Problem 3

HW3

Hanyuan Chi(chixx104), Zhi Shen(shenx704)

February 22, 2017

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

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
```

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

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
```

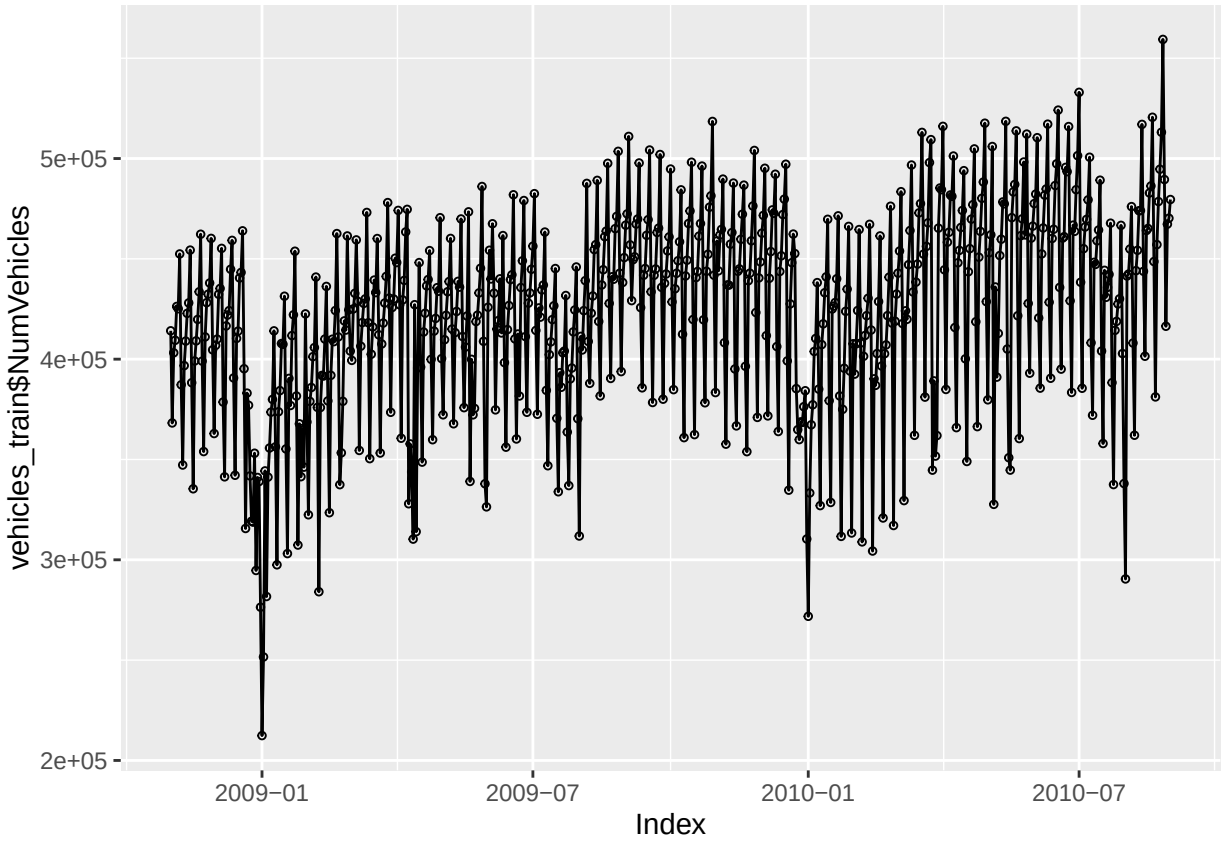
```
vehicles_train$Day <- dmy(vehicles_train$Day)
require('xts')
```

```
## Loading required package: xts
```

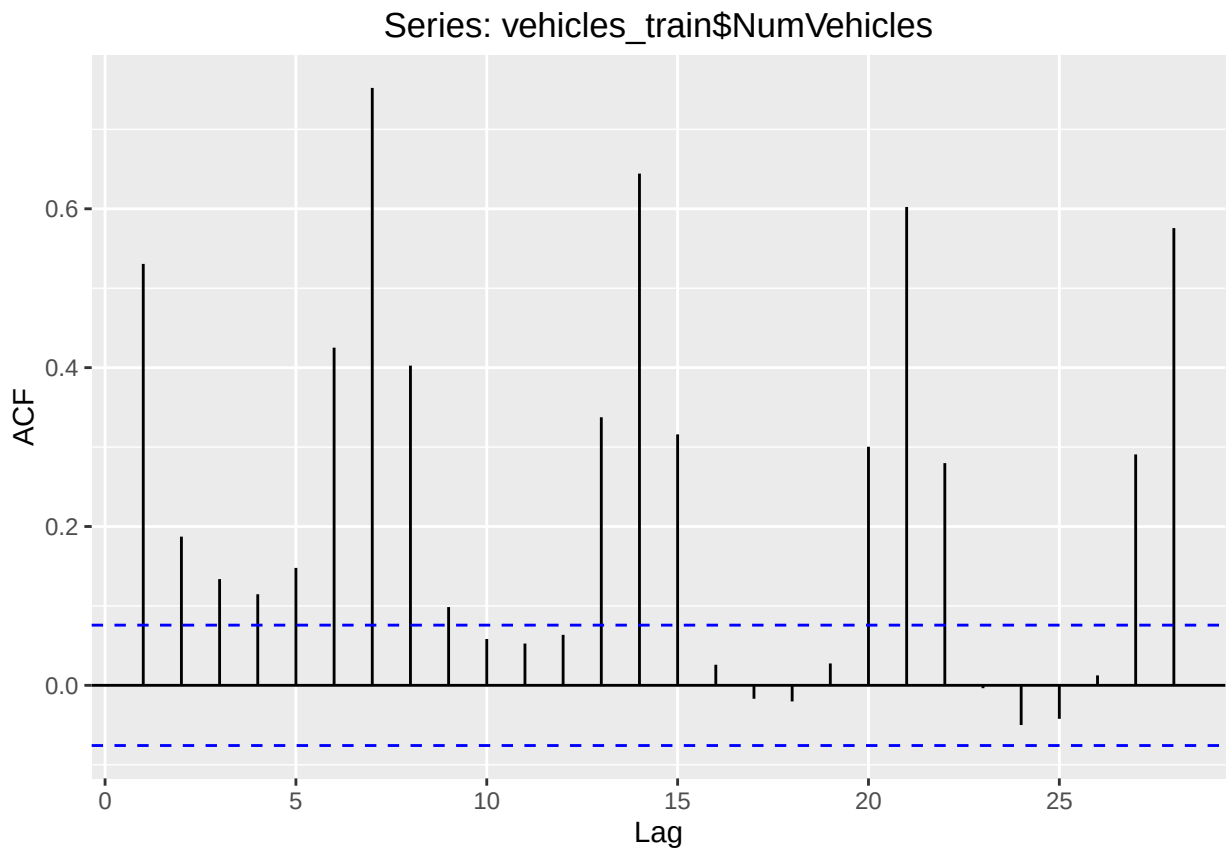
```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

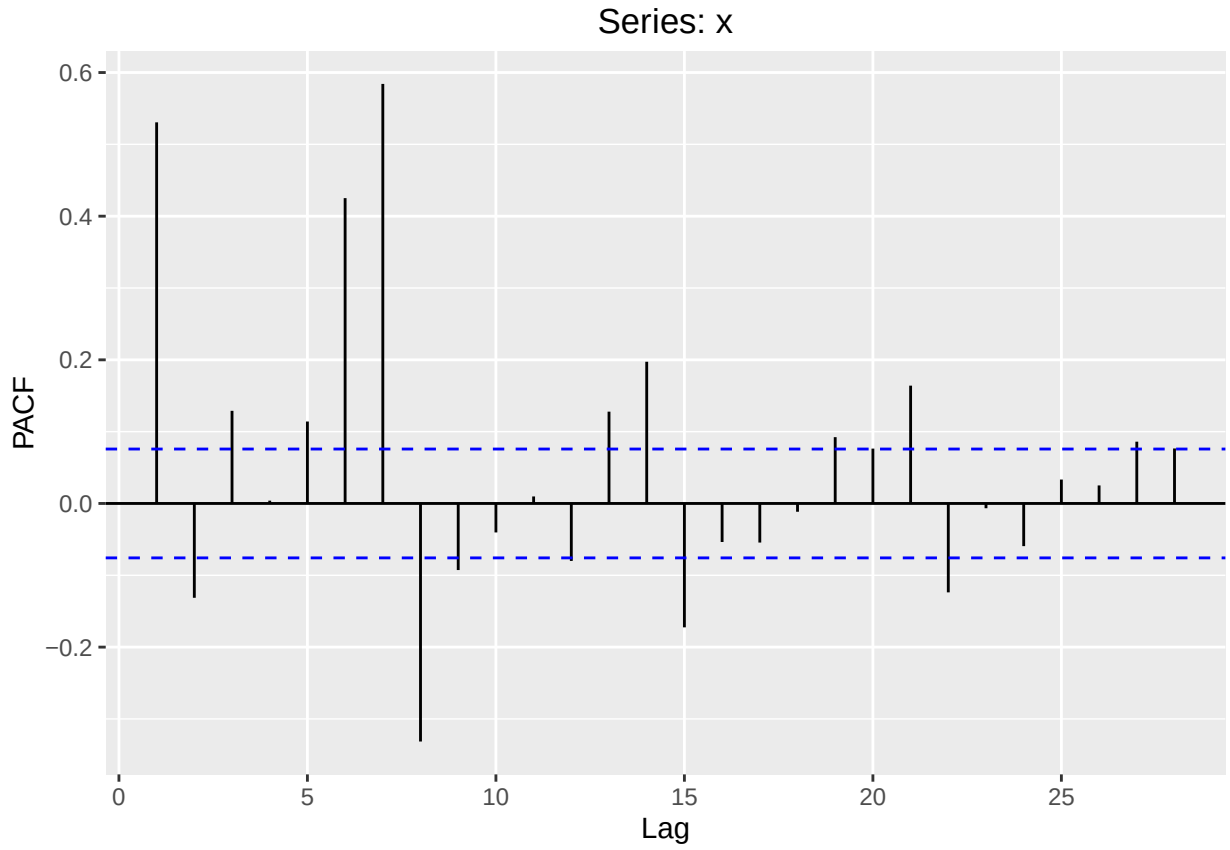
```
vehicles_train$NumVehicles <- xts(vehicles_train$NumVehicles, vehicles_train$Day)

# Find underlying process
autoplot(vehicles_train$NumVehicles) + geom_point(shape = 1, size = 1)
```



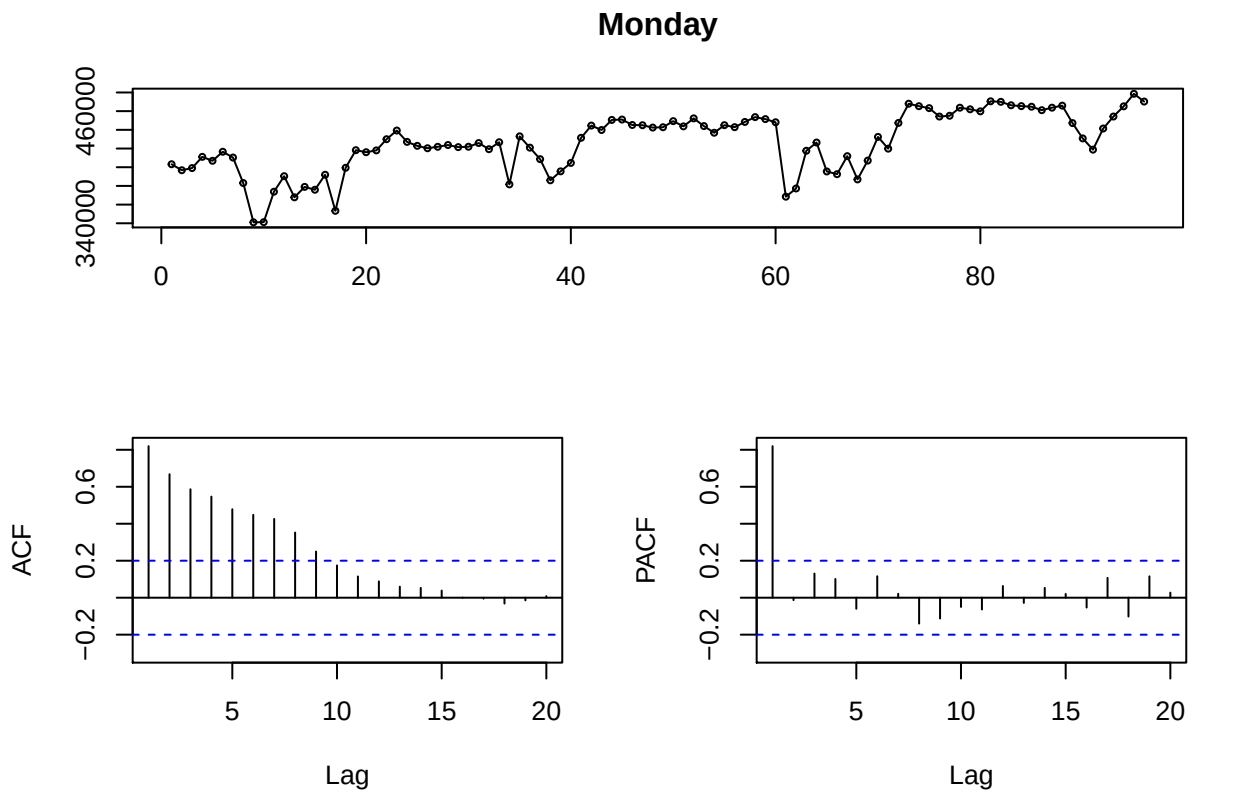

```
ggAcf(vehicles_train$NumVehicles)
```



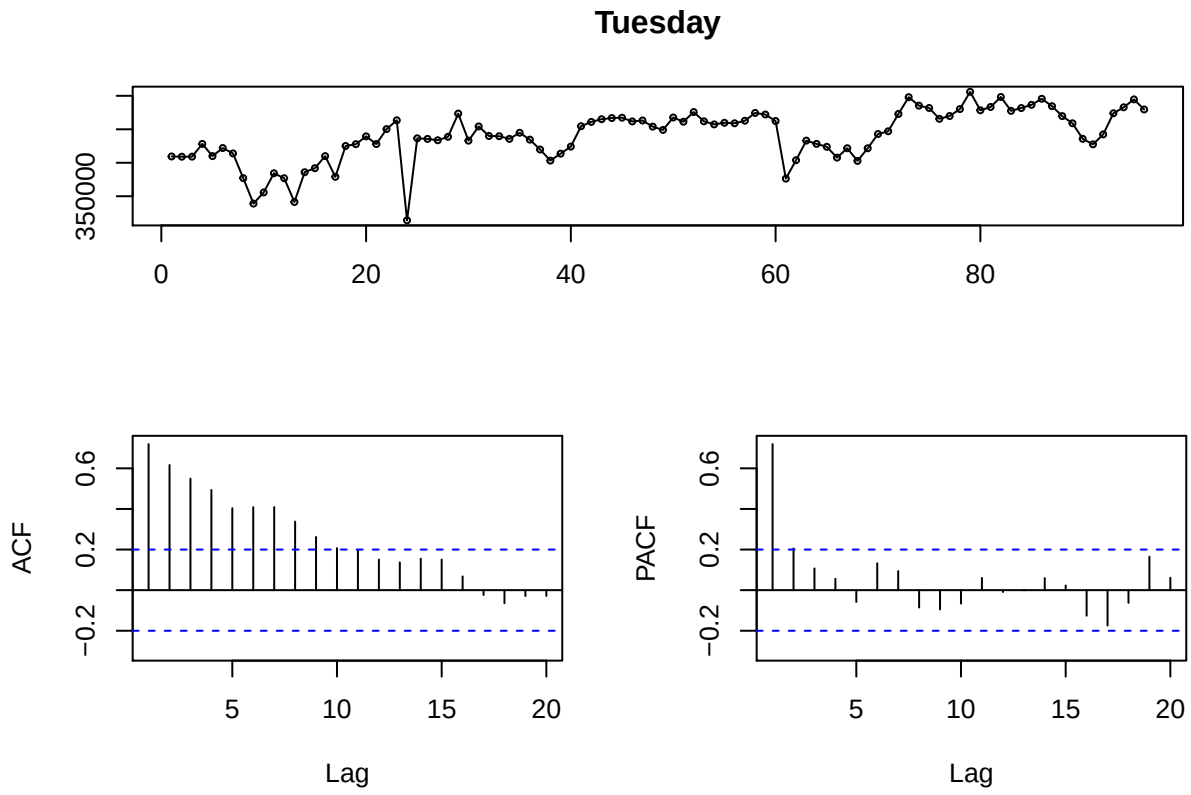

```
ggPacf(vehicles_train$NumVehicles)
```



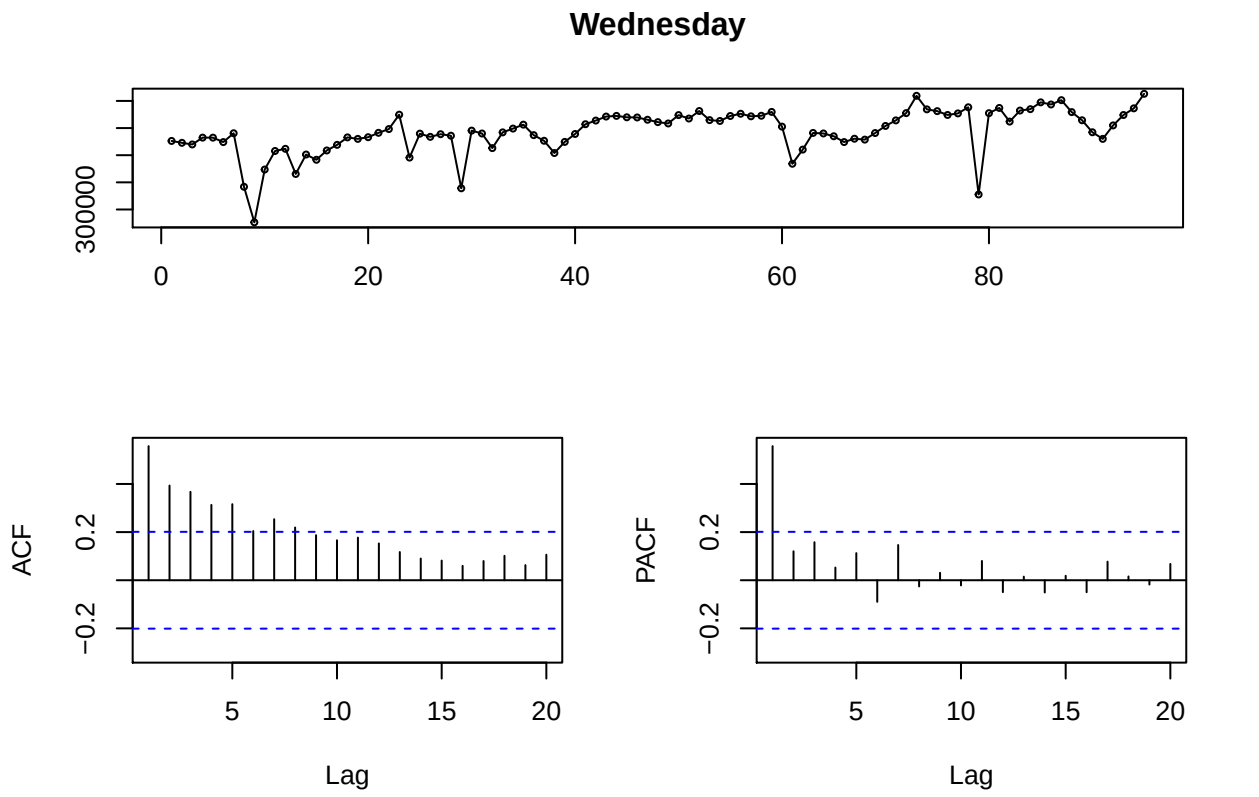

```
# The process has a seasonal component with period of 7 days

vehicles_train$weekday <- weekdays(vehicles_train$Day)

Monday <- vehicles_train[vehicles_train$weekday == 'Monday',]$NumVehicles
Tuesday <- vehicles_train[vehicles_train$weekday == 'Tuesday',]$NumVehicles
Wednesday <- vehicles_train[vehicles_train$weekday == 'Wednesday',]$NumVehicles
Thursday <- vehicles_train[vehicles_train$weekday == 'Thursday',]$NumVehicles
Friday <- vehicles_train[vehicles_train$weekday == 'Friday',]$NumVehicles
Saturday <- vehicles_train[vehicles_train$weekday == 'Saturday',]$NumVehicles
Sunday <- vehicles_train[vehicles_train$weekday == 'Sunday',]$NumVehicles

tsdisplay(Monday)
```



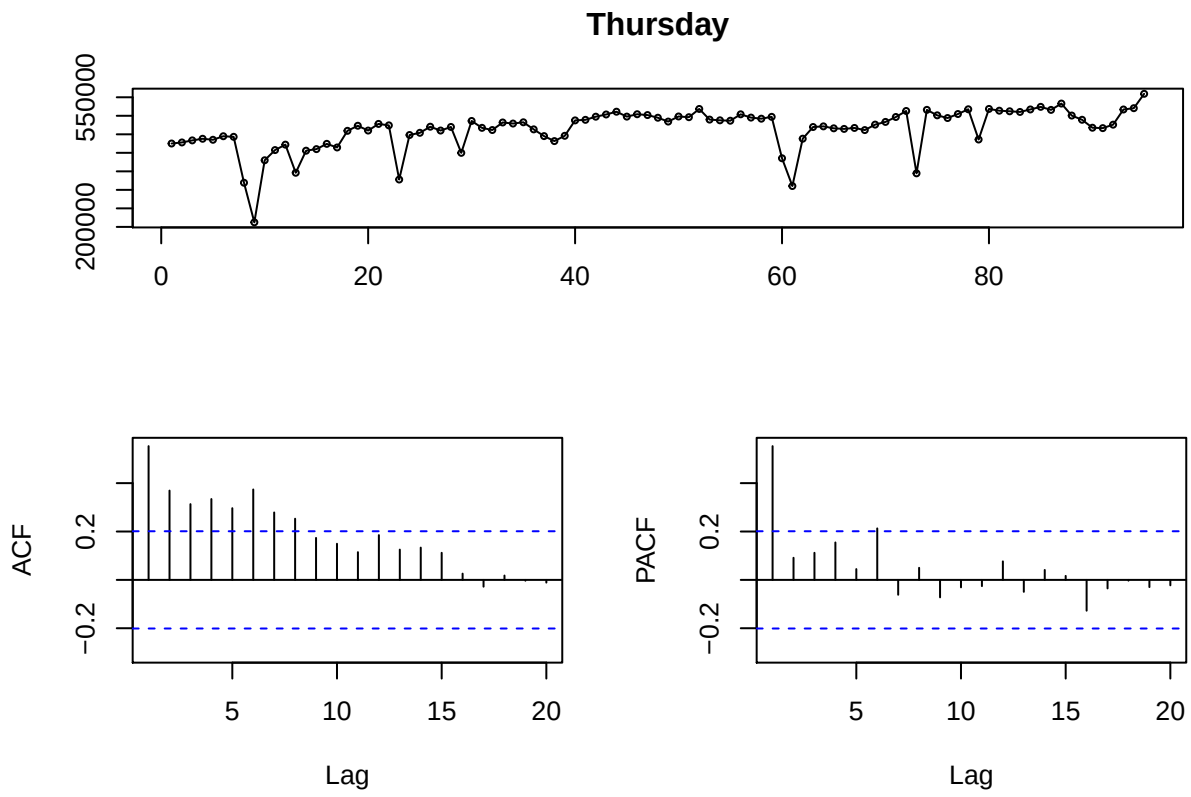

```
tsdisplay(Tuesday)
```




```
tsdisplay(Wednesday)
```




```
tsdisplay(Thursday)
```




```
# Friday, Saturday and Sunday are similar to the first four days

adf.test(Monday, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Monday
## Dickey-Fuller = -3.2358, Lag order = 4, p-value = 0.0863
## alternative hypothesis: stationary
```

```
adf.test(Tuesday, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Tuesday
## Dickey-Fuller = -3.527, Lag order = 4, p-value = 0.04362
## alternative hypothesis: stationary
```

```
adf.test(Wednesday, alternative = "stationary")
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  Wednesday
## Dickey-Fuller = -3.6707, Lag order = 4, p-value = 0.03099
## alternative hypothesis: stationary
```

```
# adf.test on Thursday, Friday, Saturday and Sunday also show the seasonal part is stationary.

eacf(Tuesday)
```

```
## 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  o  o
## 1 x o o o o o o o o o o  o  o  o
## 2 x o o o o o o o o o o  o  o  o
## 3 x o o o o o o o o o o  o  o  o
## 4 x o o o o o o o o o o  o  o  o
## 5 x o x x o o o o o o o  o  o  o
## 6 x x o o o o o o o o o  o  o  o
## 7 x x o o o o o o o o o  o  o  o
```

```
eacf(Wednesday)
```

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

```
eacf(Friday)
```

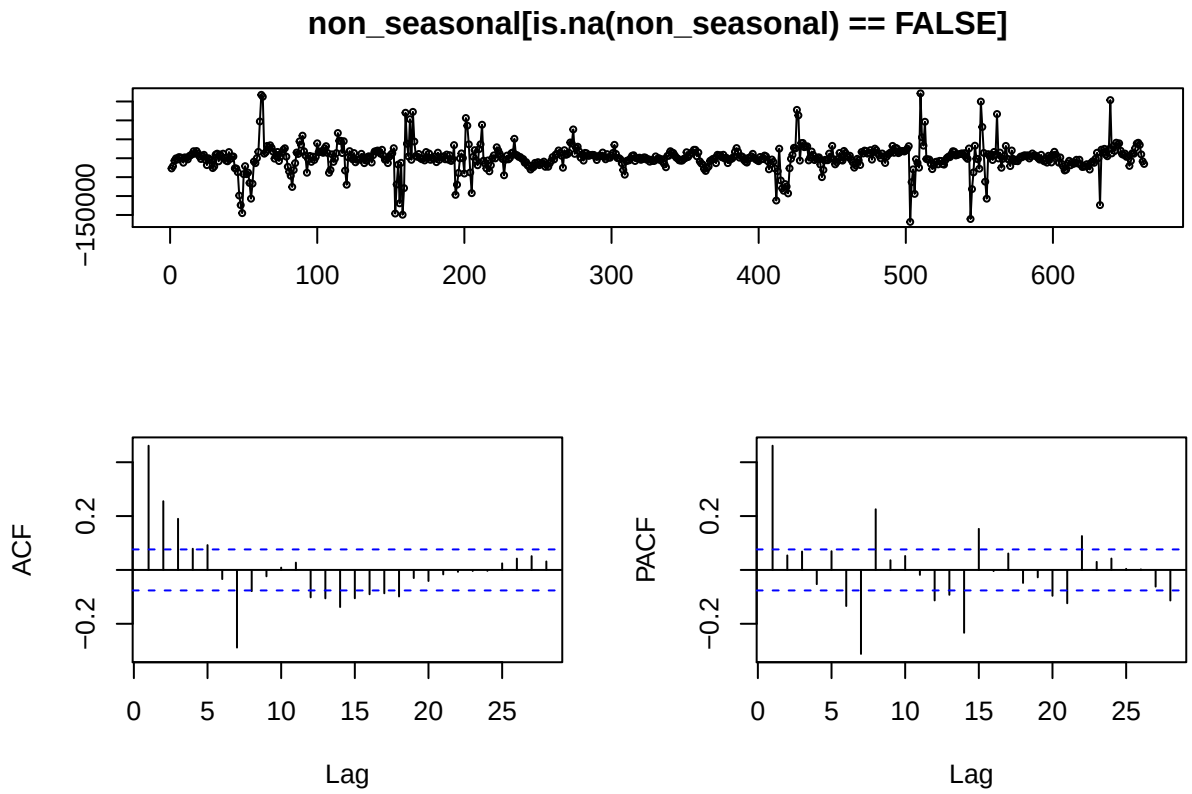
```
## AR/MA
##   0 1 2 3 4 5 6 7 8 9 10 11 12 13
## 0 x o o o o o o o o o o  o  o  o
## 1 x o o o o o o o o o o  o  o  o
## 2 x o o o o o o o o o o  o  o  o
## 3 x o o o o o o o o o o  o  o  o
## 4 x x x o o o o o o o o  o  o  o
## 5 x x x o o o o o o o o  o  o  o
## 6 x x o x o o o o o o o  o  o  o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

```
eacf(Sunday)
```

```
## 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 x x  o  o  o
## 1 x o o o o o x o o o o  o  o  o
## 2 x x o o o o x o o o o  o  o  o
## 3 x x o o o o x o o o o  o  o  o
## 4 x o o o o o o o o o o  o  o  o
## 5 x o o o o o o o o o o  o  o  o
## 6 o o o o o o o o o o o  o  o  o
## 7 x x x o o o o o o o o  o  o  o
```

```
# EACF indicates SARMA(1,1)
```

```r
# Non-seasonal part
non_seasonal <- diff(vehicles_train$NumVehicles,7)
tsdisplay(non_seasonal[is.na(non_seasonal) == FALSE])
```




```r
adf.test(non_seasonal[is.na(non_seasonal) == FALSE], alternative = "stationary")
```

```
## Warning in adf.test(non_seasonal[is.na(non_seasonal) == FALSE], alternative
## = "stationary"): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  non_seasonal[is.na(non_seasonal) == FALSE]
## Dickey-Fuller = -8.046, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

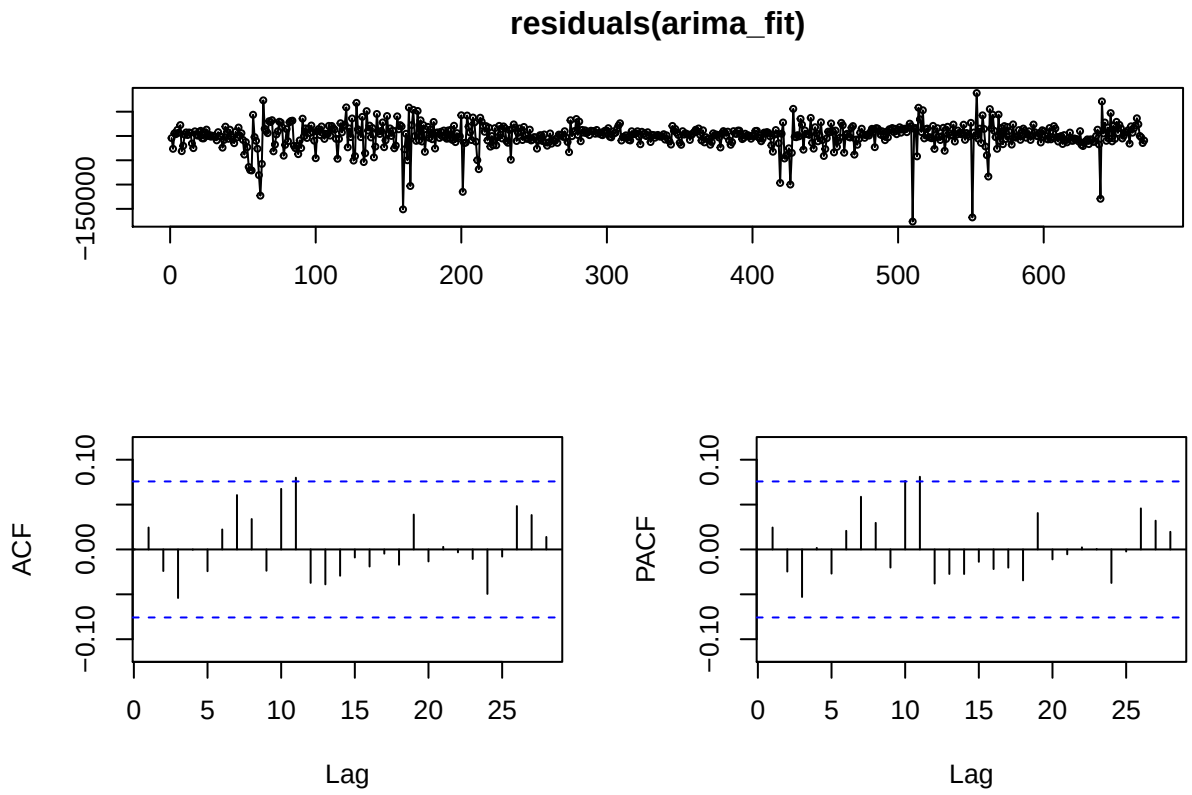
```r
eacf(non_seasonal[is.na(non_seasonal) == FALSE])
```

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

```r
arima_fit <- Arima(vehicles_train$NumVehicles, order = c(2,0,2),
              seasonal = list(order = c(1,0,1), period = 7))
summary(arima_fit)
```

```
## Series: vehicles_train$NumVehicles
## ARIMA(2,0,2)(1,0,1)[7] with non-zero mean
##
## Coefficients:
##          ar1     ar2     ma1      ma2    sar1     sma1  intercept
##       0.0525  0.6989  0.3996  -0.4452  0.9878  -0.7744  425524.11
## s.e.  0.1267  0.0888  0.1323   0.0647  0.0058   0.0352   42477.05
##
## sigma^2 estimated as 639301377:  log likelihood=-7734.59
## AIC=15485.19   AICc=15485.41   BIC=15521.24
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 488.5202 25151.78 15849.85 -0.3115586 4.054992 0.4198129
##                     ACF1
## Training set 0.02443978
```

```r
tsdisplay(residuals(arima_fit))
```




```r
Box.test(residuals(arima_fit), type="Ljung")
```

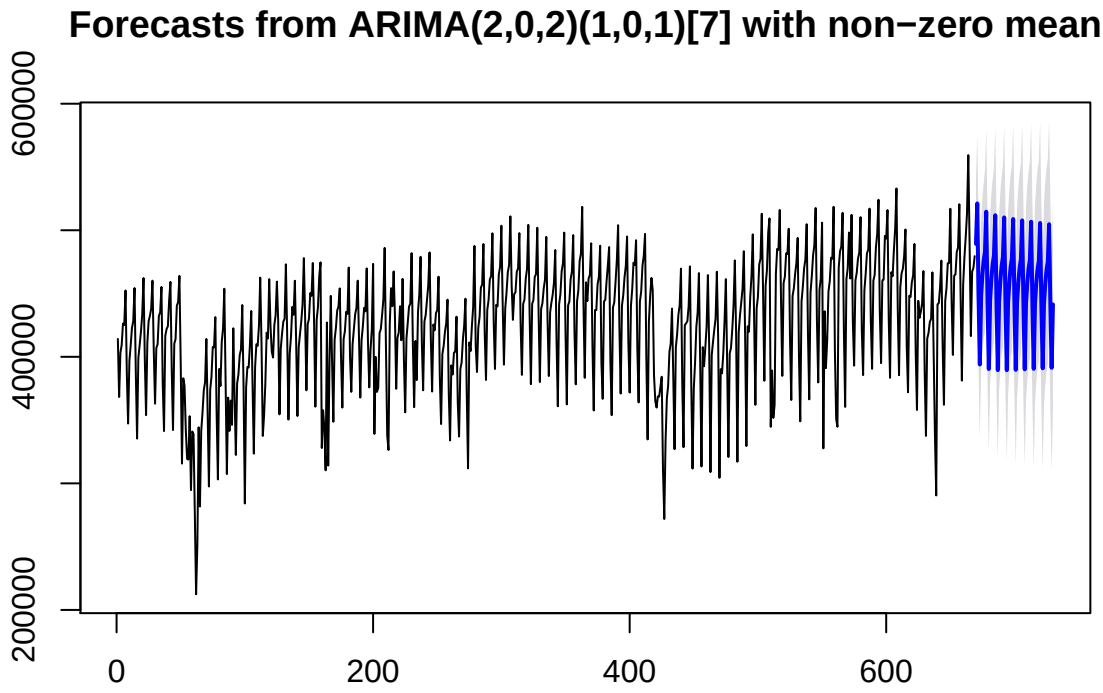
```
##
##  Box-Ljung test
```

```
##
## data:  residuals(arima_fit)
## X-squared = 0.40139, df = 1, p-value = 0.5264
```

```
# Test accuracy -- cross validation
N = 61L

L = length(vehicles_train$NumVehicles)/5

runArima <- function(ts)
 {fit <- Arima(head(ts,-N), order = c(2,0,2),
               seasonal = list(order = c(1,0,1), period = 7))
  return(accuracy(forecast(fit, h=N), tail(ts,N))[2,])
}

acc.cv <- data.frame(ME=rep(NA,5),RMSE=rep(NA,5),MAE=rep(NA,5),
                     MPE=rep(NA,5),MAPE=rep(NA,5),MASE=rep(NA,5),ACF=rep(NA,5))

for (i in 0:4)
{vehicles_sub <- vehicles_train$NumVehicles[(i*L+1):((i+1)*L)]
 accur <- sapply(vehicles_sub, FUN = runArima)
 acc.cv[(i+1),] <- accur
}

options(scipen = 50)
round(colMeans(acc.cv),5)
```

```
##          ME        RMSE         MAE         MPE        MAPE        MASE
## 11290.38894 32505.82573 25984.21389     2.24624     6.13306     0.69642
##         ACF
##          NA
```

```
# prediction
k <- 61L
arima_forecast <- forecast(arima_fit,h=k, level=c(95))
plot.forecast(arima_forecast)
```



- 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:

```
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()
## )
```

```
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>
```

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.

```
# Please write your code for forecast here

vehicles_test$NumVehicles <- arima_forecast$mean
vehicles_test$Low <- arima_forecast$lower
vehicles_test$High <- arima_forecast$upper
```

```
# PLEASE DO NOT SAVE ANY FILES!
```