Problem 3

HW3

Chandrakanth, Nisanth (palax002 and dhera003)

February 22, 2017

```
suppressPackageStartupMessages({
  library(readr)
  library(lubridate)
  library(TSA)
  library(ggplot2)
  library(dplyr)
  library(forecast)
  library(stats)
})
```

General Requirements

- Please do not change the path in `read_csv()`, your solutions will be automatically run by the bot and the bot will not have access to the folders that you have.
- Please review the resulting PDF and make sure that all code fits into the page. If you have lines of code that run outside of the page limits we will deduct points for incorrect formatting as it makes it unnecessarily hard to grade.
- Please avoid using esoteric R packages. We have already discovered some that generate arima models incorrectly. Stick to tried and true packages: base R, `forecast`, `TSA`, `zoo`, `xts`.

Forecasting

Please consider the data from file `vehicles_train.csv`. This is a real time series dataset that describes:

- the number of vehicles that travelled on a particular very popular un-named bridge over several years.

Please import it as follows using `read_csv` function from `readr` package

```
vehicles_train <- read_csv("vehicles_train.csv") # Please do not change this line
```

Your company is bidding on an electronic billboard advertising space over that bridge and your boss is asking you to issue a forecast on the number of vehicles that will travel over that bridge.

Depending on your forecasts your company will decide whether to accept the deal, revise it (say, offer to advertise only during particular time periods) or skip it altogether, so your job is to produce daily forecasts for the next 2 months as well as the 95% confidence intervals around these forecasts.

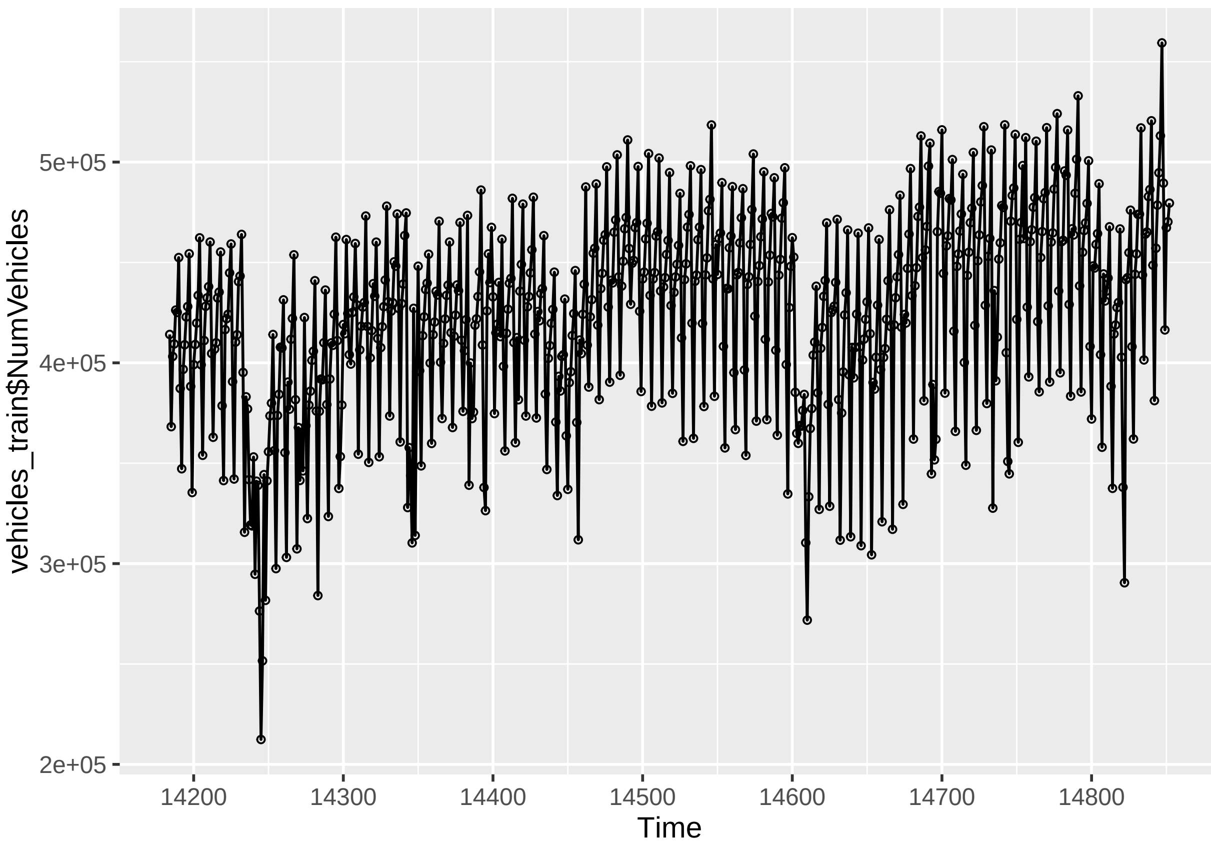
Important:

- As you may have noticed the Day column is imported as text and the system does not recognize it as a date. Also, the time series is not imported as a `ts` type (which is preferred by many functions that we studied).

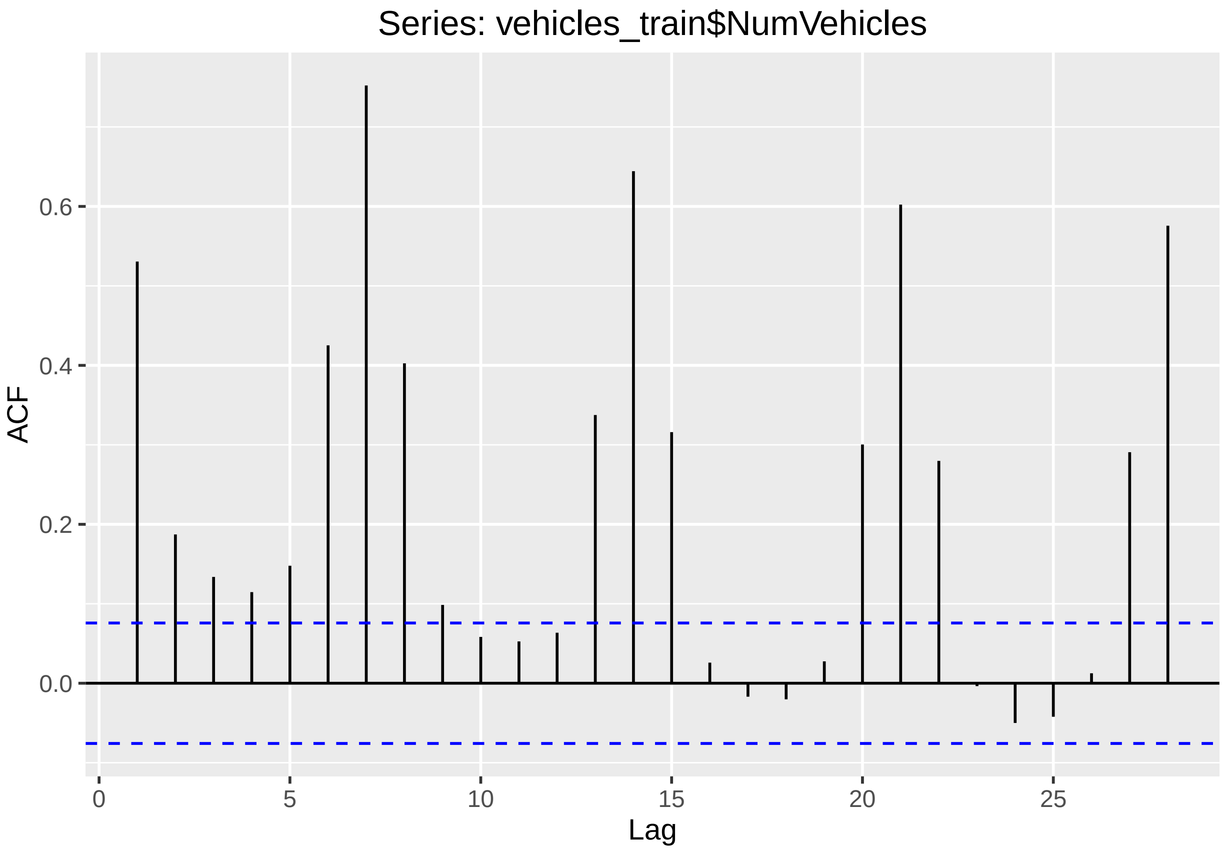
```
head(vehicles_train)
```

```
## # A tibble: 6 x 2
##         Day NumVehicles
##       <chr>       <int>
## 1 01-Nov-08      414144
## 2 02-Nov-08      368204
## 3 03-Nov-08      403180
## 4 04-Nov-08      409408
## 5 05-Nov-08      426276
## 6 06-Nov-08      425136
```

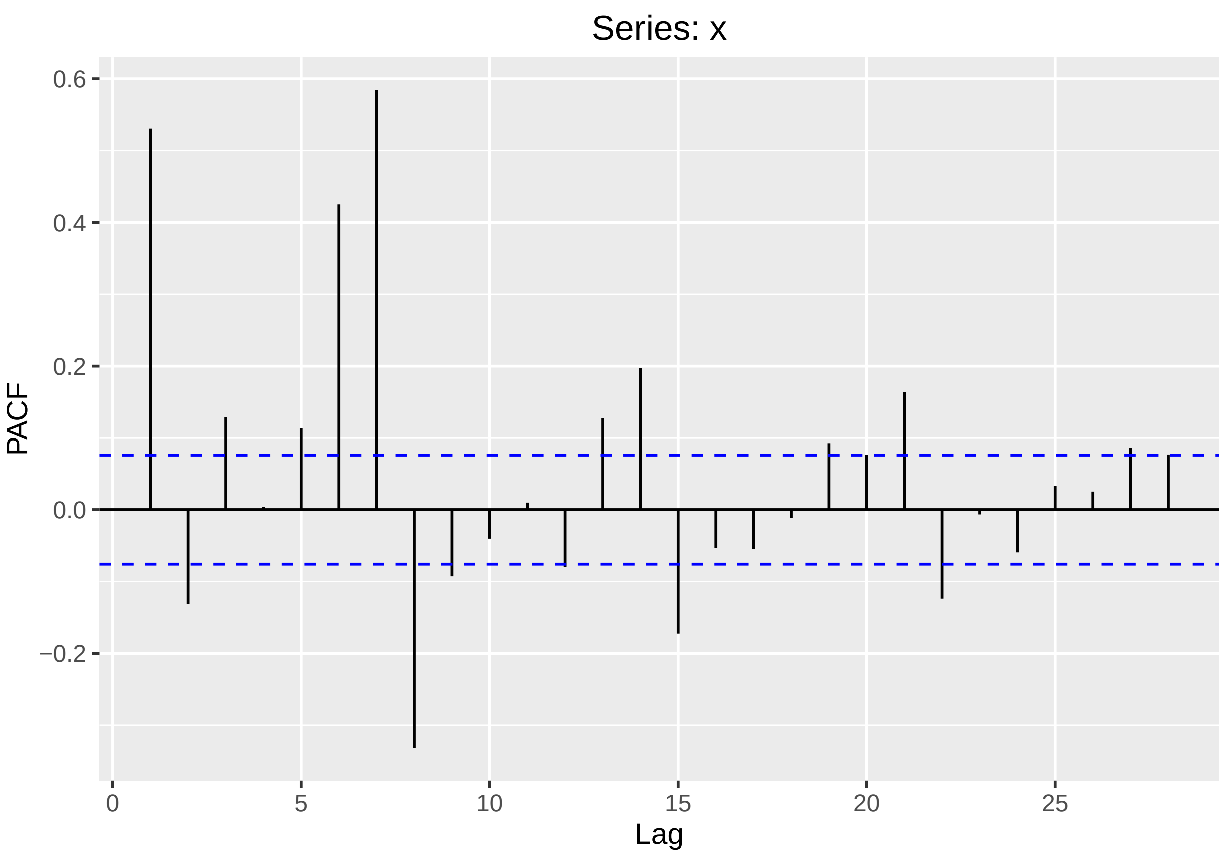
```
vehicles_train$Day<-dmy(vehicles_train$Day)
vehicles_train$NumVehicles<-ts(vehicles_train$NumVehicles,start = vehicles_train$Day[1],
                               frequency = 1)

autoplot(vehicles_train$NumVehicles)+geom_point(shape=1, size=1)
```




```
ggAcf(vehicles_train$NumVehicles)
```




```
ggPacf(vehicles_train$NumVehicles)
```




```
adf.test(vehicles_train$NumVehicles,alternative = "stationary")
```

```
## Warning in adf.test(vehicles_train$NumVehicles, alternative =
## "stationary"): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  vehicles_train$NumVehicles
## Dickey-Fuller = -4.508, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

```
auto.arima(vehicles_train$NumVehicles, approximation = FALSE,stationary = FALSE, seasonal = TRUE)
```

```
## Series: vehicles_train$NumVehicles
## ARIMA(2,1,2)
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.0226  -0.6238  -1.5343  0.7032
## s.e.  0.0424   0.0368   0.0389  0.0386
##
## sigma^2 estimated as 1.471e+09:  log likelihood=-7997.25
## AIC=16004.51   AICc=16004.6   BIC=16027.03
```

```
eacf(vehicles_train$NumVehicles)
```

```
## AR/MA
```

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

- These data issues are typical for your future daily life as a Data Scientist:
  - 80% of your effort will normally be spend on slicing and dicing the data in different ways
  - 20% will be actually about running the models on the data that you prepared and issuing some recommendations
  - Here is some discussion about it
- The job of making the data nice and tidy for the use in your model is *completely* on you. As a Data Scientist you are not expected to ask someone else to do the "data work" for you. You are that person.

Hints:

- I recommend using `readr` and `lubridate` packages in order to work with csv files and parse dates
- You may want to convert it into `ts` in order to fully use all the available functions

Question 1

Please forecast the daily number of vehicles that will travel on that bridge for the next two months.

- More specifically, please load the desired test set from here. You will need to fill in that data.frame:

```r
vehicles_test <- read_csv("vehicles_test.csv") # Please do not change this line
```

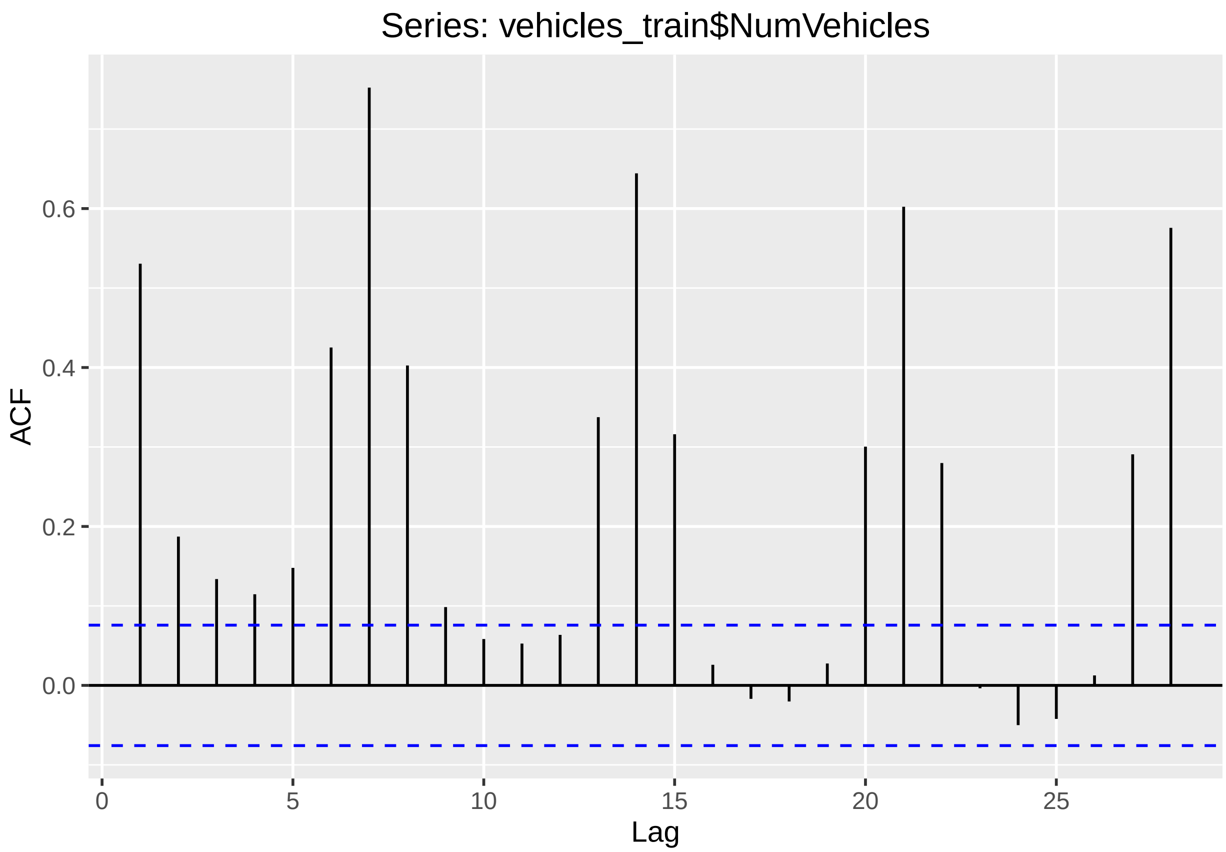
```
## Parsed with column specification:
## cols(
##   Day = col_character(),
##   NumVehicles = col_character(),
##   Low = col_character(),
##   High = col_character()
## )
```

```r
head(vehicles_test)
```

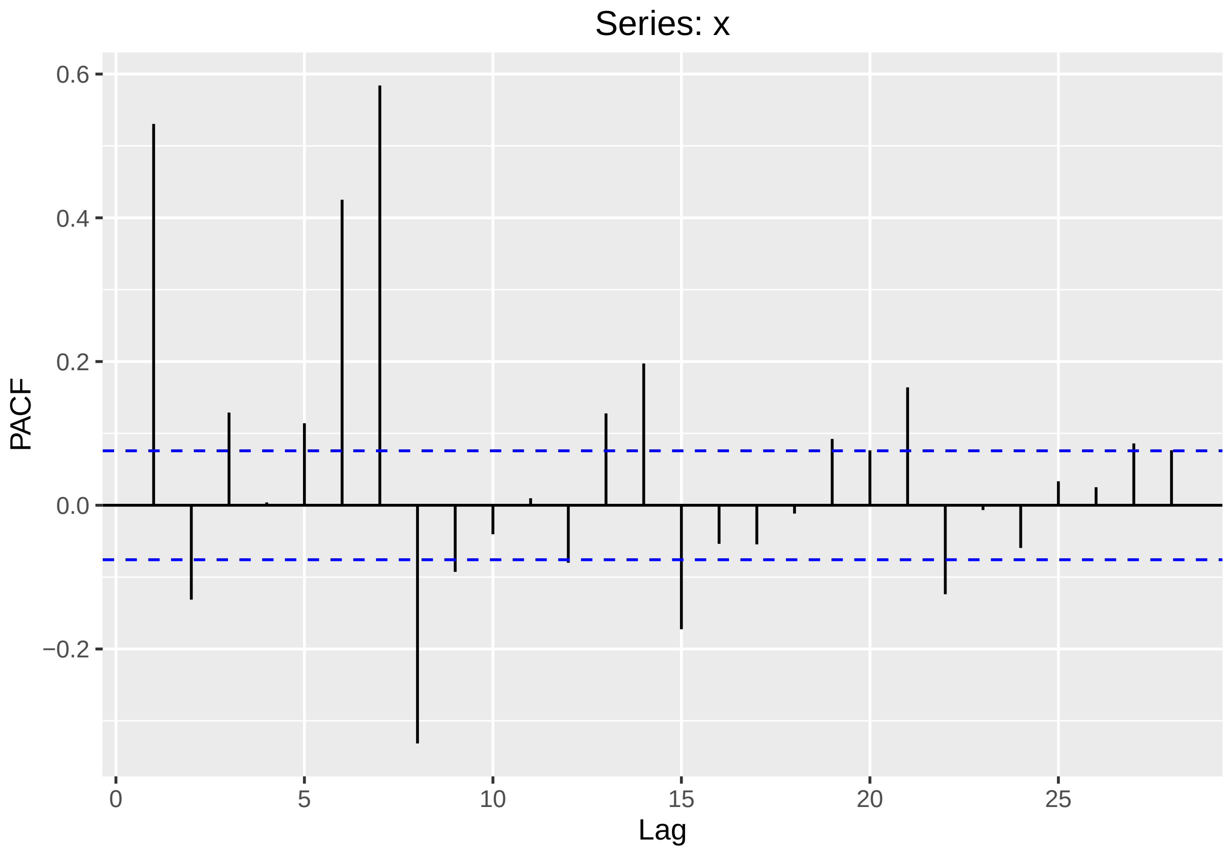
```
## # A tibble: 6 x 4
##         Day NumVehicles   Low  High
##       <chr>       <chr> <chr> <chr>
## 1 01-Sep-10        <NA>  <NA>  <NA>
## 2 02-Sep-10        <NA>  <NA>  <NA>
## 3 03-Sep-10        <NA>  <NA>  <NA>
## 4 04-Sep-10        <NA>  <NA>  <NA>
## 5 05-Sep-10        <NA>  <NA>  <NA>
## 6 06-Sep-10        <NA>  <NA>  <NA>
```

```r
vehicles_test$Day<-dmy(vehicles_test$Day)

vehicles_train <- vehicles_train %>%
  mutate(diff7 = append(diff(NumVehicles,7),rep(0,7))) %>%
  mutate(diff1 = append(diff(NumVehicles,1),rep(0,1))) %>%
```

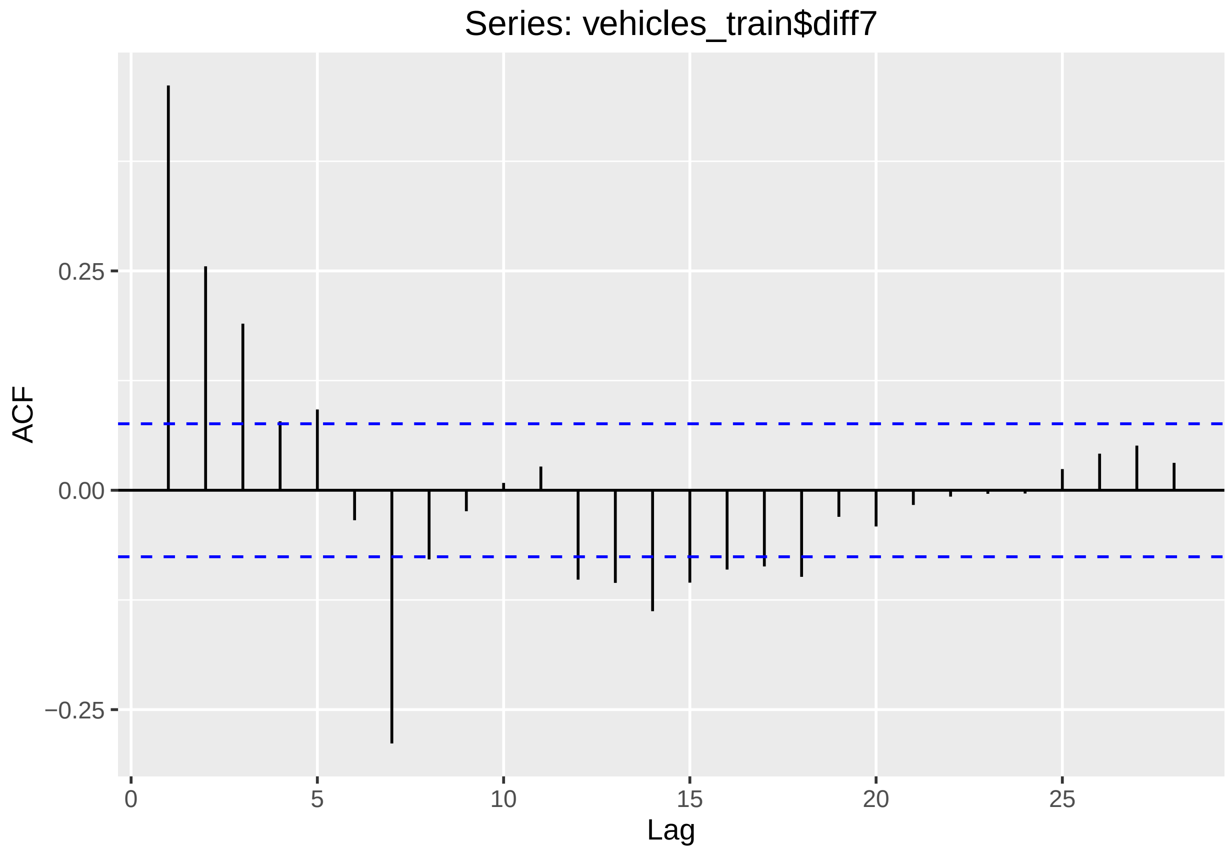
```r
  mutate(diff6 = append(diff(NumVehicles,6),rep(0,6)))

vehicles_train <- vehicles_train %>%
  mutate(diff8 = append(diff(vehicles_train$diff7,1),rep(0,1)))

# Auto correlation of vehicles train
ggAcf(vehicles_train$NumVehicles)
```



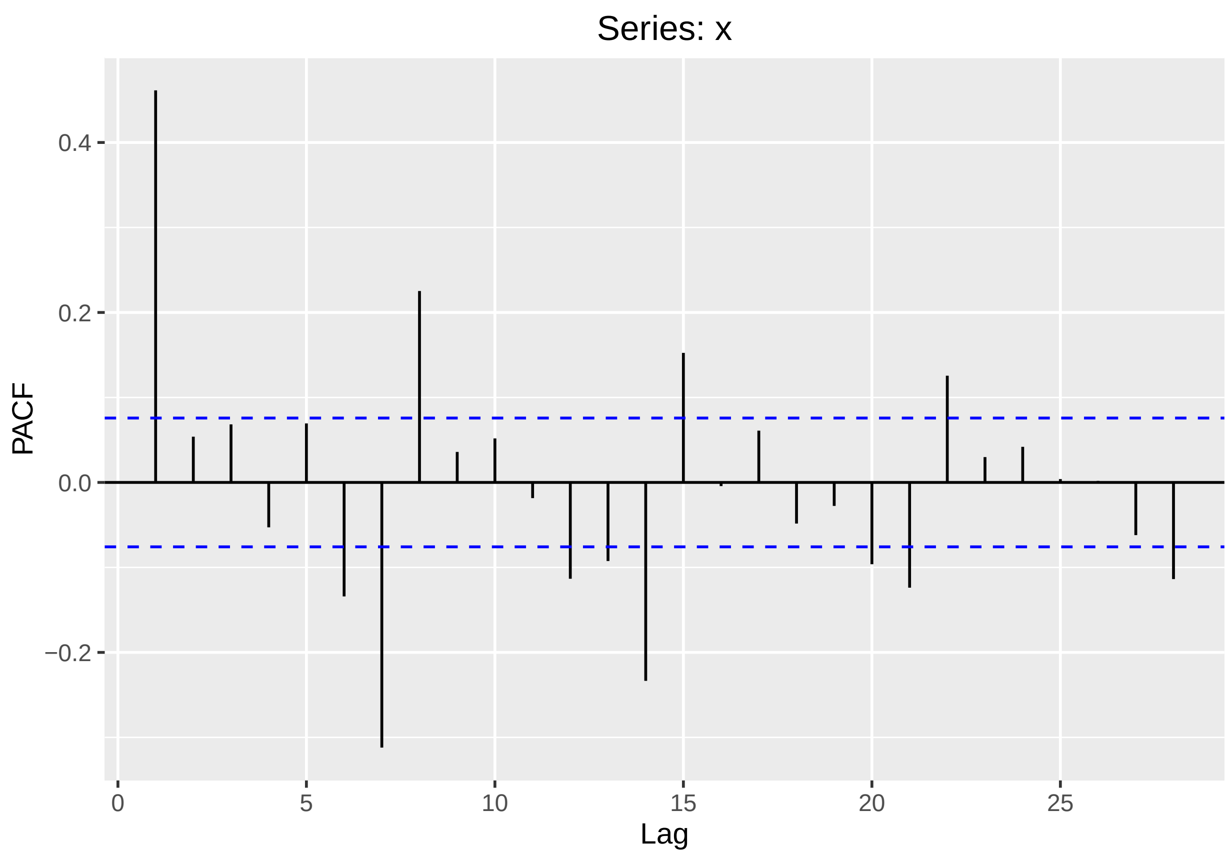

```r
# Partial auto correlation of vehicles train
ggPacf(vehicles_train$NumVehicles)
```




```r
# ACF of the diff7
ggAcf(vehicles_train$diff7)
```




```r
# PACF of the diff7
ggPacf(vehicles_train$diff7)
```




```r
head(vehicles_train)
```

```
## # A tibble: 6 x 6
##          Day NumVehicles  diff7  diff1  diff6 diff8
##       <date>       <int>  <dbl>  <dbl>  <dbl> <dbl>
## 1 2008-11-01      414144 -26920 -45940  38300  5960
## 2 2008-11-02      368204 -20960  34976  19020 14556
## 3 2008-11-03      403180  -6404   6228 -55936  6008
## 4 2008-11-04      409408   -396  16868 -12632 -3052
## 5 2008-11-05      426276  -3448  -1140 -17264  6356
## 6 2008-11-06      425136   2908  27308  -2308 -1000
```

As you can see `vehicles_test` contains 4 columns:

- `Day` — the date of the desired forecast in the same format as above
- `NumVehicles` — your point-estimate forecast for the number of vehicles that will travel on that day
- `Low` — the lower bound of the 95% confidence interval for your forecast
- `High` — the upper bound of the 95% confidence interval for your forecast

Output:

- Please fill in the data.frame `vehicles_test` as requested. The content of this data.frame is your submission for this problem.
- You do *not* need to save `vehicles_test.csv`. Please *DO NOT* try to save any files in your code, you will just confuse the bot.

Grading:

- I have the test set safely hidden in my possession and your submission will be evaluated based on that test set. This is pure out-of-sample evaluation.

  - Use cross-validation!
  - Don't overfit your training data! It won't help you.

- If your forecast performs worse than either naive (last observation), mean (average value) or naive with the drift (last observation + trend), your submission will be treated as incorrect.
  - You may want to try running all these baselines models first so that you know the baselines that you should beat.
  - You should think carefully about your cross-validation procedure so that it gives you a good approximation to the test error on the out-of-sample data.
- You do need to put the code for producing your final forecast (but you do not need to report all the temporary things that you tried)
- The time limit for your Rmarkdown is *3 minutes* of CPU time.

Hints:

- Please make sure that your forecast on the test set is not accidentally shifted by one time period. Say, if you predicted traffic for Monday and accidentally put that value into Sunday instead of Monday (due to some data misalignment) - this may end up giving you a large predictive performance hit on the out-of-sample data.

```r
# Please write your code for forecast here

t <- 120 # number of days in sliced data
n <- nrow(vehicles_train)

rmse_model <- matrix(NA,(n-t)/60)
rmse_naive <- matrix(NA,(n-t)/60)
rmse_naivedrift <- matrix(NA,(n-t)/60)
rmse_mean <- matrix(NA,(n-t)/60)

train_length <- vehicles_train$Day[1]+days(t)

for(i in seq(0,(n-t-60),60))
{
  train_data <- vehicles_train$NumVehicles[vehicles_train$Day >=
                (vehicles_train$Day[1]+days(i)) &
                vehicles_train$Day < (train_length+days(i))]
  test_data <-vehicles_train$NumVehicles[vehicles_train$Day >= (train_length+days(i))
               & vehicles_train$Day < (train_length+days(i+60))]

  model_fit <- Arima(train_data, order=c(1,0,2), seasonal = list(order = c(1,0,3),
             period=7),method="ML")


  model_forecast <- forecast(model_fit, h=60)
  rmse_model[(i/60)+1] <- sqrt(mean((model_forecast[['mean']]-test_data)^2))

  naive_forecast <-rwf(train_data,h=60)
  rmse_naive[(i/60)+1] <- sqrt(mean((naive_forecast[['mean']]-test_data)^2))

  naivedrift_forecast <-rwf(train_data,drift=TRUE,h=60)
  rmse_naivedrift[(i/60)+1] <- sqrt(mean((naivedrift_forecast[['mean']]-test_data)^2))
```

```r
  mean_forecast <-meanf(train_data,h=60)
  rmse_mean[(i/60)+1] <- sqrt(mean((mean_forecast[['mean']]-test_data)^2))

}


mean(rmse_model,na.rm = TRUE)
```

```
## [1] 34000.77
```

```r
mean(rmse_naive,na.rm = TRUE)
```

```
## [1] 56755.44
```

```r
mean(rmse_naivedrift,na.rm = TRUE)
```

```
## [1] 65138.06
```

```r
mean(rmse_mean,na.rm = TRUE)
```

```
## [1] 49406.21
```

```r
# Please write your code for forecast here

final_model <- Arima(vehicles_train$NumVehicles, order=c(1,0,2),
                seasonal = list(order = c(1,0,3), period=7),method="ML")

test_forecast <- forecast(final_model,h = 61)

vehicles_test$NumVehicles <- test_forecast$mean
vehicles_test$Low <- test_forecast$lower[,2]
vehicles_test$High <- test_forecast$upper[,2]

# PLEASE DO NOT SAVE ANY FILES!
```