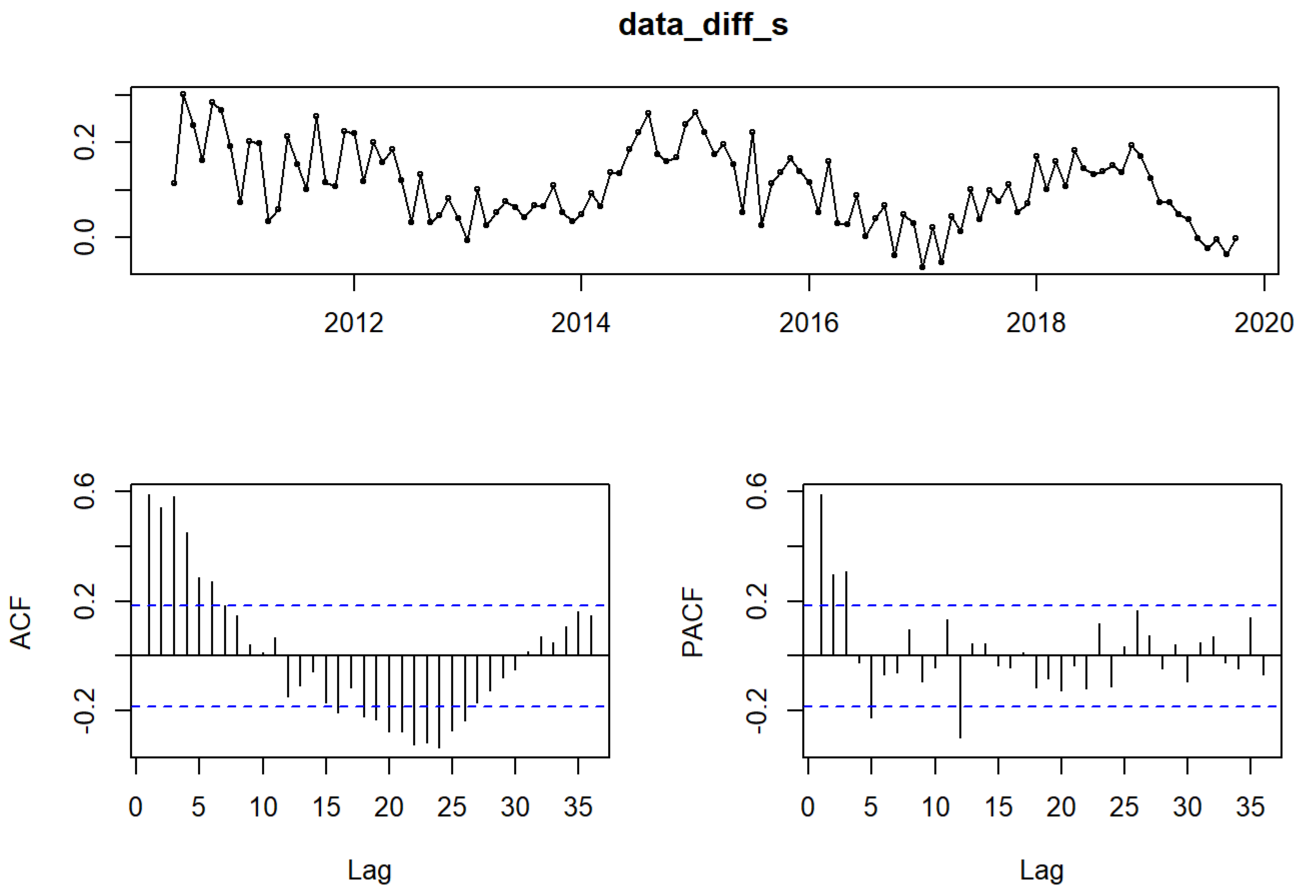
```
print(adf_diff2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data_diff2
## Dickey-Fuller = -4.8074, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

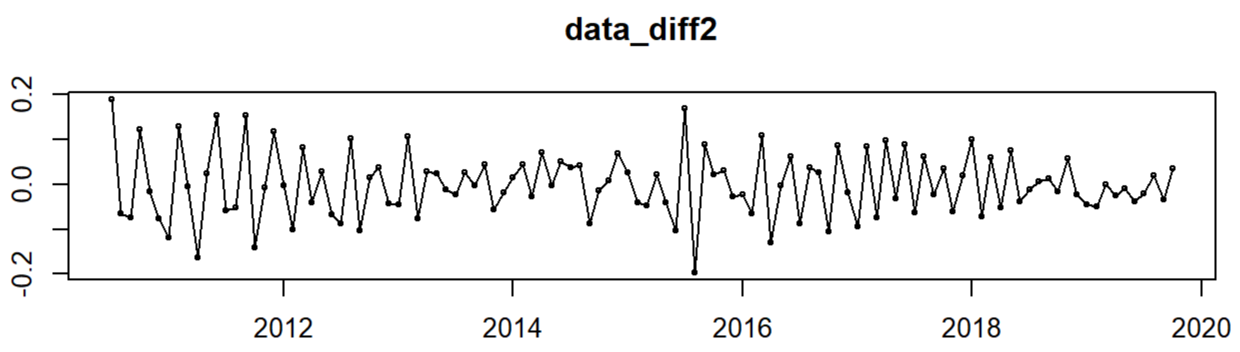
```
tsdisplay(data_trans[,'jobs'])
```

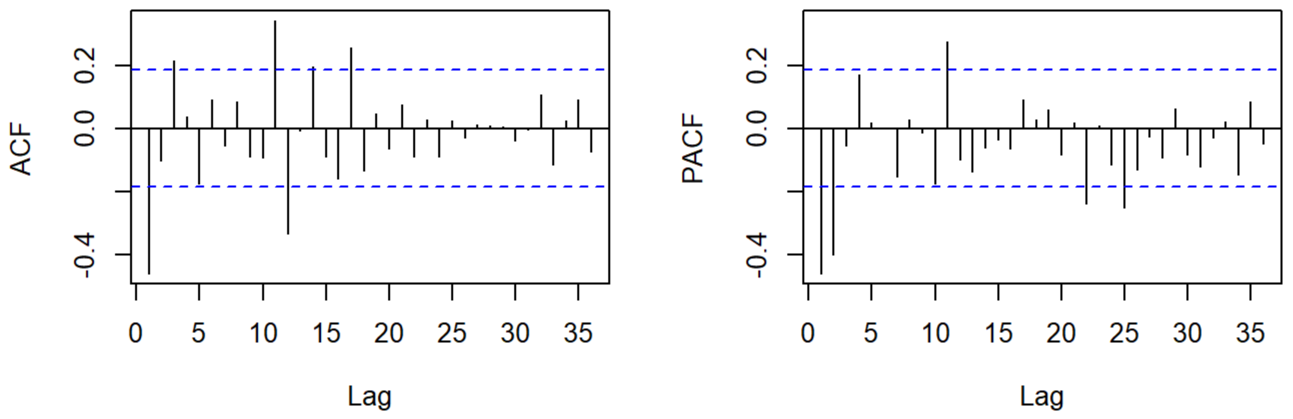


```
tsdisplay(data_diff_s)
```



```
tsdisplay(data_diff2)
```





Step 3: split into train and test data

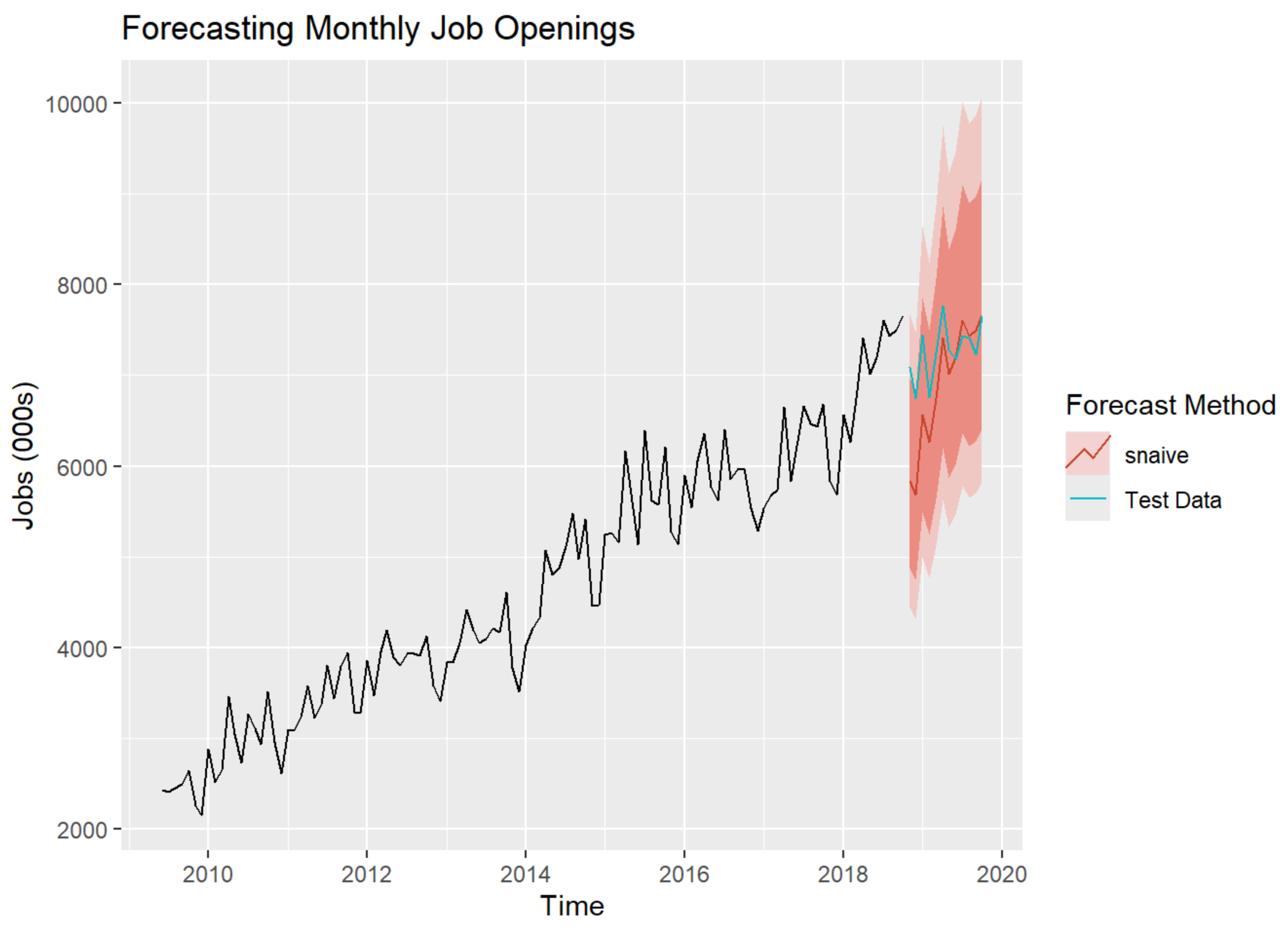
```
# save last 12 months as test data
train_end <- c(2018,10)
test_start <- c(2018,11)
train_data <- window(data, end=train_end)
test_data <- window(data, start=test_start)

jobs_train <- train_data[,'jobs']
sentiment_train <- train_data[,'sentiment']
retail_train <- train_data[,'retail']

jobs_test <-test_data[,'jobs']
sentiment_test <- test_data[,'sentiment']
retail_test <- test_data[,'retail']
```

Step 4: build models

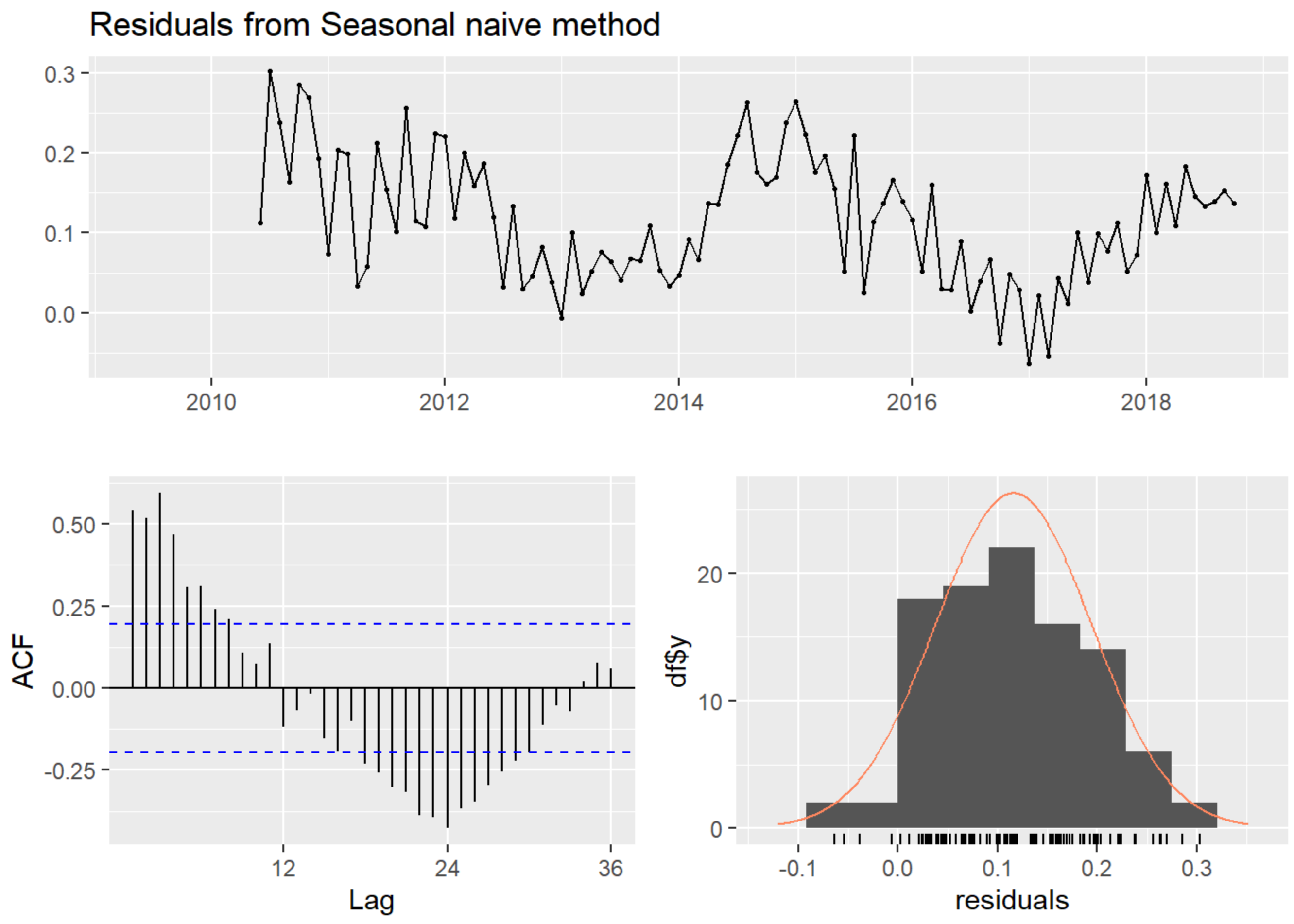
We'll build several models without any external regressors

Step 4a: create a baseline model with snaive()

```
snaive_model <- snaive(jobs_train, lambda=lambda)
snaive_forecast <- forecast(snaive_model, h=12)

autoplot(jobs_train) +
  autolayer(snaive_forecast, series='snaive') +
  autolayer(jobs_test, series='Test Data') +
  ggtitle('Forecasting Monthly Job Openings') +
  ylab('Jobs (000s)') +
  guides(colour=guide_legend(title='Forecast Method'))
```



```
checkresiduals(snaive_model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from Seasonal naive method
## Q* = 246.72, df = 23, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 23
```

```
# function to calc RMSE
calculate_rmse <- function(actual, forecast){
  sqrt(mean((actual-forecast)^2, na.rm=TRUE))
}

# function to calc MAPE
calculate_mape <- function(actual, forecast){
  mean(abs((actual - forecast) / actual)) * 100
}

# calc forecast RMSE and MAPE
rmse_snaive <- calculate_rmse(jobs_test, snaive_forecast$mean)
cat('Test RMSE for Seasonal Naive model: ', rmse_snaive)
```

```
## Test RMSE for Seasonal Naive model:  596.6854
```

```
mape_snaive <- calculate_mape(jobs_test, snaive_forecast$mean)
cat('Test MAPE for Seaonal Naive Model: ', mape_snaive)
```

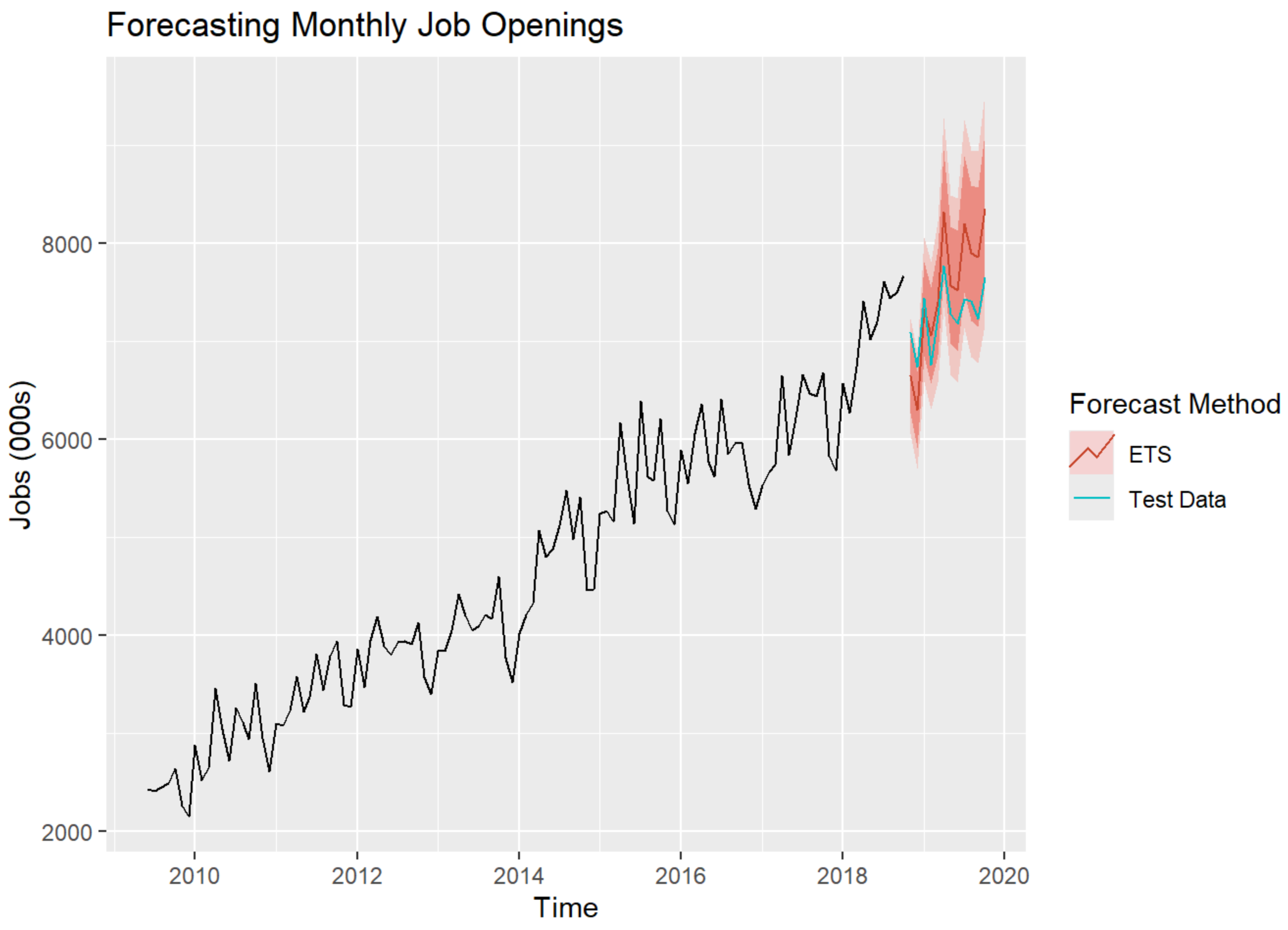
```
## Test MAPE for Seaonal Naive Model:  6.221725
```

Step 4b: ETS Model

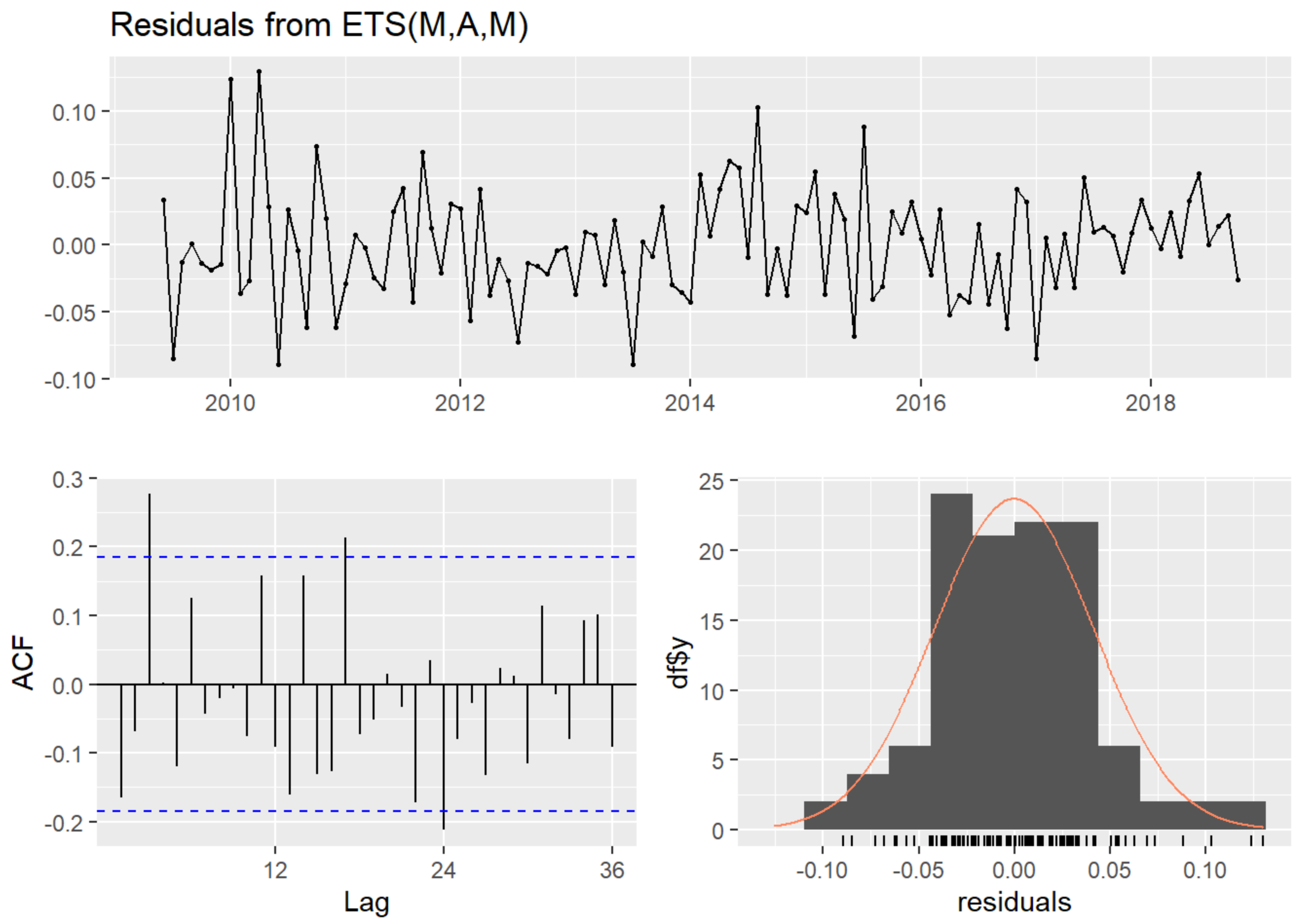
```
ets_model <- ets(jobs_train)
summary(ets_model) # for Ljung Box Test: small p-value <- reject Ho <- residuals show au
tocorrelation
```

```
## ETS(M,A,M)
##
## Call:
##  ets(y = jobs_train)
##
##   Smoothing parameters:
##     alpha = 0.391
##     beta  = 1e-04
##     gamma = 1e-04
##
##   Initial states:
##     l = 2336.7697
##     b = 43.5425
##     s = 1.0029 1.1075 0.9911 0.9516 0.993 0.8586
##            0.9121 1.0749 1.0175 1.0278 1.0733 0.9897
##
##   sigma:  0.0449
##
##      AIC     AICc      BIC
## 1748.176 1754.618 1794.541
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.9164944 187.0803 147.3232 -0.2102109 3.262021 0.2786084
##                    ACF1
## Training set -0.1646604
```

```
ets_forecast <- forecast(ets_model, h=12)

autoplot(jobs_train) +
  autolayer(ets_forecast, series='ETS') +
  autolayer(jobs_test, series='Test Data') +
  ggtitle('Forecasting Monthly Job Openings') +
  ylab('Jobs (000s)') +
  guides(colour=guide_legend(title='Forecast Method'))
```



```
checkresiduals(ets_model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,A,M)
## Q* = 44.487, df = 23, p-value = 0.004591
##
## Model df: 0.   Total lags used: 23
```

```
# calc forecast RMSE
rmse_ets <- calculate_rmse(jobs_test, ets_forecast$mean)
cat('Test RMSE for ETS model: ', rmse_ets)
```

```
## Test RMSE for ETS model:  476.0594
```

```
mape_ets <- calculate_mape(jobs_test, ets_forecast$mean)
cat('Test MAPE for ETS Model: ', mape_ets)
```

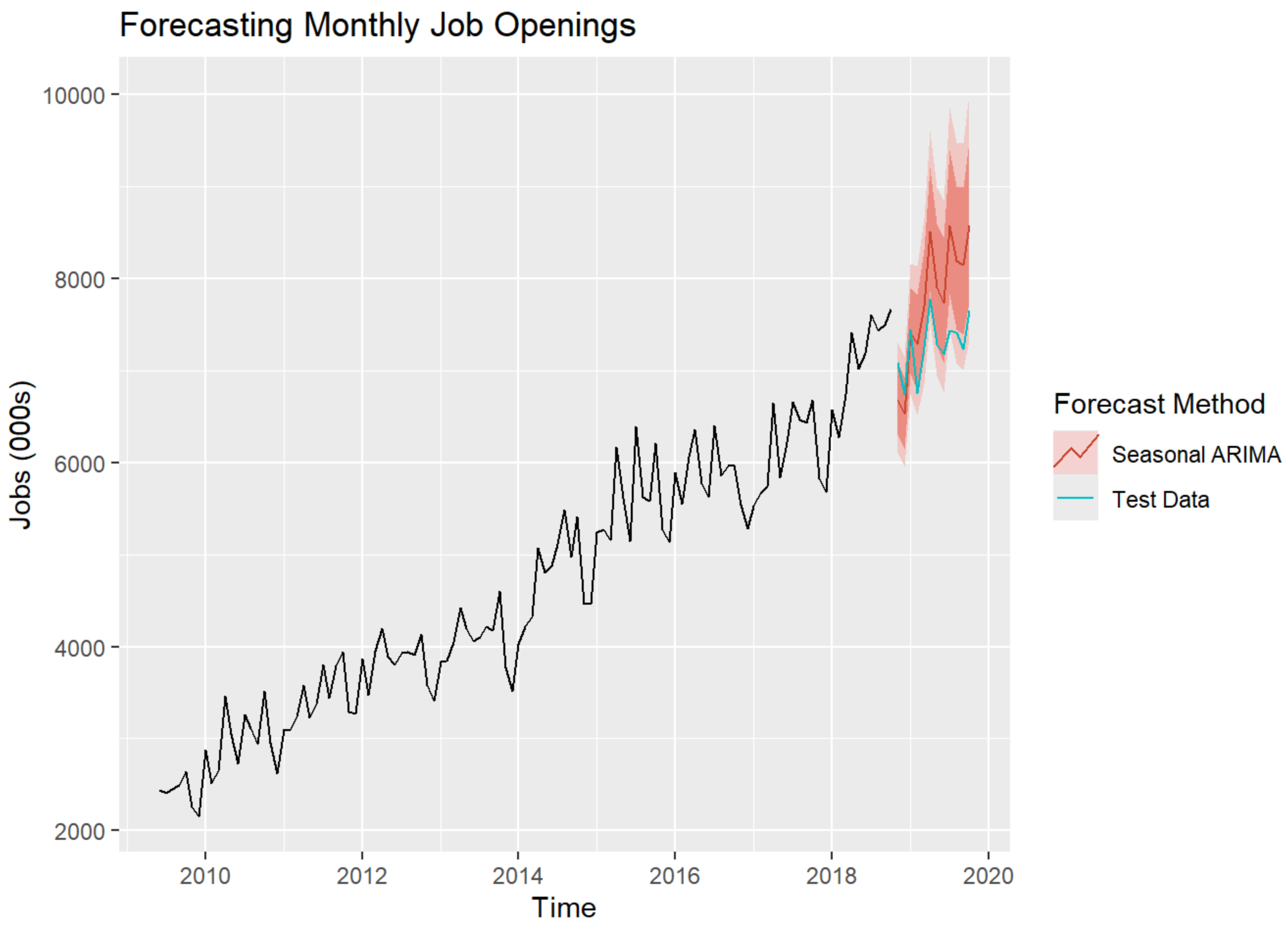
```
## Test MAPE for ETS Model:  5.93344
```

Step 4c: ARIMA model

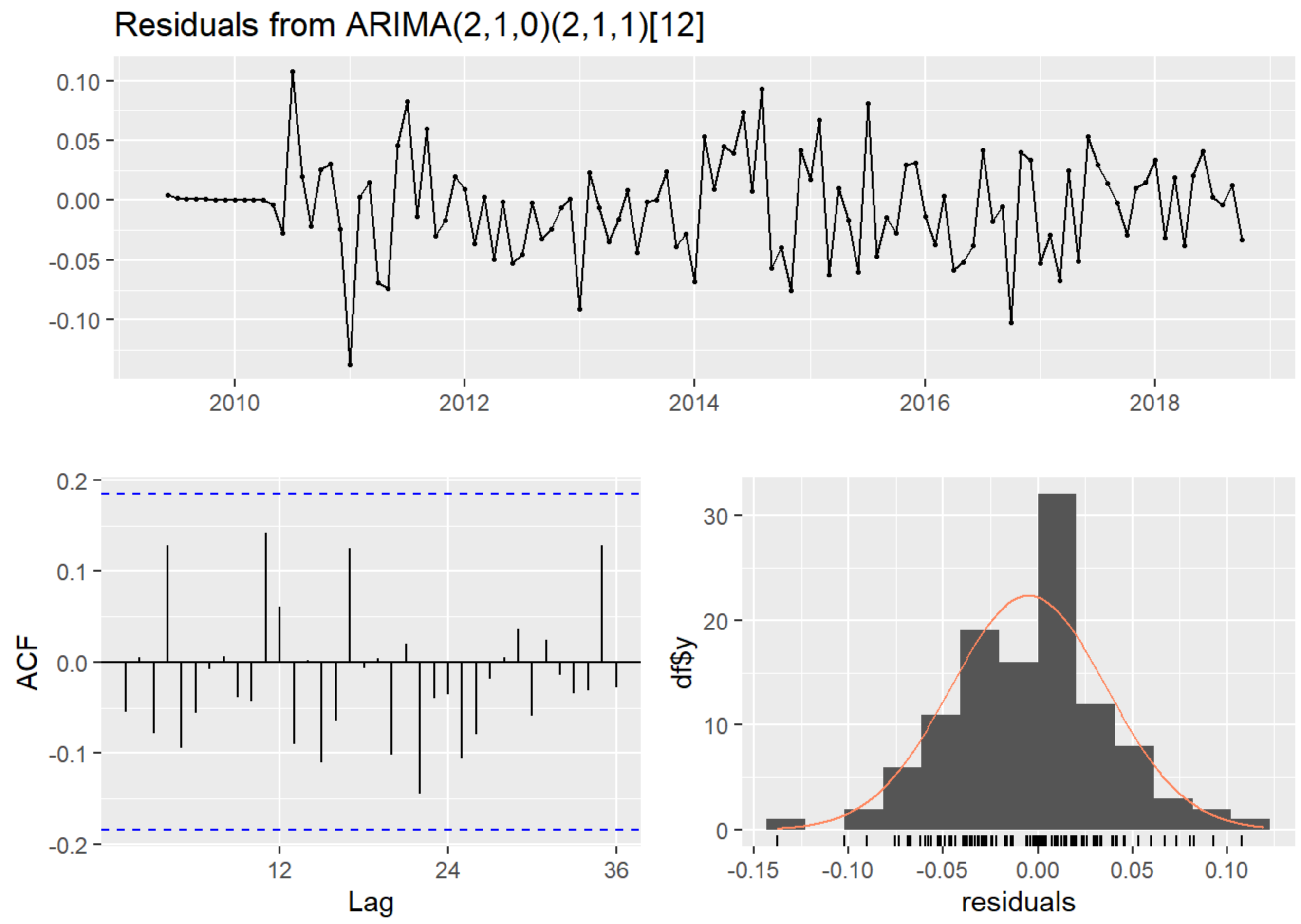
```
arima_model <- auto.arima(jobs_train, seasonal=TRUE, lambda=0)
summary(arima_model) # for Ljung Box Test: small p-value <- reject Ho <- residuals show
autocorrelation
```

```
## Series: jobs_train
## ARIMA(2,1,0)(2,1,1)[12]
## Box Cox transformation: lambda= 0
##
## Coefficients:
##           ar1      ar2     sar1     sar2     sma1
##       -0.7568  -0.5071  -0.1438  -0.2855  -0.6474
## s.e.   0.0889   0.0894   0.1564   0.1372   0.1767
##
## sigma^2 = 0.002044:  log likelihood = 163.43
## AIC=-314.86   AICc=-313.96   BIC=-299.23
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -24.50178 200.9735 151.9025 -0.5841275 3.154182 0.2872686
##                     ACF1
## Training set -0.09766029
```

```
arima_forecast <- forecast(arima_model, h=12)

autoplot(jobs_train) +
  autolayer(arima_forecast, series='Seasonal ARIMA') +
  autolayer(jobs_test, series='Test Data') +
  ggtitle('Forecasting Monthly Job Openings') +
  ylab('Jobs (000s)') +
  guides(colour=guide_legend(title='Forecast Method'))
```



```
checkresiduals(arima_model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,0)(2,1,1)[12]
## Q* = 17.987, df = 18, p-value = 0.4565
##
## Model df: 5.   Total lags used: 23
```

```
# calc forecast RMSE
rmse_arima <- calculate_rmse(jobs_test, arima_forecast$mean)
cat('Test RMSE for ARIMA model: ', rmse_arima)
```

```
## Test RMSE for ARIMA model:  679.0017
```

```
mape_arima <- calculate_mape(jobs_test, arima_forecast$mean)
cat('Test MAPE for Seasonal ARIMA Model: ', mape_arima)
```

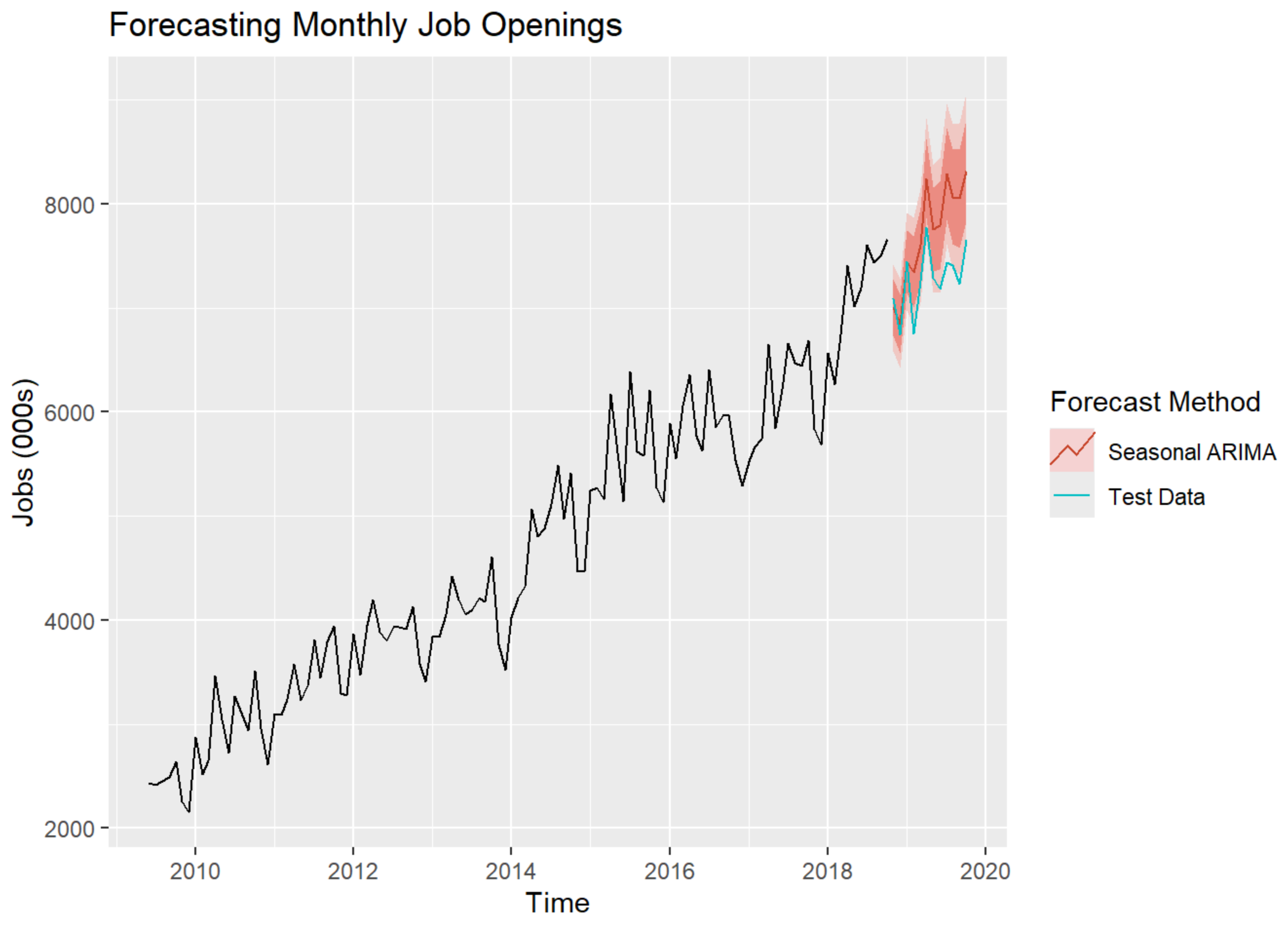
```
## Test MAPE for Seasonal ARIMA Model:  8.30189
```

Comment: The sar1 coefficient is not statistically significant. let's manually use the Arima() function and remove it

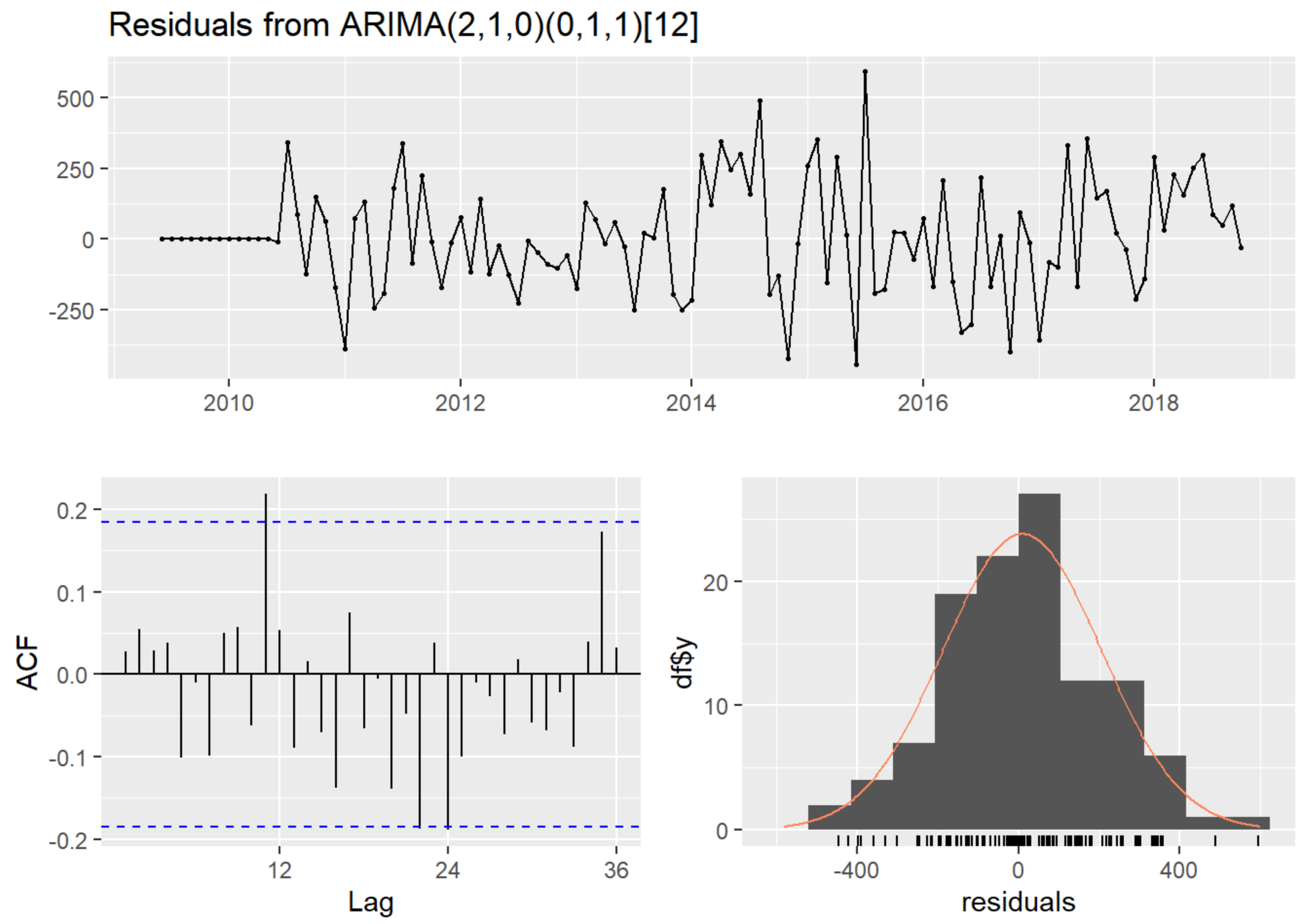
```
arima_model <- Arima(jobs_train, order=c(2,1,0), seasonal=list(order=c(0,1,1), period=1
2))
summary(arima_model) # for Ljung Box Test: small p-value <- reject Ho <- residuals show
autocorrelation
```

```
## Series: jobs_train
## ARIMA(2,1,0)(0,1,1)[12]
##
## Coefficients:
##           ar1      ar2     sma1
##       -0.7712  -0.3767  -0.7599
## s.e.   0.0941   0.0934   0.1133
##
## sigma^2 = 44889:  log likelihood = -681.44
## AIC=1370.89   AICc=1371.31   BIC=1381.31
##
## Training set error measures:
##                    ME    RMSE      MAE         MPE    MAPE      MASE       ACF1
## Training set 8.870617 196.298 149.2179 -0.02648977 3.16061 0.2821916 0.02794118
```

```
arima_forecast <- forecast(arima_model, h=12)

autoplot(jobs_train) +
  autolayer(arima_forecast, series='Seasonal ARIMA') +
  autolayer(jobs_test, series='Test Data') +
  ggtitle('Forecasting Monthly Job Openings') +
  ylab('Jobs (000s)') +
  guides(colour=guide_legend(title='Forecast Method'))
```



```
checkresiduals(arima_model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,0)(0,1,1)[12]
## Q* = 24.671, df = 20, p-value = 0.2143
##
## Model df: 3.   Total lags used: 23
```

```
# calc forecast RMSE
rmse_arima <- calculate_rmse(jobs_test, arima_forecast$mean)
cat('Test RMSE for Manual ARIMA model: ', rmse_arima)
```

```
## Test RMSE for Manual ARIMA model:  546.7742
```

```
mape_arima <- calculate_mape(jobs_test, arima_forecast$mean)
cat('Test MAPE for Seasonal ARIMA Model: ', mape_arima)
```

```
## Test MAPE for Seasonal ARIMA Model:  6.479555
```

Comment: We get marginally better results when we specify the terms of the seasonal ARIMA model (to exclude the seasonal AR components), but both the ETS and seasonal ARIMA models underperform the seasonal naive model. In general, the forecasts look OK, but there’s clearly room for improvement.

Step 5: Build models incorporating external regressors

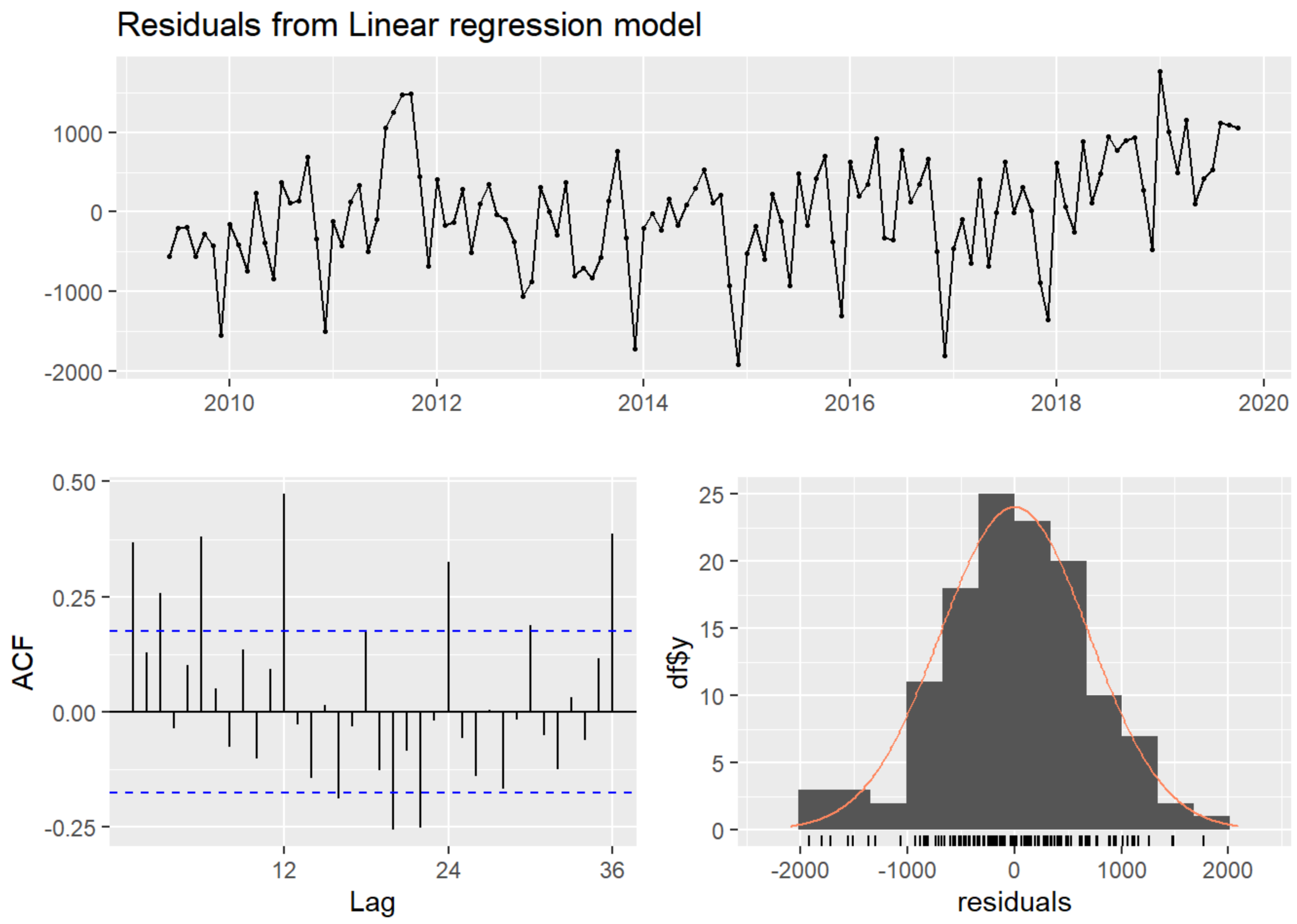
Step 5a: build a regression model with ARIMA errors

Comment: Begin by building a linear model where we predict jobs with consumer sentiment and retail sales. Then check the residuals of that model to make sure they show autocorrelation.

```
# first, let's create linear model where we predict jobs with sentiment and retail sales
linear_model <- tslm(jobs ~ sentiment + retail, data=data)
summary(linear_model)
```

```
##
## Call:
## tslm(formula = jobs ~ sentiment + retail, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1923.88  -421.08     1.34   422.77  1773.97
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.091e+03  5.101e+02 -11.941  < 2e-16 ***
## sentiment    8.508e+01  8.491e+00  10.021  < 2e-16 ***
## retail       1.002e-02  1.972e-03   5.079 1.38e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 703.2 on 122 degrees of freedom
## Multiple R-squared:  0.8008, Adjusted R-squared:  0.7975
## F-statistic: 245.2 on 2 and 122 DF,  p-value: < 2.2e-16
```

```
checkresiduals(linear_model)
```



```
##
##  Breusch-Godfrey test for serial correlation of order up to 24
##
## data:  Residuals from Linear regression model
## LM test = 82.359, df = 24, p-value = 2.55e-08
```

Comment: As shown above, the residuals from our multiple linear regression model are not indicative of white noise. This is what we want. Now we can model the error from the regression model as an ARIMA model, which we do below.

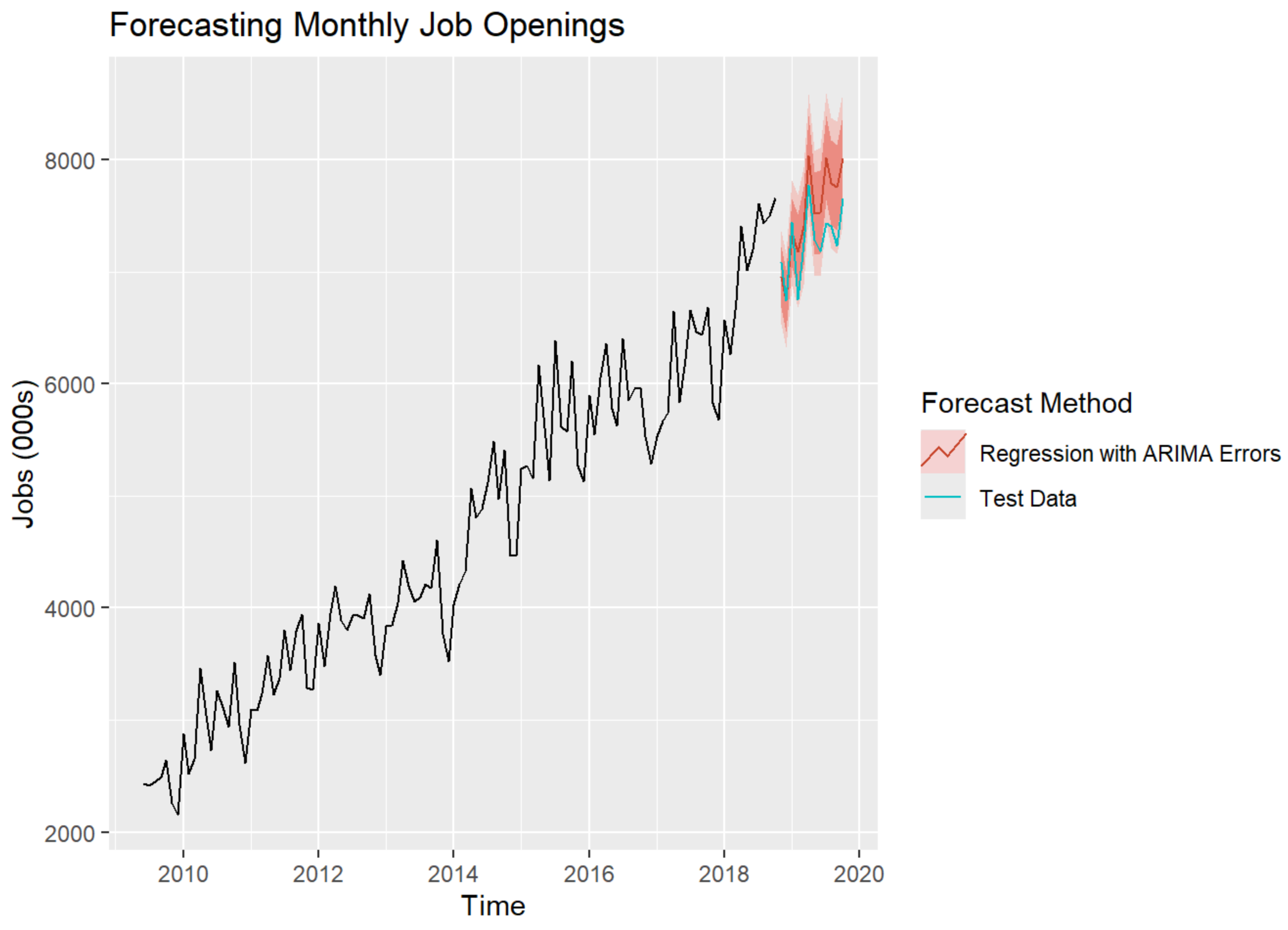
```
reg_arima_model <- auto.arima(jobs_train, xreg=cbind(sentiment_train, retail_train), sea
sonal=TRUE)
summary(reg_arima_model)
```

```
## Series: jobs_train
## Regression with ARIMA(1,0,2)(0,1,1)[12] errors
##
## Coefficients:
##          ar1      ma1     ma2     sma1    drift  sentiment_train  retail_train
##       0.8726  -0.7367  0.3852  -0.7533  42.7080          -1.9345        0.0005
## s.e.  0.0704   0.1090  0.1096   0.1215   4.9992           4.7701        0.0033
##
## sigma^2 = 44883:  log likelihood = -686.09
## AIC=1388.17   AICc=1389.74   BIC=1409.09
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE    MASE       ACF1
## Training set -4.040007 193.2266 147.9321 -0.3336515 3.196505 0.27976 0.04171122
```

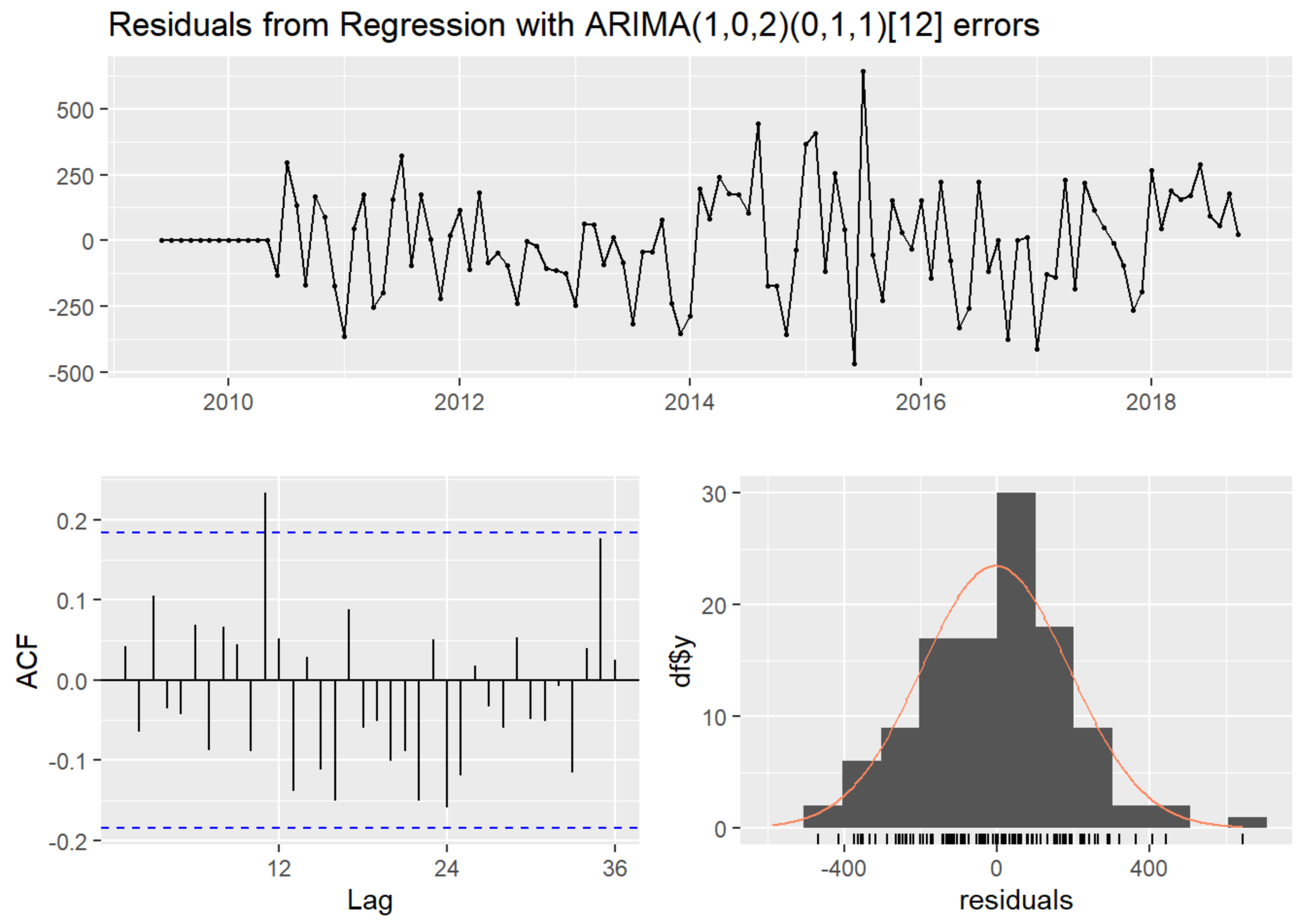
```
reg_arima_forecast <- forecast(reg_arima_model, xreg=cbind(sentiment_test, retail_test),
h=12)
```

```
## Warning in forecast.forecast_ARIMA(reg_arima_model, xreg =
## cbind(sentiment_test, : xreg contains different column names from the xreg used
## in training. Please check that the regressors are in the same order.
```

```
autoplot(jobs_train) +
  autolayer(reg_arima_forecast, series='Regression with ARIMA Errors') +
  autolayer(jobs_test, series='Test Data') +
  ggtitle('Forecasting Monthly Job Openings') +
  ylab('Jobs (000s)') +
  guides(colour=guide_legend(title='Forecast Method'))
```



```
checkresiduals(reg_arima_model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from Regression with ARIMA(1,0,2)(0,1,1)[12] errors
## Q* = 28.182, df = 19, p-value = 0.07999
##
## Model df: 4.   Total lags used: 23
```

```
# calc rmse and mape
rmse_reg_arima <- calculate_rmse(jobs_test, reg_arima_forecast$mean)
cat('RMSE for Regression with ARIMA Errors Model: ', rmse_reg_arima)
```

```
## RMSE for Regression with ARIMA Errors Model:  339.053
```

```
mape_reg_arima <- calculate_mape(jobs_test, reg_arima_forecast$mean)
cat('Test MAPE for Regression w/ ARIMA Errors Model: ', mape_reg_arima)
```

```
## Test MAPE for Regression w/ ARIMA Errors Model:  4.015795
```

Comment: As shown above, the residuals (the ARIMA errors) are not significantly different from white noise. This is a good sign that this is a sensible model. This also represents the best model yet in terms of RMSE and MAPE. However, our model still overshoots its forecast in the later periods. Let's see if we can improve with other modeling approaches.

Step 5b: build a vector autoregressive moving average model (VARMA)

```
print('Recommended lag length:')
```

```
## [1] "Recommended lag length:"
```

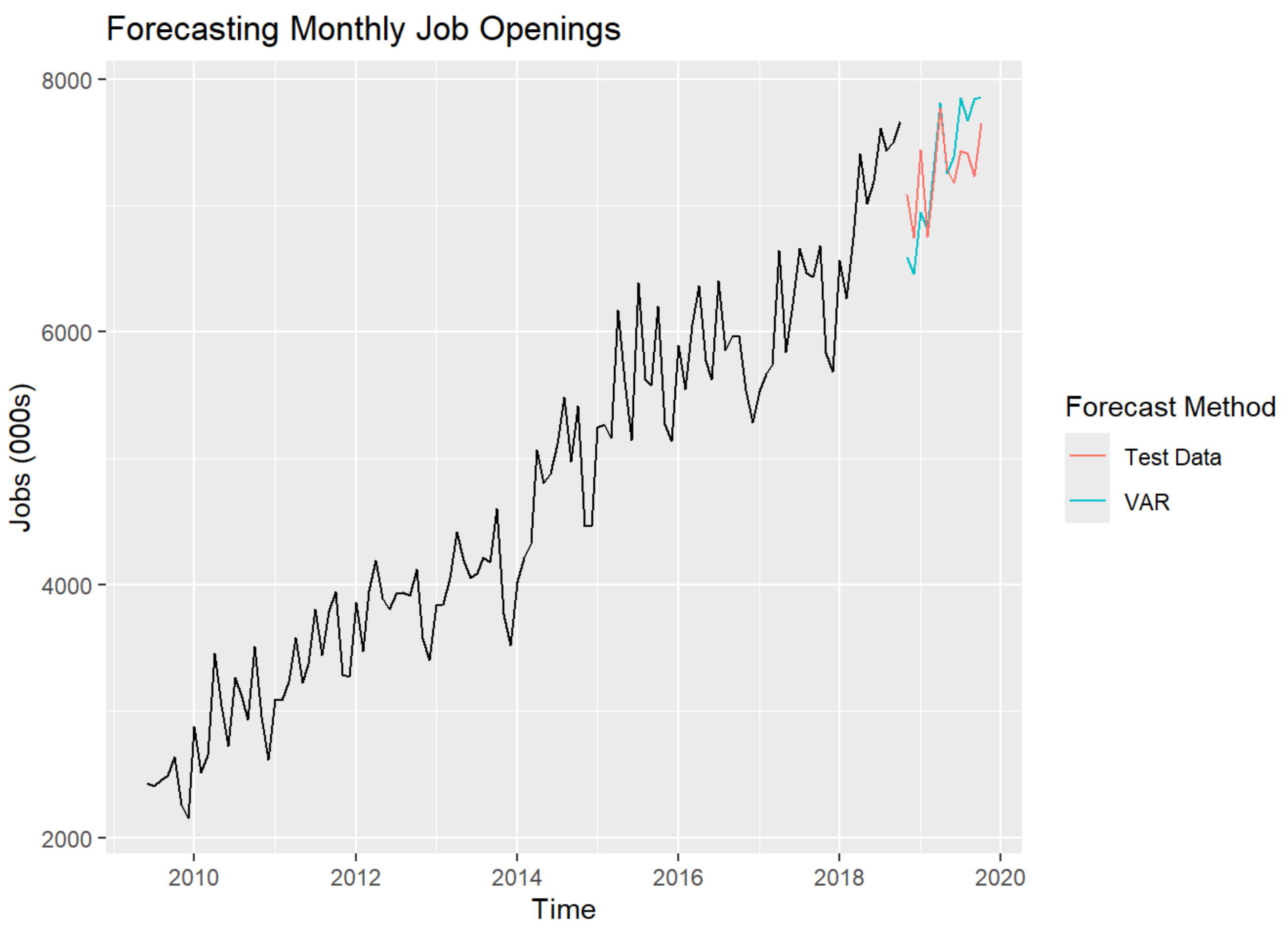
```
VARselect(train_data, lag.max=12, type='both')$selection
```

```
## AIC(n)  HQ(n)  SC(n) FPE(n) 
##     12     12     12     12
```

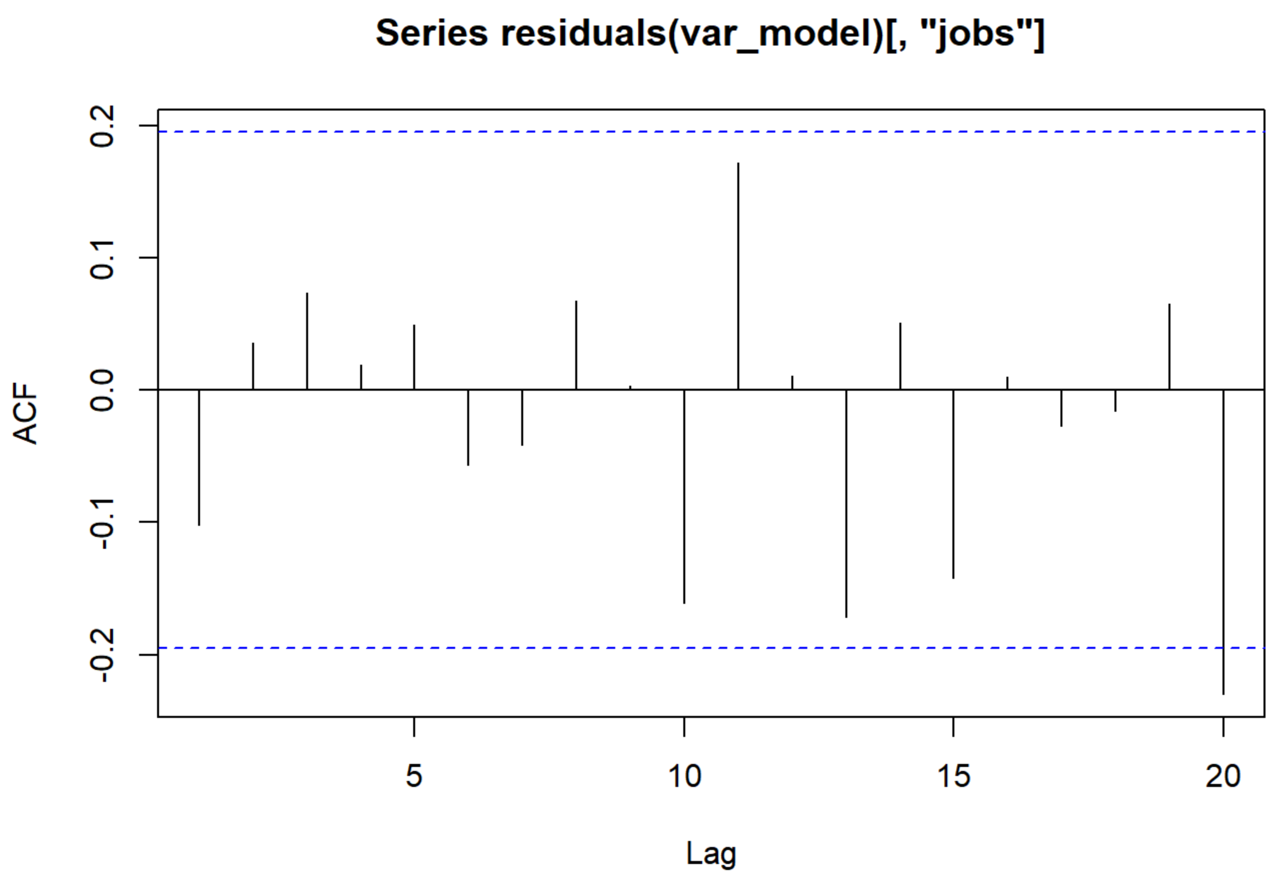
```
var_model <- VAR(train_data, p=12, type='both')
summary(var_model)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: jobs, sentiment, retail
## Deterministic variables: both
## Sample size: 101
## Log Likelihood: -1895.644
## Roots of the characteristic polynomial:
## 1.005 1.005 1.004 1.004 1.001 1.001     1     1 0.9995 0.9995 0.9865 0.9555 0.9555 0.
9475 0.9448 0.9448 0.9299 0.9299 0.903 0.903 0.8961 0.8961 0.8762 0.8762 0.8689 0.8689
0.7864 0.7864 0.775 0.775 0.7418 0.7418 0.7401 0.7401 0.7322 0.7322
## Call:
## VAR(y = train_data, p = 12, type = "both")
##
##
## Estimation results for equation jobs:
## =====================================
## jobs = jobs.l1 + sentiment.l1 + retail.l1 + jobs.l2 + sentiment.l2 + retail.l2 + job
s.l3 + sentiment.l3 + retail.l3 + jobs.l4 + sentiment.l4 + retail.l4 + jobs.l5 + sentime
nt.l5 + retail.l5 + jobs.l6 + sentiment.l6 + retail.l6 + jobs.l7 + sentiment.l7 + retai
l.l7 + jobs.l8 + sentiment.l8 + retail.l8 + jobs.l9 + sentiment.l9 + retail.l9 + jobs.l1
0 + sentiment.l10 + retail.l10 + jobs.l11 + sentiment.l11 + retail.l11 + jobs.l12 + sent
iment.l12 + retail.l12 + const + trend
##
##                 Estimate Std. Error t value Pr(>|t|)
## jobs.l1        2.611e-01  1.218e-01   2.144 0.035932 *
## sentiment.l1   3.429e+00  7.934e+00   0.432 0.667108
## retail.l1     -5.181e-03  2.575e-03  -2.012 0.048476 *
## jobs.l2        4.277e-01  1.239e-01   3.452 0.000999 ***
## sentiment.l2  -7.851e+00  9.898e+00  -0.793 0.430653
## retail.l2     -2.953e-03  2.584e-03  -1.143 0.257386
## jobs.l3        3.952e-01  1.445e-01   2.735 0.008100 **
## sentiment.l3   9.117e+00  1.007e+01   0.906 0.368600
## retail.l3     -9.762e-04  2.559e-03  -0.381 0.704174
## jobs.l4       -5.896e-02  1.478e-01  -0.399 0.691246
## sentiment.l4   2.685e+00  9.977e+00   0.269 0.788691
## retail.l4      3.123e-03  2.550e-03   1.225 0.225258
## jobs.l5       -3.178e-01  1.455e-01  -2.183 0.032736 *
## sentiment.l5  -7.088e+00  9.930e+00  -0.714 0.477971
## retail.l5      1.177e-03  2.616e-03   0.450 0.654248
## jobs.l6        6.708e-02  1.505e-01   0.446 0.657269
## sentiment.l6   5.235e+00  1.011e+01   0.518 0.606399
## retail.l6     -1.146e-03  2.632e-03  -0.436 0.664678
## jobs.l7       -1.645e-01  1.549e-01  -1.062 0.292186
## sentiment.l7  -1.960e+00  9.894e+00  -0.198 0.843604
## retail.l7      2.980e-03  2.247e-03   1.326 0.189618
## jobs.l8       -3.688e-02  1.556e-01  -0.237 0.813379
## sentiment.l8  -3.957e+00  9.760e+00  -0.405 0.686538
## retail.l8      5.397e-03  2.158e-03   2.501 0.014994 *
## jobs.l9        7.606e-02  1.556e-01   0.489 0.626576
## sentiment.l9   1.319e+00  9.363e+00   0.141 0.888391
## retail.l9      4.047e-03  2.183e-03   1.854 0.068479 .
## jobs.l10      -8.402e-02  1.476e-01  -0.569 0.571157    
## sentiment.l10 -5.353e+00  9.239e+00  -0.579 0.564398    
## retail.l10     5.640e-03  2.315e-03   2.436 0.017705 *  
## jobs.l11       1.346e-01  1.433e-01   0.939 0.351237    
## sentiment.l11  1.172e+01  9.420e+00   1.244 0.218068    
## retail.l11    -1.815e-03  2.037e-03  -0.891 0.376290    
## jobs.l12      -2.628e-02  1.456e-01  -0.180 0.857360    
## sentiment.l12  5.482e+00  7.875e+00   0.696 0.488906    
## retail.l12    -8.118e-03  1.986e-03  -4.087 0.000126 ***
## const         -6.476e+02  2.783e+03  -0.233 0.816735    
## trend          6.873e+00  1.383e+01   0.497 0.620999    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## 
## Residual standard error: 218.6 on 63 degrees of freedom
## Multiple R-Squared: 0.9823,  Adjusted R-squared: 0.9719 
## F-statistic: 94.65 on 37 and 63 DF,  p-value: < 2.2e-16 
## 
## 
## Estimation results for equation sentiment: 
## ========================================== 
## sentiment = jobs.l1 + sentiment.l1 + retail.l1 + jobs.l2 + sentiment.l2 + retail.l2 + 
jobs.l3 + sentiment.l3 + retail.l3 + jobs.l4 + sentiment.l4 + retail.l4 + jobs.l5 + sent
iment.l5 + retail.l5 + jobs.l6 + sentiment.l6 + retail.l6 + jobs.l7 + sentiment.l7 + ret
ail.l7 + jobs.l8 + sentiment.l8 + retail.l8 + jobs.l9 + sentiment.l9 + retail.l9 + jobs.
l10 + sentiment.l10 + retail.l10 + jobs.l11 + sentiment.l11 + retail.l11 + jobs.l12 + se
ntiment.l12 + retail.l12 + const + trend 
## 
##                 Estimate Std. Error t value Pr(>|t|)    
## jobs.l1       -1.882e-03  1.898e-03  -0.992   0.3251    
## sentiment.l1   7.203e-01  1.236e-01   5.828 2.08e-07 ***
## retail.l1     -5.129e-05  4.011e-05  -1.279   0.2056    
## jobs.l2        1.055e-03  1.930e-03   0.546   0.5867    
## sentiment.l2  -1.672e-01  1.542e-01  -1.084   0.2824    
## retail.l2     -6.523e-05  4.025e-05  -1.621   0.1101    
## jobs.l3        1.862e-04  2.251e-03   0.083   0.9343    
## sentiment.l3   1.647e-03  1.568e-01   0.011   0.9917    
## retail.l3     -8.011e-05  3.987e-05  -2.009   0.0488 *  
## jobs.l4        1.550e-03  2.302e-03   0.673   0.5032    
## sentiment.l4   5.768e-02  1.554e-01   0.371   0.7118    
## retail.l4     -5.854e-05  3.973e-05  -1.473   0.1456    
## jobs.l5        2.764e-04  2.267e-03   0.122   0.9033    
## sentiment.l5  -4.167e-02  1.547e-01  -0.269   0.7885    
## retail.l5     -1.141e-05  4.075e-05  -0.280   0.7803    
## jobs.l6        3.149e-03  2.344e-03   1.343   0.1840    
## sentiment.l6   1.572e-02  1.575e-01   0.100   0.9208    
## retail.l6      1.432e-05  4.100e-05   0.349   0.7281    
## jobs.l7       -1.209e-03  2.412e-03  -0.501   0.6181    
## sentiment.l7  -6.201e-02  1.541e-01  -0.402   0.6888    
## retail.l7      2.202e-05  3.501e-05   0.629   0.5316    
## jobs.l8        1.598e-03  2.423e-03   0.659   0.5120    
## sentiment.l8  -7.418e-03  1.520e-01  -0.049   0.9612
## retail.l8      4.587e-05  3.362e-05   1.364   0.1773
## jobs.l9       -2.366e-03  2.423e-03  -0.976   0.3326
## sentiment.l9  -8.735e-04  1.459e-01  -0.006   0.9952
## retail.l9      4.509e-05  3.401e-05   1.326   0.1897
## jobs.l10       1.319e-03  2.299e-03   0.574   0.5681
## sentiment.l10 -1.697e-02  1.439e-01  -0.118   0.9065
## retail.l10     2.756e-05  3.607e-05   0.764   0.4477
## jobs.l11      -3.077e-03  2.232e-03  -1.378   0.1729
## sentiment.l11  4.584e-02  1.467e-01   0.312   0.7558
## retail.l11     8.893e-06  3.173e-05   0.280   0.7802
## jobs.l12      -1.928e-03  2.269e-03  -0.850   0.3987
## sentiment.l12 -1.518e-01  1.227e-01  -1.238   0.2204
## retail.l12    -2.862e-05  3.094e-05  -0.925   0.3585
## const          8.107e+01  4.335e+01   1.870   0.0661 .
## trend          4.102e-01  2.155e-01   1.904   0.0615 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 3.405 on 63 degrees of freedom
## Multiple R-Squared: 0.9418,  Adjusted R-squared: 0.9076
## F-statistic: 27.55 on 37 and 63 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation retail:
## =======================================
## retail = jobs.l1 + sentiment.l1 + retail.l1 + jobs.l2 + sentiment.l2 + retail.l2 + jo
bs.l3 + sentiment.l3 + retail.l3 + jobs.l4 + sentiment.l4 + retail.l4 + jobs.l5 + sentim
ent.l5 + retail.l5 + jobs.l6 + sentiment.l6 + retail.l6 + jobs.l7 + sentiment.l7 + retai
l.l7 + jobs.l8 + sentiment.l8 + retail.l8 + jobs.l9 + sentiment.l9 + retail.l9 + jobs.l1
0 + sentiment.l10 + retail.l10 + jobs.l11 + sentiment.l11 + retail.l11 + jobs.l12 + sent
iment.l12 + retail.l12 + const + trend
##
##                 Estimate Std. Error t value Pr(>|t|)
## jobs.l1        3.880e+00  3.050e+00   1.272 0.207997
## sentiment.l1  -2.595e+01  1.986e+02  -0.131 0.896463
## retail.l1     -3.326e-02  6.446e-02  -0.516 0.607748
## jobs.l2        1.150e+01  3.102e+00   3.707 0.000445 ***
## sentiment.l2  -9.691e+01  2.478e+02  -0.391 0.697100
## retail.l2      6.070e-02  6.469e-02   0.938 0.351645
## jobs.l3       -4.076e+00  3.619e+00  -1.126 0.264292
## sentiment.l3   3.774e+01  2.521e+02   0.150 0.881473
## retail.l3     -3.354e-02  6.408e-02  -0.523 0.602493
## jobs.l4       -3.631e+00  3.700e+00  -0.981 0.330163
## sentiment.l4  -1.355e+01  2.498e+02  -0.054 0.956906
## retail.l4      8.520e-02  6.386e-02   1.334 0.186946
## jobs.l5        5.338e-01  3.644e+00   0.146 0.883998
## sentiment.l5  -4.310e+02  2.486e+02  -1.734 0.087894 .
## retail.l5      1.923e-01  6.549e-02   2.937 0.004629 **
## jobs.l6       -5.624e+00  3.767e+00  -1.493 0.140477
## sentiment.l6  -1.320e+02  2.531e+02  -0.522 0.603807
## retail.l6     -3.912e-02  6.589e-02  -0.594 0.554845    
## jobs.l7       -3.368e+00  3.877e+00  -0.869 0.388366    
## sentiment.l7  -2.462e+02  2.477e+02  -0.994 0.324138    
## retail.l7      5.867e-02  5.626e-02   1.043 0.301047    
## jobs.l8       -1.276e-01  3.895e+00  -0.033 0.973968    
## sentiment.l8   2.575e+02  2.444e+02   1.054 0.295960    
## retail.l8     -4.659e-04  5.403e-02  -0.009 0.993148    
## jobs.l9       -4.518e+00  3.895e+00  -1.160 0.250469    
## sentiment.l9  -1.492e+02  2.344e+02  -0.636 0.526819    
## retail.l9     -2.500e-01  5.467e-02  -4.574  2.3e-05 ***
## jobs.l10      -2.480e+00  3.695e+00  -0.671 0.504515    
## sentiment.l10  4.798e+02  2.313e+02   2.074 0.042150 *  
## retail.l10    -1.933e-01  5.797e-02  -3.335 0.001435 ** 
## jobs.l11       8.176e+00  3.588e+00   2.279 0.026087 *  
## sentiment.l11 -5.291e+02  2.358e+02  -2.243 0.028394 *  
## retail.l11    -1.198e-01  5.100e-02  -2.349 0.021999 *  
## jobs.l12      -2.016e+00  3.647e+00  -0.553 0.582317    
## sentiment.l12  1.074e+02  1.972e+02   0.544 0.588066    
## retail.l12     7.490e-01  4.973e-02  15.059  < 2e-16 ***
## const          2.194e+05  6.968e+04   3.149 0.002506 ** 
## trend          8.820e+02  3.463e+02   2.547 0.013334 *  
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## 
## Residual standard error: 5473 on 63 degrees of freedom
## Multiple R-Squared: 0.9891,  Adjusted R-squared: 0.9827 
## F-statistic: 154.2 on 37 and 63 DF,  p-value: < 2.2e-16
## 
## 
## 
## Covariance matrix of residuals:
##                jobs sentiment     retail
## jobs       47778.916    -2.178   -14438.2
## sentiment     -2.178    11.595     -195.2
## retail    -14438.189  -195.193 29953258.2
## 
## Correlation matrix of residuals:
##               jobs sentiment   retail
## jobs       1.000000 -0.002927 -0.01207
## sentiment -0.002927  1.000000 -0.01047
## retail    -0.012069 -0.010474  1.00000
```

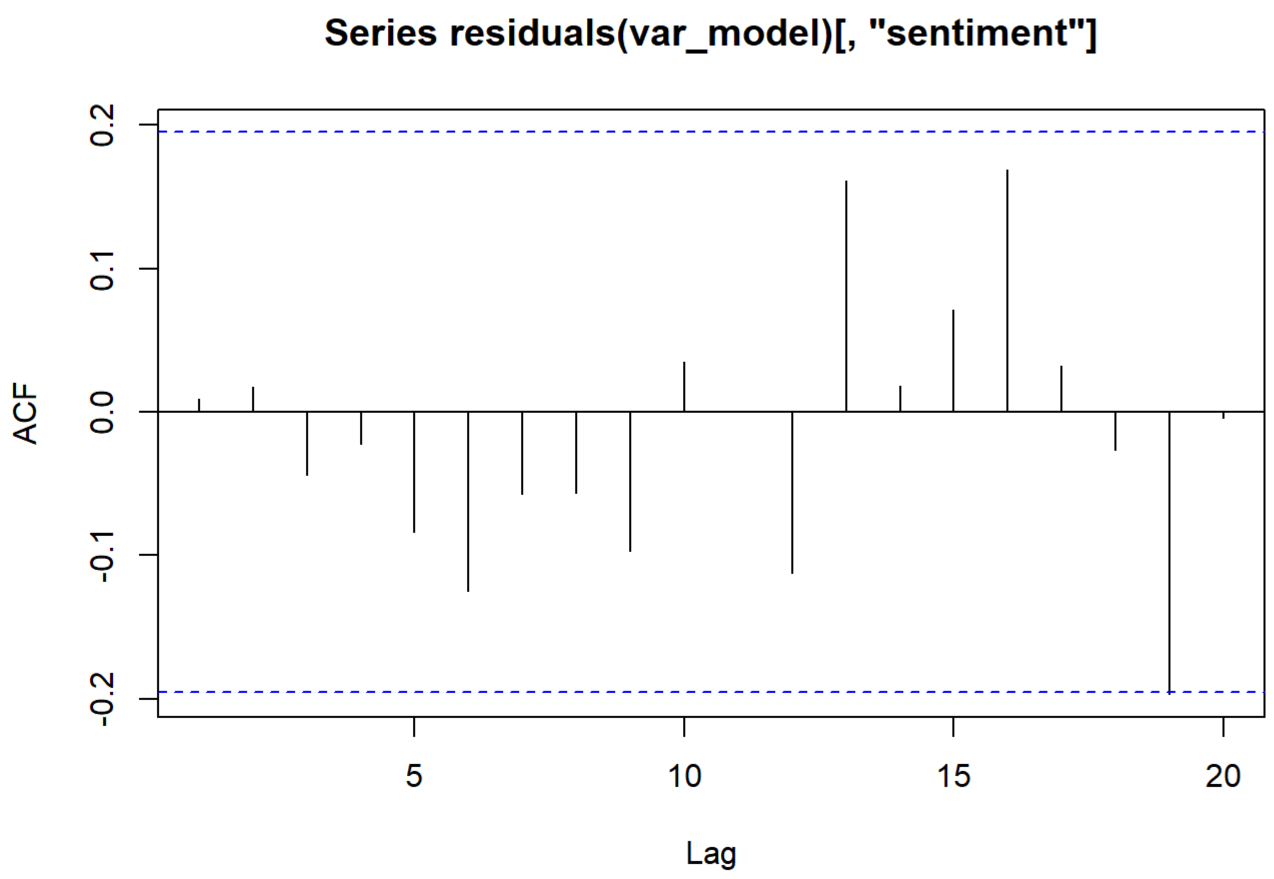
```
var_forecast <- predict(var_model, n.ahead=12)
var_forecast_jobs <- ts(var_forecast$fcst$jobs[,'fcst'], start=test_start, frequency=12)

autoplot(jobs_train) +
  autolayer(var_forecast_jobs, series='VAR') +
  autolayer(jobs_test, series='Test Data') +
  ggtitle('Forecasting Monthly Job Openings') +
  ylab('Jobs (000s)') +
  guides(colour=guide_legend(title='Forecast Method'))
```



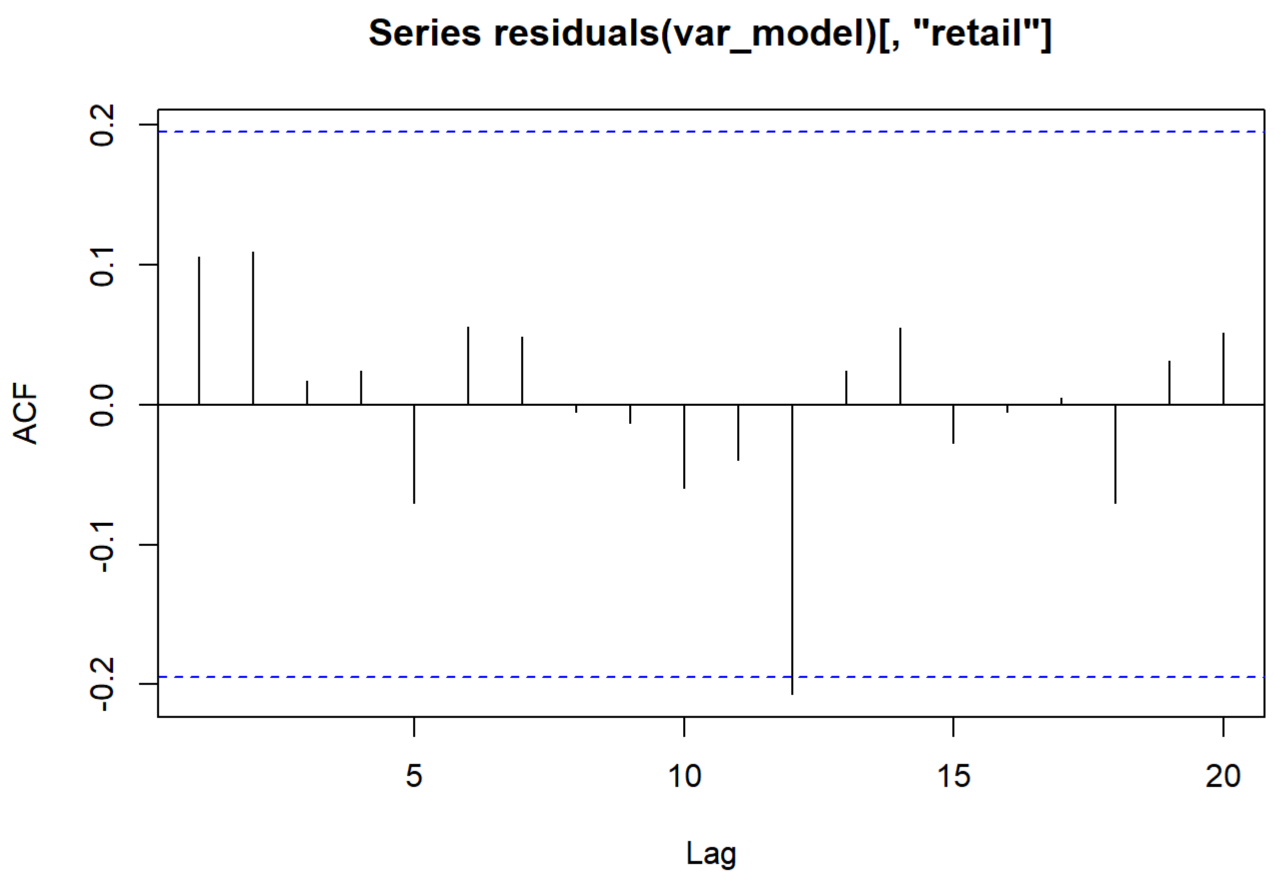
```
acf(residuals(var_model)[,'jobs']) # residuals for jobs
```



```
acf(residuals(var_model)[,'sentiment']) # residuals for sentiment
```



```
acf(residuals(var_model)[,'retail']) # residuals for retail
```



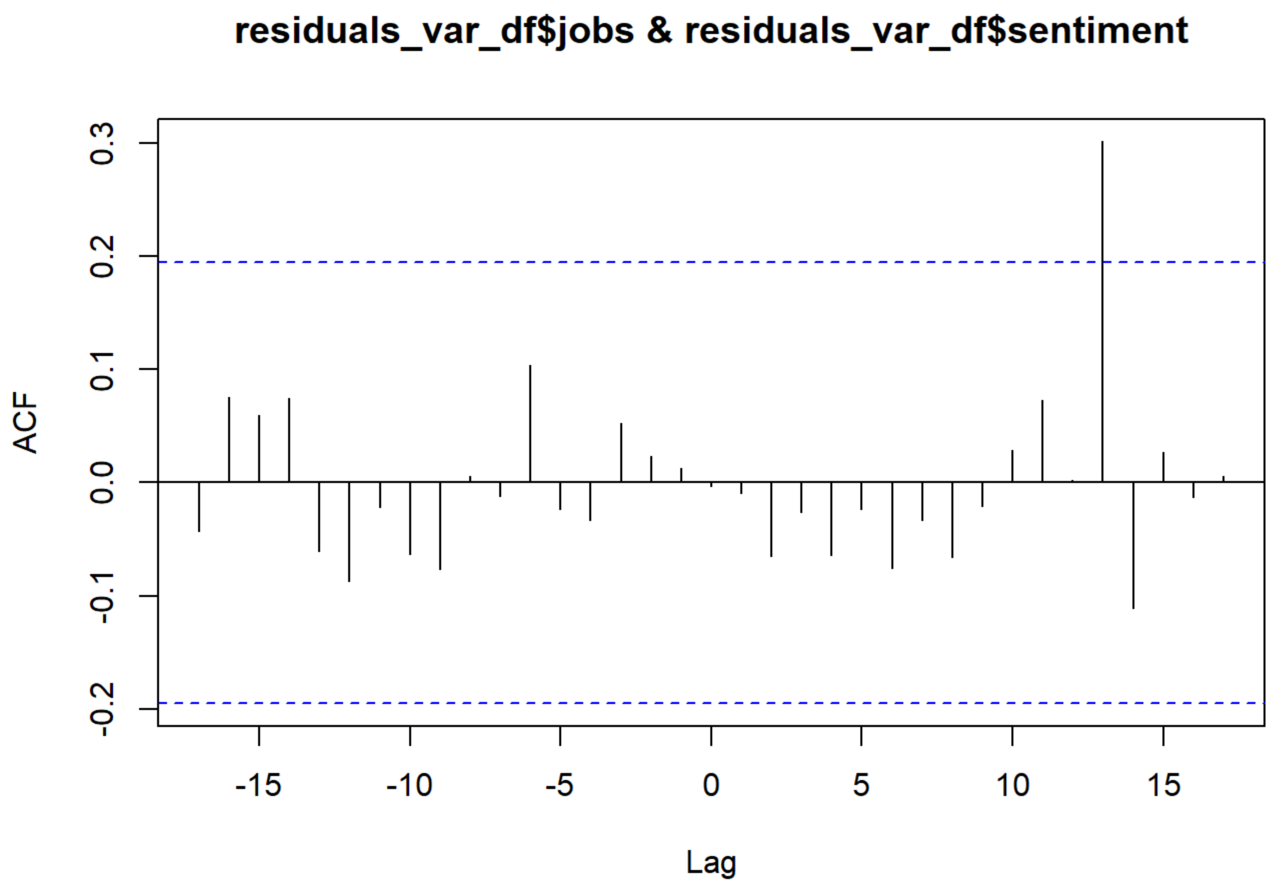
```
# calc rmse and mape
rmse_var <- calculate_rmse(jobs_test, var_forecast_jobs)
cat('RMSE for VAR1 Model: ', rmse_var)
```

```
## RMSE for VAR1 Model:  327.0475
```

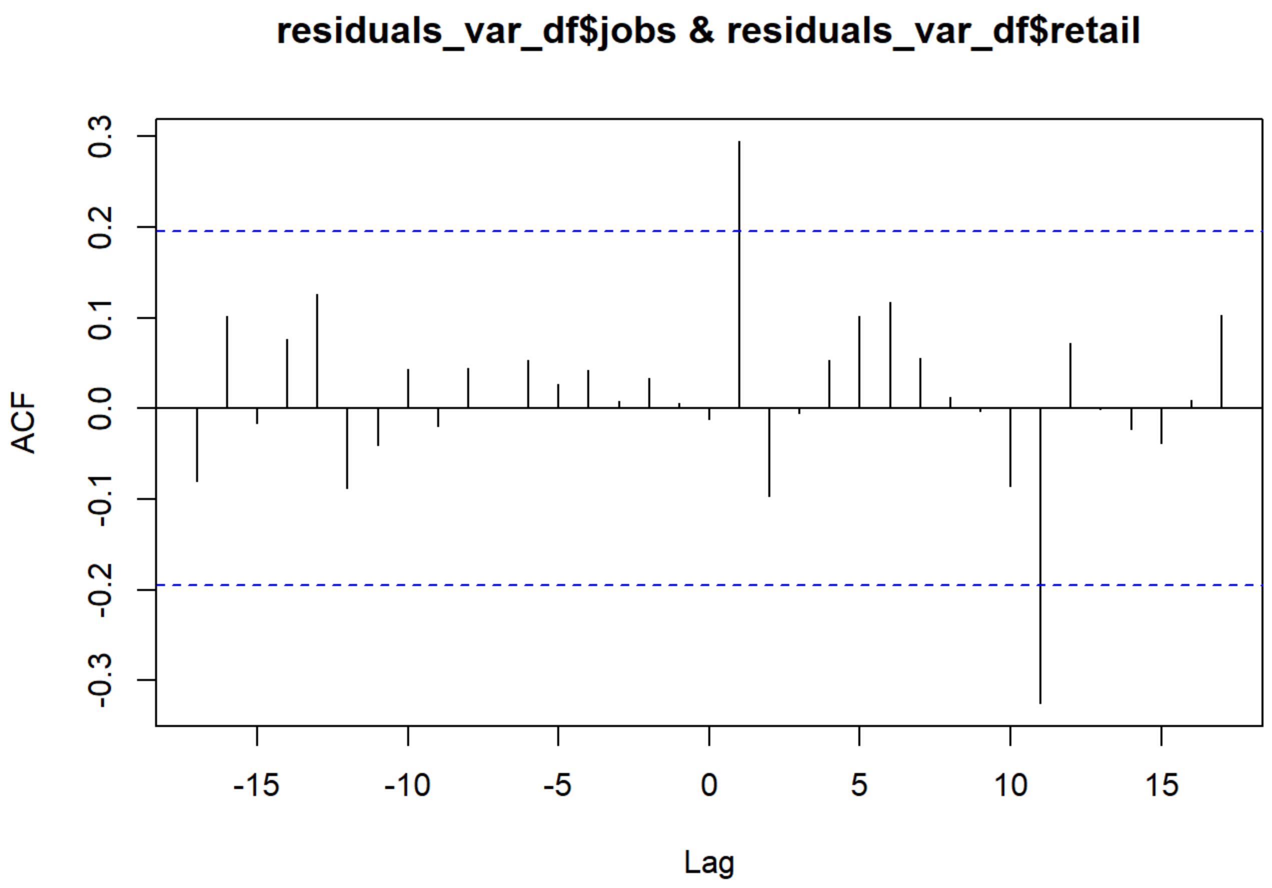
```
mape_var <- calculate_mape(jobs_test, var_forecast_jobs)
cat('Test MAPE for VAR Model: ', mape_var)
```

```
## Test MAPE for VAR Model:  3.658579
```

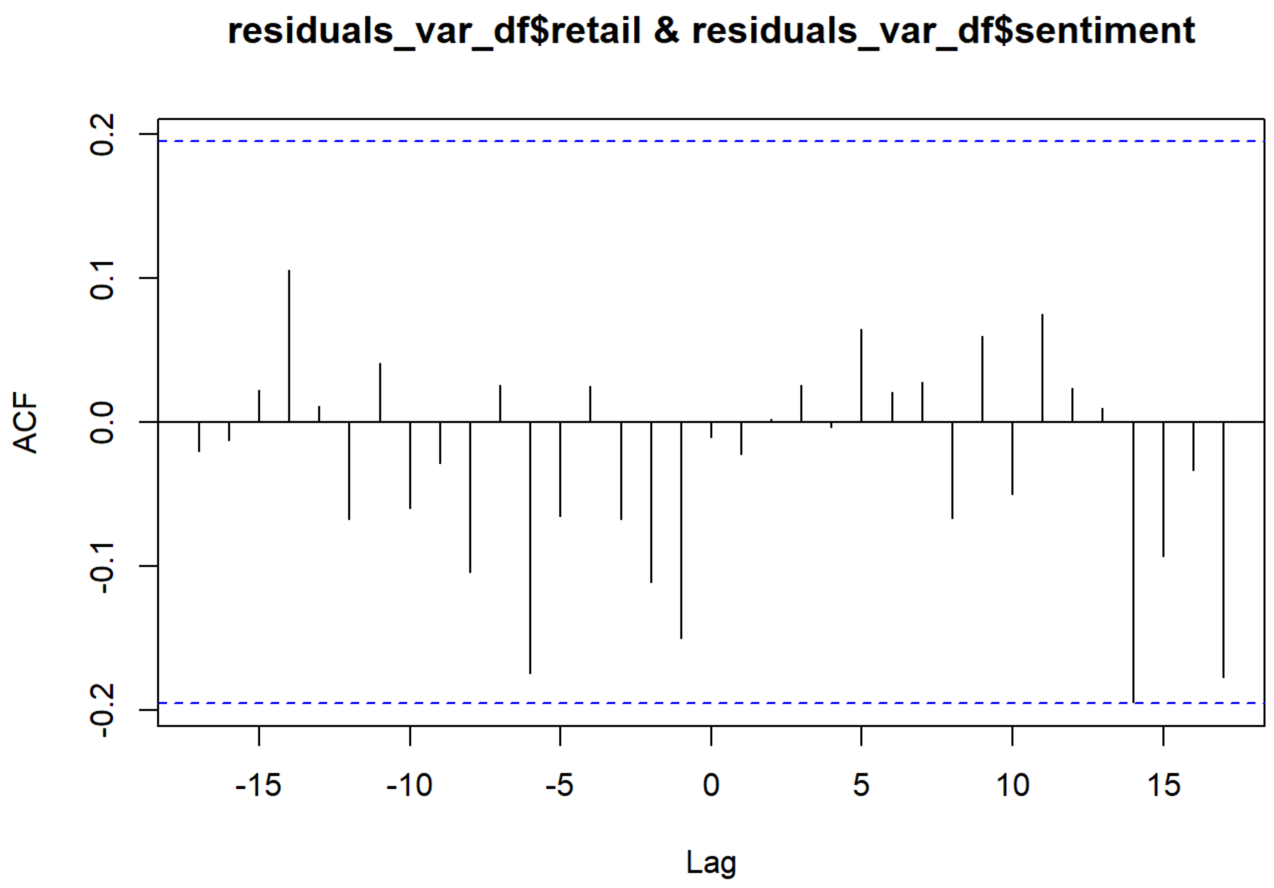
```
# check residuals
residuals_var <- residuals(var_model)
residuals_var_df <- as.data.frame(residuals_var)
ccf(residuals_var_df$jobs, residuals_var_df$sentiment)
```



```
ccf(residuals_var_df$jobs, residuals_var_df$retail)
```



```
ccf(residuals_var_df$retail, residuals_var_df$sentiment)
```



Comment: As shown above, we determined an optimal lag length of p=12 for our VAR() model. While this improved the model's ability to capture the gyrating nature of the time series and improved RMSE and MAPE, it results in an overly complex model, which can lead to overfitting.

Step 5c: fit a neural network model

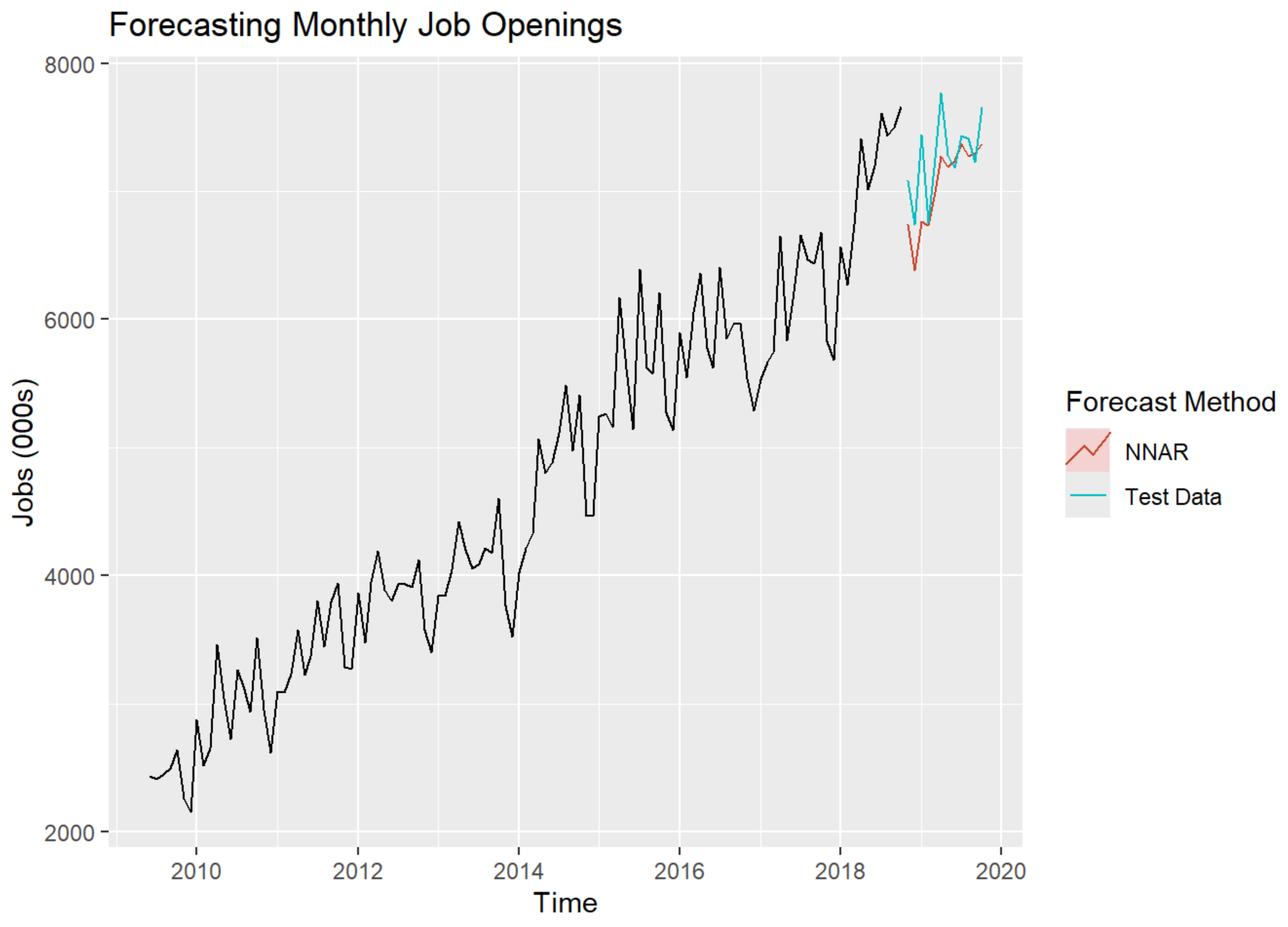
```
nnar_model <- nnetar(jobs_train, xreg=cbind(sentiment_train, retail_train), size=2, deca
y=0.1, P=1)
summary(nnar_model)
```

```
##           Length Class        Mode
## x         113    ts           numeric
## m           1    -none-       numeric
## p           1    -none-       numeric
## P           1    -none-       numeric
## scalex      2    -none-       list
## scalexreg   2    -none-       list
## size        1    -none-       numeric
## xreg      226    mts          numeric
## subset    113    -none-       numeric
## model      20    nnetarmodels list
## nnetargs    1    -none-       list
## fitted    113    ts           numeric
## residuals 113    ts           numeric
## lags        2    -none-       numeric
## series      1    -none-       character
## method      1    -none-       character
## call        6    -none-       call
```

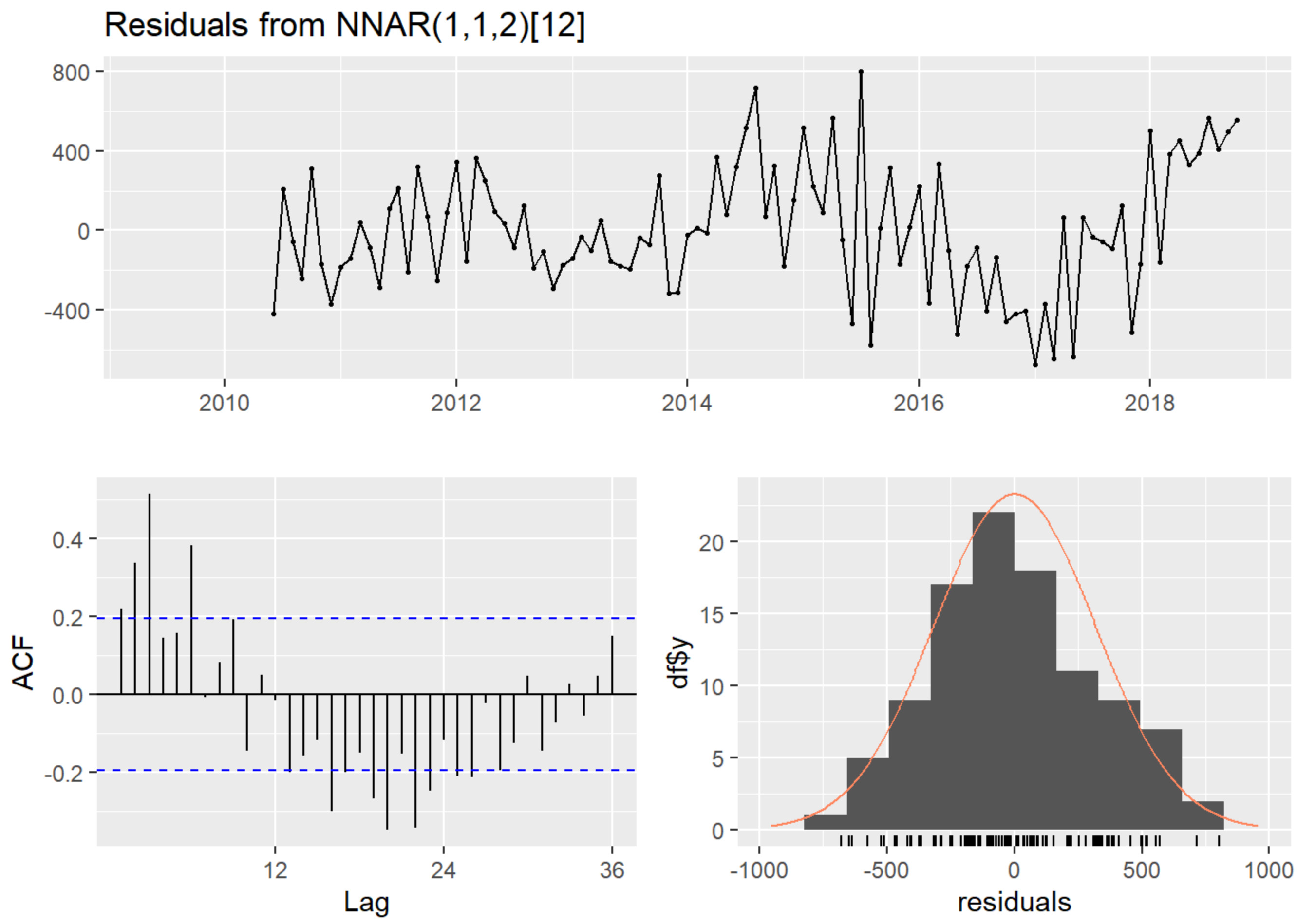
```
nnar_forecast <- forecast(nnar_model, xreg=cbind(sentiment_test, retail_test), h=12)
```

```
## Warning in forecast.nnetar(nnar_model, xreg = cbind(sentiment_test,
## retail_test), : xreg contains different column names from the xreg used in
## training. Please check that the regressors are in the same order.
```

```
autoplot(jobs_train) +
  autolayer(nnar_forecast, series='NNAR') +
  autolayer(jobs_test, series='Test Data') +
  ggtitle('Forecasting Monthly Job Openings') +
  ylab('Jobs (000s)') +
  guides(colour=guide_legend(title='Forecast Method'))
```



```
checkresiduals(nnar_model)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from NNAR(1,1,2)[12]
## Q* = 152.67, df = 23, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 23
```

```
# calc rmse and mape
rmse_nnar <- calculate_rmse(jobs_test, nnar_forecast$mean)
cat('RMSE for NNAR Model: ', rmse_nnar)
```

```
## RMSE for NNAR Model:  308.2435
```

```
mape_nnar <- calculate_mape(jobs_test, nnar_forecast$mean)
cat('Test MAPE for NNAR Model: ', mape_nnar)
```

```
## Test MAPE for NNAR Model:  3.244793
```

Comment: On one hand, the neural network model generated a solid over all forecast, with the best RMSE and MAPE of all our models. However, the model's residuals show signs of autocorrelation.

```
results_df <- data.frame(
  Model = c('Seasonal Naive', 'ETS', 'Seasonal ARIMA', 'Regression w/ ARIMA Errors', 'VA
R', 'NNAR'),
  RMSE = round(c(rmse_snaive, rmse_ets, rmse_arima, rmse_reg_arima, rmse_var, rmse_nna
r),0),
  MAPE = round(c(mape_snaive, mape_ets, mape_arima, mape_reg_arima, mape_var, mape_nna
r),2)
)


results_df[order(results_df$RMSE),]
```

```
##                        Model RMSE MAPE
## 6                       NNAR  308 3.24
## 5                        VAR  327 3.66
## 4 Regression w/ ARIMA Errors  339 4.02
## 2                        ETS  476 5.93
## 3             Seasonal ARIMA  547 6.48
## 1             Seasonal Naive  597 6.22
```