Seoul Bike Rental in 2018

Group 23: Ming Zhao, Walid El Mouahidi, & Alfonso Larumbe Fernández

03/17/2022

Introduction

Every day in Seoul millions of people leave their homes to commute to their jobs, schools or just to enjoy being outdoors. In the highly connected society of large cities such as Seoul, many of the goods that were once considered personal are now commonly shared. That is the case with transportation vehicles such as bicycles. This new type of economy gave birth to companies that would take care of these shared goods.

Our final project will put us in the shoes of a company that coordinates bicycle rentals in the city of Seoul, and our goal is to help them efficiently predict the number of bikes that will be rented each day. There are two things to keep in mind: we do not have an infinite number of bikes, and getting them out on the street and picking them up costs money.

Therefore, to save them some money and effort, we will find an optimal model to predict the number of bikes that will be rented on any given day. For this purpose, we will be using a data set containing information on the hourly bike rental in the city of Seoul in the year 2018. This data set also includes weather information, which is evidently highly correlated with our goal variable.

Proposed Method

To carry out this study, we decided to take two different approaches:

- The first approach considers each record independently. That is, the number of rented bikes depends only on data about the time (hour, day of the week, holiday, season…) and current weather (temperature, humidity, wind speed, rain…).
- The second approach will make use of time methods. It assumes that the data depends on the polynomial time trend and there are dependencies within the data.

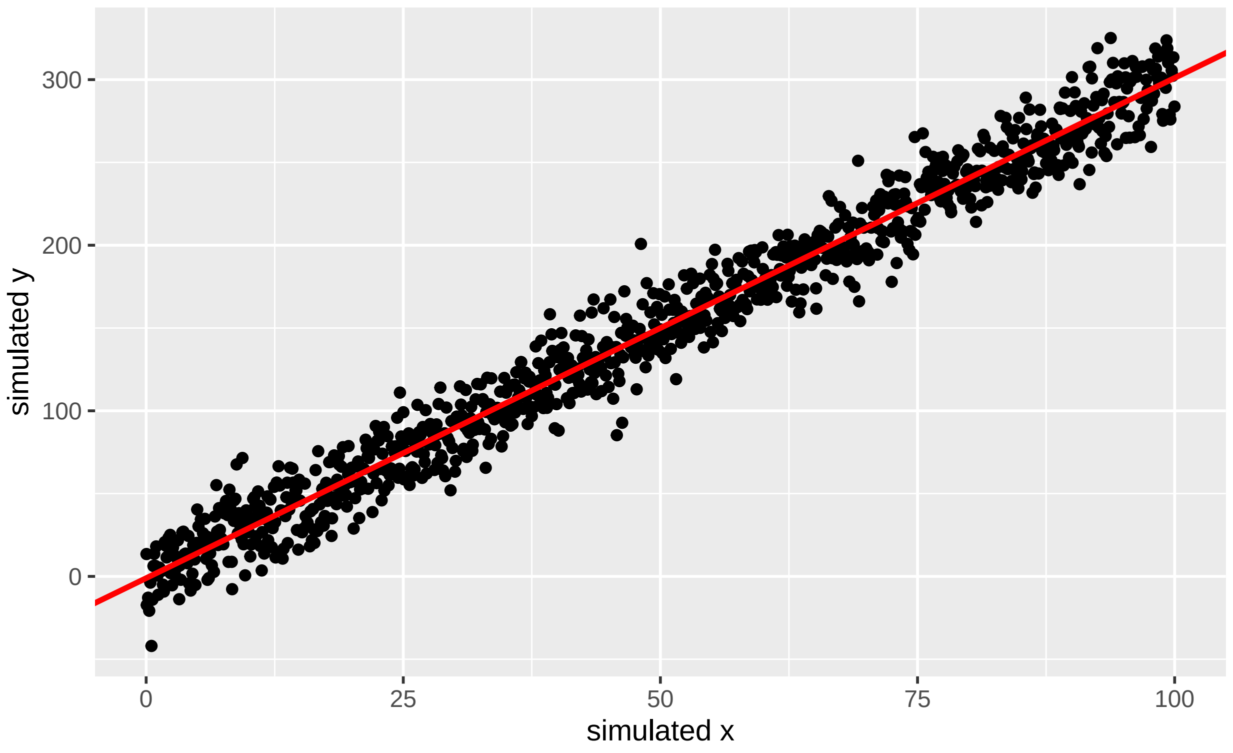
Simulation Study

Before we start working with the real data, we want to make sure that our functions work properly. This whole project is based on linear regression, so we created our own version of the OLS method using Cholesky decomposition (given that it was the most efficient method as it was proved in the homework). To test our function, we will generate some data randomly following the function $y = 3x$ plus some normal error term of mean 0, to preserve all the assumed properties of linear regression.

```r
# generate simulated data
set.seed(2022)

x_ = as.matrix(seq(0,100,0.1))
y_ = 3*x_ + rnorm(length(x_),mean = 0, sd = 15)

x_1 = as.matrix(rep(1,length(x_)))
x_ = cbind(x_1, x_)
y_ = as.matrix(y_)

# lm coefficient function
ols.coef <- function(y, X) {

  #get number of observations and parameters
  n <- nrow(X)
  p <- ncol(X)

  #compute coefficients
  beta <- chol2inv(chol(crossprod(X)))%*%crossprod(X,y)
  colnames(beta) <- "beta"

  #compute sigma
  e = y - X%*%beta
  sigma = sum(e*e)/(n-p)

  #compute se
  V <- chol2inv(chol(crossprod(X))) * sigma
  se <- sqrt(diag(V))

  #compute p-value
  pvalue <- round(2*pt(abs(beta/se), n,lower.tail = FALSE),5)
  colnames(pvalue) <- "p.value"

  #return result
  result <- list(coefs=cbind(X=colnames(X), round(beta,3), se=round(se,3), round(pvalue,3)),
                 sigma=sigma)
  return(result)
}

# simulation test
coef = ols.coef(y_,x_)
coef
```

```
## $coefs
##        beta    se p.value
## [1,] -1.064 0.946   0.261
## [2,]  3.021 0.016   0.000
##
## $sigma
## [1] 224.0915
```

We can see that our function predicted $\hat{\beta} = 3.021$, which is a very accurate prediction given that we generated our data as $y_i = 3x_i + \epsilon_i$. Since this was a very simple example, we can plot these points and our predicted function in red:



Other functions that we built and used in this project are the following:

```
# lm residual function: Computes the residual for a given linear regression model
ols.resid <- function(y, X) {

  #compute coefficients
  beta <- chol2inv(chol(crossprod(X)))%*%crossprod(X,y)

  #compute residuals
  e = y - X%*%beta

  #return residuals
  return(e)
}

# lm prediction function: Given y and X, returns the fitted values for the y using the OLS
ols.pred <- function(y, X) {

  #compute coefficients
  beta <- chol2inv(chol(crossprod(X)))%*%crossprod(X,y)

  #compute predicted values
  fit = as.matrix(X %*% beta)

  #return predicted values
  return(fit)
}
```

Real Data Study

We use “Seoul Bike Sharing Demand Data Set” for this project. The data set contains hourly data for one year, so the sample size is $365 \times 24 = 8760$. There are 14 variables in the original data set:

- Date : year-month-day
- Rented Bike count: count of bikes rented at each hour
- Temperature: in Celsius
- Humidity: in percentage
- Windspeed: in m/s
- Visibility: in 10m
- Dew point temperature: in Celsius
- Solar radiation: in MJ/m2
- Rainfall: in mm
- Snowfall: in cm
- Hour: from 0 to 23 hour of the day
- Seasons: Winter/Spring/Summer/Autumn
- Holiday: Holiday/Non holiday
- Functional Day: NoFunc(Non Functional Hours)/Fun(Functional hours)

The outcome variable is the count of bikes rented. Explanatory variables have both continuous and categorical variables. Categorical variables include Hour, Seasons, Holiday, and Functional Day; we recode these variables into dummies for the matrix computation. Additionally, we add the day of the week, Monday to Sunday, to the data set. After constructing all the variables we use, we delete the date variable. For the purpose of computation, we remove the reference level for every categorical variable: 0 hour, Monday, Winter, Non holiday, Non functional hours. As a consequence, the final data set has 43 variables in total including the outcome.

The time series of original rented bike count have serious fluctuations as the Figure 0a shows, so we decide to use 7-day moving average as our outcome for analysis. Basically, we add up counts from the last 7 days, which are $24 \times 7 = 168$ hours or observations and divide it by 168. The new outcome as shown becomes smoother. We then take 7-day moving average on all the continuous explanatory variables for the analysis.

Figure0a. Time Series: Original vs. 7−day Moving Average Data

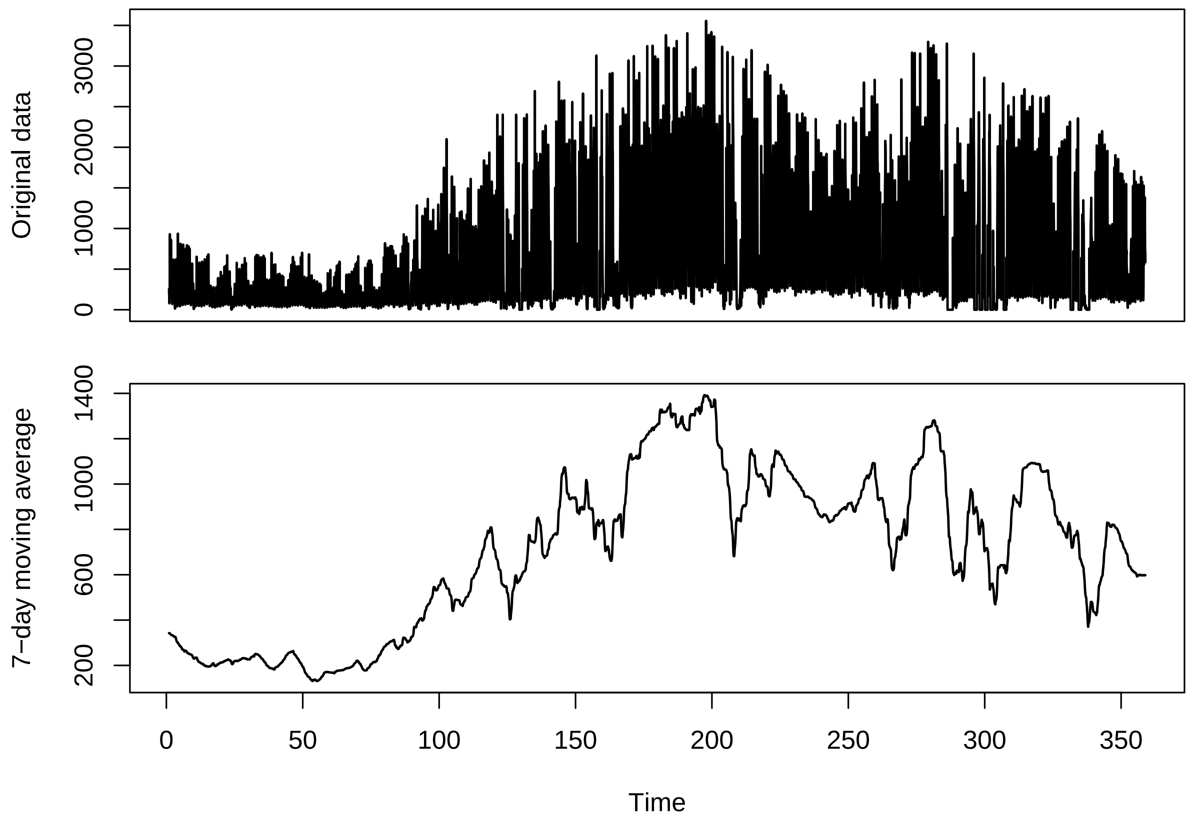
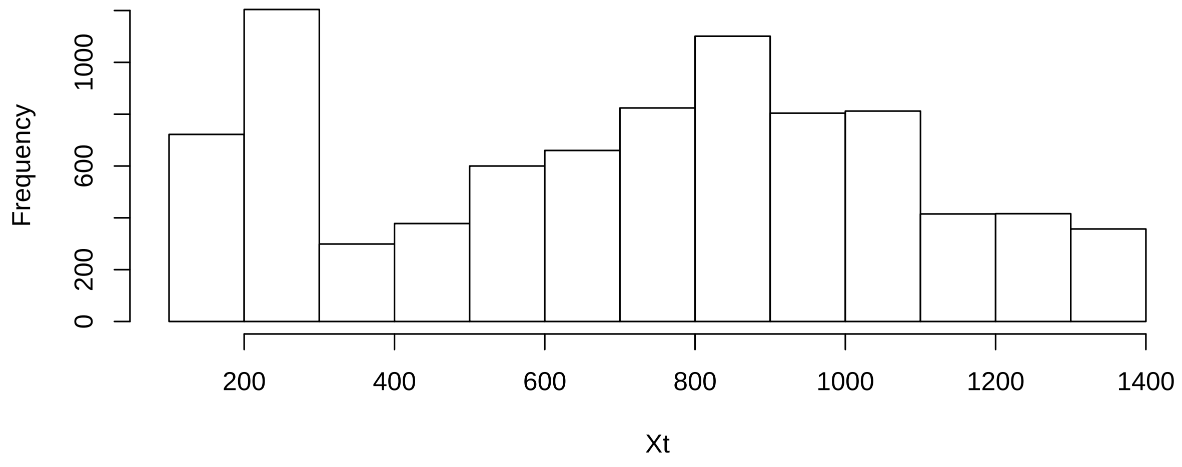


Figure0b. Distribution of 7−day Moving Average Data



Linear regression

We first fit linear regression using all the variables in the data set and use the formula below for the model computation. Xt here represents the outcome, 7-day moving average count, Z_t is a vector of all explanatory variables, α and W_t denote intercept and error term, respectively. According to the output by our function, all the continuous variables are statistically significant, but many of the categorical variables are not. The significant categorical variables include Holiday, Functional Day, and season. For our interest, we run stepwise algorithm and find the best combination of predictors using AIC, which contains all the continuous variables as well as Holiday, Functional Day, season, and Tuesday. Nevertheless, the coefficients are not far different from the two models, so we just decide to keep using the model with all variables for our predictions. Figure 1 shows the predictions by the full model and the predictions do not look bad. For residuals in Figure 2, they are not stationary; for this part, we expect improvement by applying other models. Figure 3 shows diagnostics for the linear model. From the residuals vs fitted plot, it seems that the residuals are related to the fitted values. We therefore argue that the outcome values are dependent and time series analysis may help to solve this problem.

$$X_t = \alpha + \beta Z_t + W_t$$

```
## $coefs
##       X                             beta       se        p.value
##  [1,] "constant"                    "2242.311" "124.333" "0"
##  [2,] "Temperature..C."             "-82.591"  "4.693"   "0"
##  [3,] "Humidity..."                 "-21.622"  "1.438"   "0"
##  [4,] "Wind.speed..m.s."            "-205.803" "6.599"   "0"
##  [5,] "Visibility..10m."            "-0.022"   "0.007"   "0.001"
##  [6,] "Dew.point.temperature..C."   "89.51"    "4.996"   "0"
##  [7,] "Solar.Radiation..MJ.m2."     "950.316"  "16.231"  "0"
##  [8,] "Rainfall.mm."                "21.187"   "8.432"   "0.012"
##  [9,] "Snowfall..cm."               "40.846"   "8.409"   "0"
## [10,] "Holiday"                     "-70.226"  "6.667"   "0"
## [11,] "Functioning.Day"             "92.24"    "8.201"   "0"
## [12,] "Spring"                      "115.174"  "7.621"   "0"
## [13,] "Summer"                      "259.98"   "10.388"  "0"
## [14,] "Autumn"                      "289.439"  "7.443"   "0"
## [15,] "Tuesday"                     "8.099"    "5.339"   "0.129"
## [16,] "Wednesday"                   "4.575"    "5.309"   "0.389"
## [17,] "Thursday"                    "2.442"    "5.312"   "0.646"
## [18,] "Friday"                      "3.451"    "5.288"   "0.514"
## [19,] "Saturday"                    "0.786"    "5.309"   "0.882"
## [20,] "Sunday"                      "2.539"    "5.301"   "0.632"
## [21,] "X1"                          "0.035"    "9.797"   "0.997"
## [22,] "X2"                          "0.042"    "9.797"   "0.997"
## [23,] "X3"                          "0.048"    "9.797"   "0.996"
## [24,] "X4"                          "0.05"     "9.797"   "0.996"
## [25,] "X5"                          "0.041"    "9.797"   "0.997"
## [26,] "X6"                          "0.039"    "9.797"   "0.997"
## [27,] "X7"                          "-0.2"     "9.797"   "0.984"
## [28,] "X8"                          "-0.148"   "9.797"   "0.988"
## [29,] "X9"                          "-0.148"   "9.797"   "0.988"
## [30,] "X10"                         "-0.155"   "9.797"   "0.987"
## [31,] "X11"                         "-0.174"   "9.797"   "0.986"
## [32,] "X12"                         "-0.199"   "9.797"   "0.984"
## [33,] "X13"                         "-0.223"   "9.797"   "0.982"
## [34,] "X14"                        "-0.23"    "9.797"   "0.981"
## [35,] "X15"                        "-0.247"   "9.797"   "0.98"
## [36,] "X16"                        "-0.254"   "9.797"   "0.979"
## [37,] "X17"                        "-0.223"   "9.797"   "0.982"
## [38,] "X18"                        "-0.173"   "9.797"   "0.986"
## [39,] "X19"                        "-0.122"   "9.797"   "0.99"
## [40,] "X20"                        "-0.087"   "9.797"   "0.993"
## [41,] "X21"                        "-0.055"   "9.797"   "0.996"
## [42,] "X22"                        "-0.028"   "9.797"   "0.998"
## [43,] "X23"                        "0.009"    "9.797"   "0.999"
##
## $sigma
## [1] 17181.34
```

Figure1. Predictions by Linear Regression

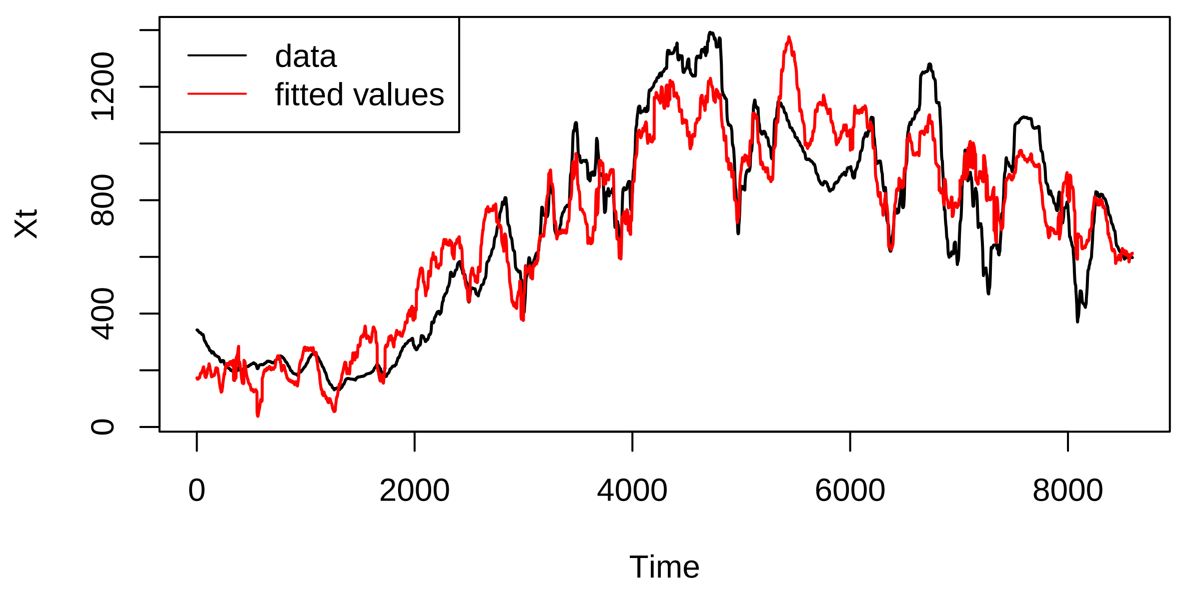


Figure2. Residuals of Linear Regression

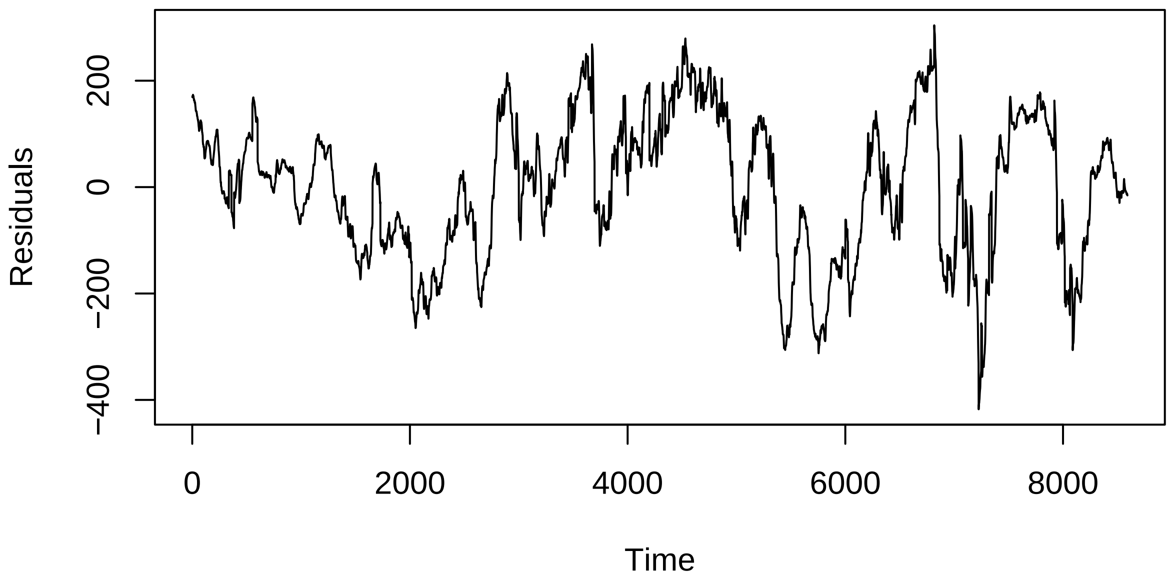
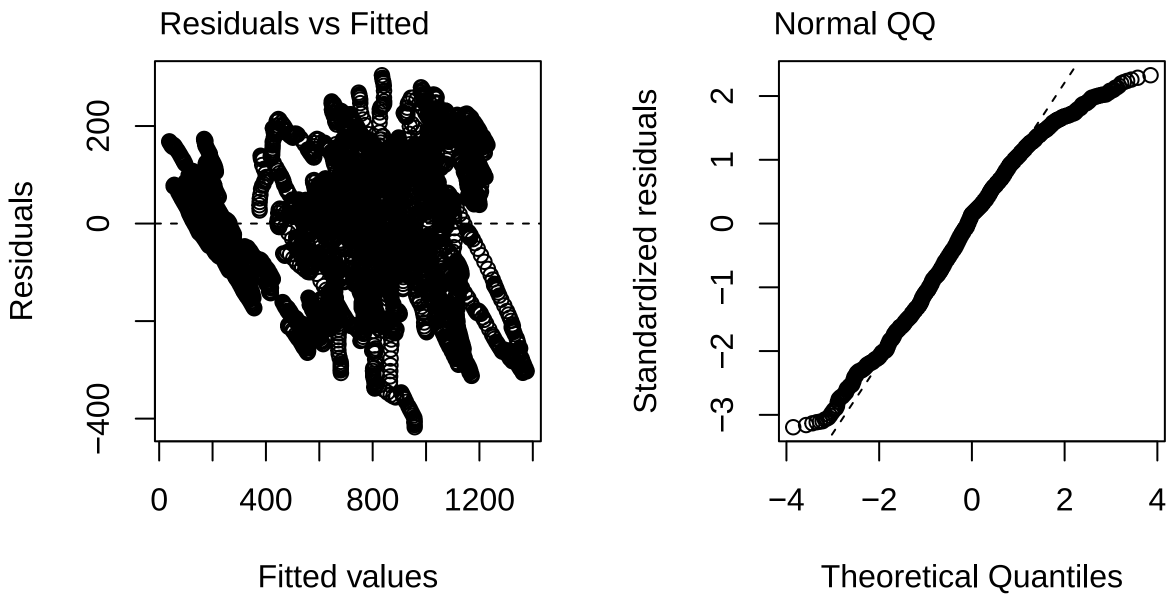


Figure3. Diagnostics for Linear Regression



Linear regression with polynomial trends

Then, we try linear regression with polynomial trends. For this method, we consider the outcome values are not independent and time matters, so we add polynomial of the time trend into the previous linear regression model. We decide to use 4 degrees since we identify 2 valleys and 1 peak from the moving average plot. The formula is constructed as below, where T denotes the time trend. The results show that the polynomial trends are statistically significant to predict the outcome. Looking at Figure 4, the predictions are closer to the observed values than those of linear regression. However, the residuals are not obviously better than the previous ones and the diagnostics do not have noticeable improvement as shown in Figure 5 and 6.

$$X_t = \alpha + \beta Z_t + \gamma_1 T + \gamma_2 T^2 + \gamma_3 T^3 + \gamma_4 T^4 + W_t$$

```
## $coefs
##       X                              beta       se       p.value
##  [1,] "constant"                     "2579.293" "96.151" "0"
##  [2,] "Temperature..C."              "-74.17"   "3.617"  "0"
##  [3,] "Humidity..."                  "-21.043"  "1.115"  "0"
##  [4,] "Wind.speed..m.s."             "-175.274" "5.308"  "0"
##  [5,] "Visibility..10m."             "-0.091"   "0.007"  "0"
##  [6,] "Dew.point.temperature..C."    "67.254"   "3.916"  "0"
##  [7,] "Solar.Radiation..MJ.m2."      "241.59"   "16.961" "0"
##  [8,] "Rainfall.mm."                 "-227.801" "7.813"  "0"
##  [9,] "Snowfall..cm."                "-52.146"  "6.76"   "0"
## [10,] "Holiday"                      "-43.696"  "5.153"  "0"
## [11,] "Functioning.Day"              "88.834"   "6.309"  "0"
## [12,] "Spring"                       "68.949"   "7.888"  "0"
## [13,] "Summer"                       "246.743"  "11.421" "0"
## [14,] "Autumn"                       "535.102"  "13.869" "0"
## [15,] "Tuesday"                      "8.909"    "4.104"  "0.03"
## [16,] "Wednesday"                    "6.693"    "4.082"  "0.101"
## [17,] "Thursday"                     "5.992"    "4.084"  "0.142"
## [18,] "Friday"                       "3.43"     "4.065"  "0.399"
## [19,] "Saturday"                     "-1.298"   "4.083"  "0.751"
## [20,] "Sunday"                       "1.164"    "4.075"  "0.775"
## [21,] "X1"                           "0.064"    "7.531"  "0.993"
## [22,] "X2"                           "0.099"    "7.531"  "0.989"
## [23,] "X3"                           "0.132"    "7.531"  "0.986"
## [24,] "X4"                           "0.165"    "7.531"  "0.983"
## [25,] "X5"                           "0.187"    "7.531"  "0.98"
## [26,] "X6"                           "0.214"    "7.531"  "0.977"
## [27,] "X7"                           "0.02"     "7.531"  "0.998"
## [28,] "X8"                           "0.112"    "7.531"  "0.988"
## [29,] "X9"                           "0.194"    "7.531"  "0.979"
## [30,] "X10"                          "0.242"    "7.531"  "0.974"
## [31,] "X11"                          "0.295"    "7.531"  "0.969"
## [32,] "X12"                          "0.35"     "7.531"  "0.963"
## [33,] "X13"                          "0.399"    "7.531"  "0.958"
## [34,] "X14"                          "0.453"    "7.531"  "0.952"
## [35,] "X15"                          "0.501"    "7.531"  "0.947"
## [36,] "X16"                          "0.553"    "7.531"  "0.941"
## [37,] "X17"                          "0.626"    "7.531"  "0.934"
## [38,] "X18"                          "0.713"    "7.531"  "0.925"
## [39,] "X19"                          "0.795"    "7.531"  "0.916"
## [40,] "X20"                          "0.861"    "7.531"  "0.909"
## [41,] "X21"                          "0.921"    "7.531"  "0.903"
## [42,] "X22"                          "0.979"    "7.531"  "0.897"
## [43,] "X23"                          "1.044"    "7.531"  "0.89"
## [44,] "trend1"                          "-20.869"  "0.336"  "0"
## [45,] "trend2"                          "0.304"    "0.004"  "0"
## [46,] "trend3"                          "-0.001"   "0"      "0"
## [47,] "trend4"                          "0"        "0"      "0"
##
## $sigma
## [1] 10152.62
```

Figure4. Predictions by Polynomial Trends

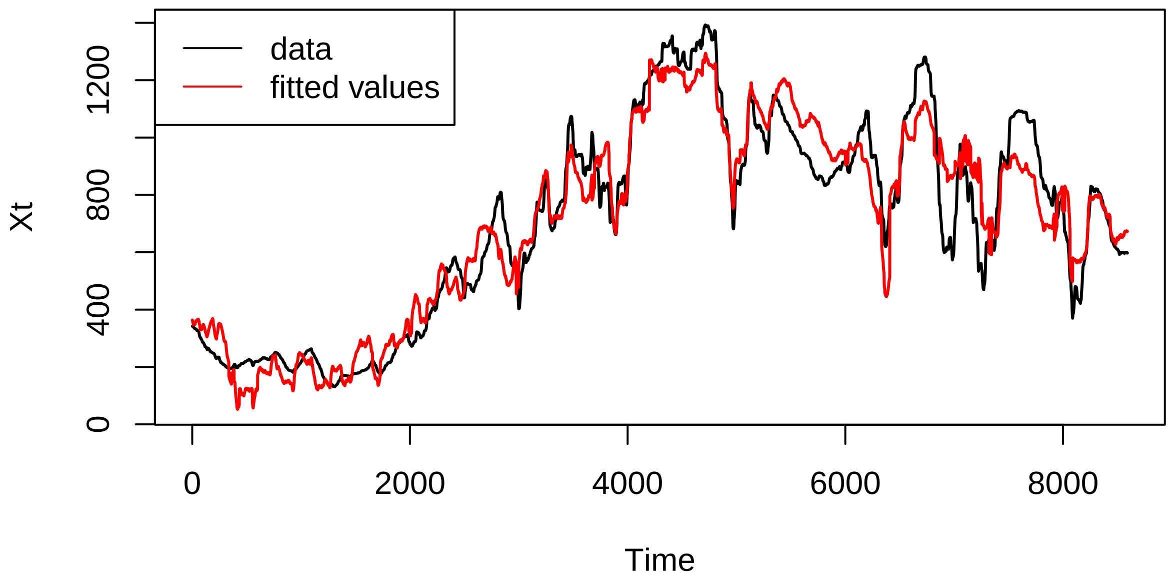


Figure5. Residuals of Polynomial Trends

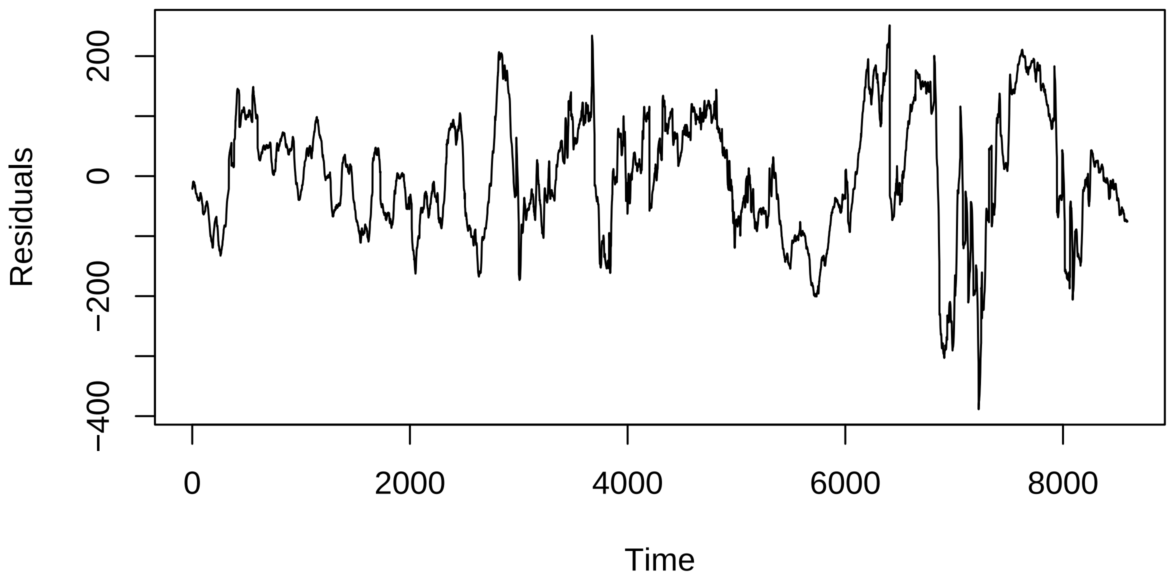
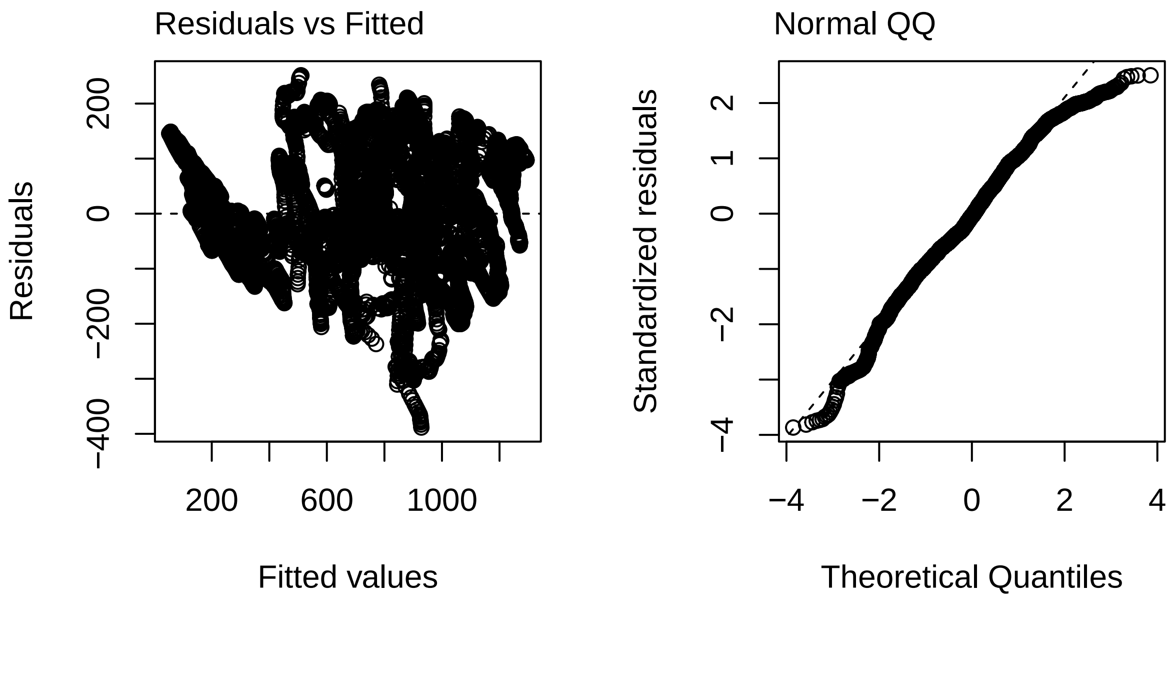


Figure6. Diagnostics for Polynomial Trends



Linear regression with polynomial trends and auto-correlated errors

To improve our predictions, residuals, and diagnostics, we consider using auto-correlated errors to the polynomial model. The idea is that we regress the error term from the polynomial model on its lags. The formula is given below. As stated, W_t is the error term from the polynomial model, $W_{t-1}, W_{t-2} \ldots, W_{t-7}$ are its 7 lags. We use the 7 order because in Figure 7, ACF, auto-correlation function, tails off while PACF, partial auto-correlation function, cuts off after approximately the 7th lag although an exception is detected. Specifically, ACF measures the relationship between W_t and $W_{t-7}$and PACF measures the relationship between W_t and W_{t-7} after removing the effects of lags $1, 2, \ldots, 6$. Other numbers of order, up to 8, are also tried for the purpose of the best fit, but it turns out that the 7th order has the lowest AIC. The coefficient results show that lags 1, 4, and 7 are statistically significant. The predictions in Figure 8 are very close to the observed values. The residuals in Figure 9 are stationary with some variations yet acceptable. The diagnostic plot shows that the assumption of constant variance of residuals holds while the assumption of normality is violated. However, we argue that our results are valid as we have sufficiently large samples.

Another important aspect, that remarks the lag choices for the model, is the Ljung-Box test; by definition the Ljung-Box test shows if a time series is independently distributed or in other words if the autocorrelation for some specified lags is null. For the case we are taking under analysis, the aim of the test is to remark that including terms with lag greater than 8 for the error term W_t in the final model, is not improving the quality of the fitting. Here is below the exact notation for the test with the null hypothesis H_0 and the alternative one H_1:

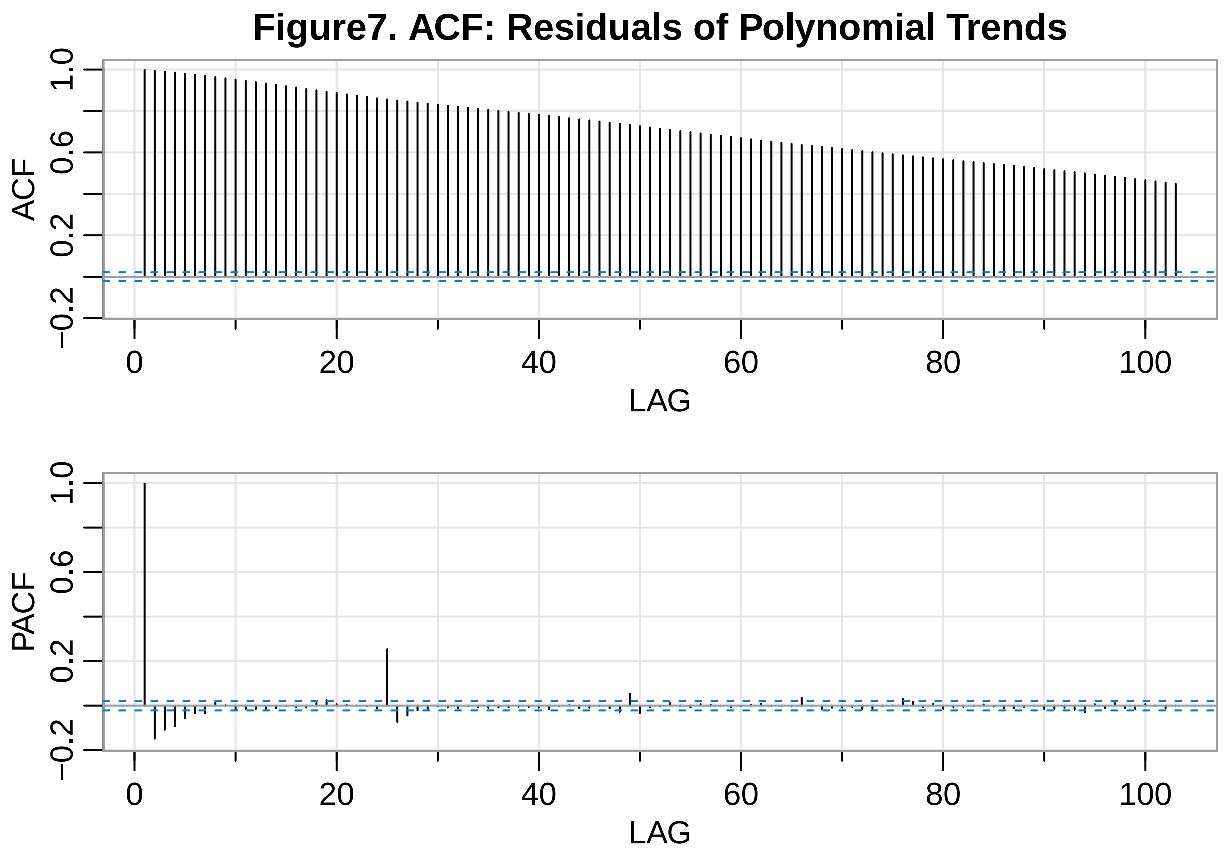
H_0 : $\rho(8) = \rho(9) = ... = \rho(20) = 0$ where $\rho(h) = Corr(W_t, W_{t-1})$
H_1 : there exists at least one $\rho(h)$ where $8 \leq h \leq 20$ such that $\rho(h) \neq 0$

The test is carried under a significance level of 95% and as it is shown by the plot of the p-values; the null hypothesis H_0 can not be rejected for none of the lags between 8 and 20. Therefore, the model is complete and the initial assumptions about the autocorrelation can be confirmed.

An important aspect to remark about the bike sharing time series is their stationarity. According to the definition, a time series is stationary if its mean and variance are constant over time t and if the autocorrelation of the series is constant over time t given the lag (but it can be different between different lags and same time t). After an initial look at the graph of the bike sharing series, the mean seems not to be constant during all the observation period but it becomes more stable after the second half, in addition, the variance has a similar behavior while the autocorrelation seems to respect the stationarity conditions. Potentially if the observation period was larger and forward in time, the series could assume a more stationary behavior which would lead to an easier analysis. The reasons why it would become stationary is also related to the bike sharing business itself, as once it will fulfill the total demand of the bikes in Seoul, then the daily demand of bikes will have a constant mean and variance.

The fact that the series is not stationary can lead to building an ARIMA model, which is basically an ARMA model of the d^{th} order difference of the series defined as $\bigtriangledown X_t = X_t - X_{t-d}$; but after analyzing the effect of all the other explanatory variables over the series, it appears that the non stationarity is not directly due to the nature of the series itself but instead it is due to the explanatory variables' effect, in fact the graph of the difference between the series and the explanatory variables shows to have constant mean and variance over time, which added to the autocorrelation properties, clearly evidence stationarity. Therefore, after including the linear regression of the explanatory variables, the bike sharing series can be easily modeled as an ARMA model of a certain order and the model would also have a great forecasting power.

$$W_t = c + \phi_1 W_{t-1} + \phi_2 W_{t-2} + \cdots + \phi_7 W_{t-7} + \varepsilon_t$$



```
## $coefs
##      X          beta     se      p.value
## [1,] "constant" "-0.005" "0.075" "0.949"
## [2,] "lag1"     "1.112"  "0.011" "0"
## [3,] "lag2"     "-0.029" "0.016" "0.074"
## [4,] "lag3"     "-0.007" "0.016" "0.673"
## [5,] "lag4"     "-0.032" "0.016" "0.05"
## [6,] "lag5"     "-0.018" "0.016" "0.258"
## [7,] "lag6"     "0.005"  "0.016" "0.761"
## [8,] "lag7"     "-0.036" "0.011" "0.001"
##
## $sigma
## [1] 48.40281
```

Figure8. Predictions by Polynomial Trends & Auto−Corr Errors

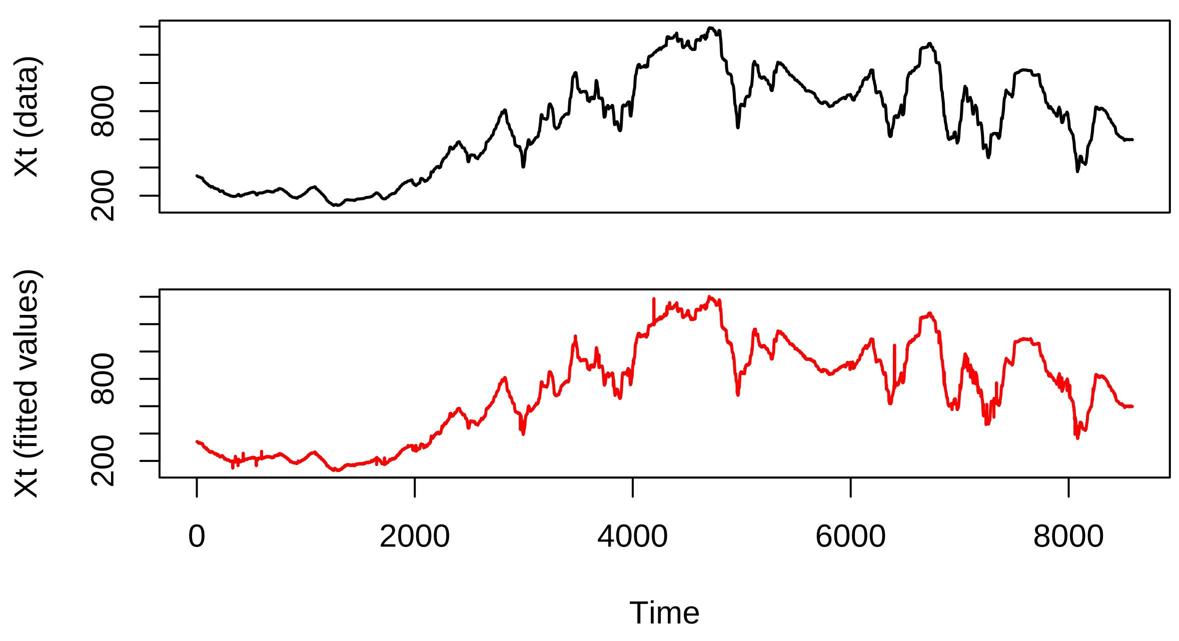


Figure9. Residuals of Polynomial Trends & Auto−Corr Errors

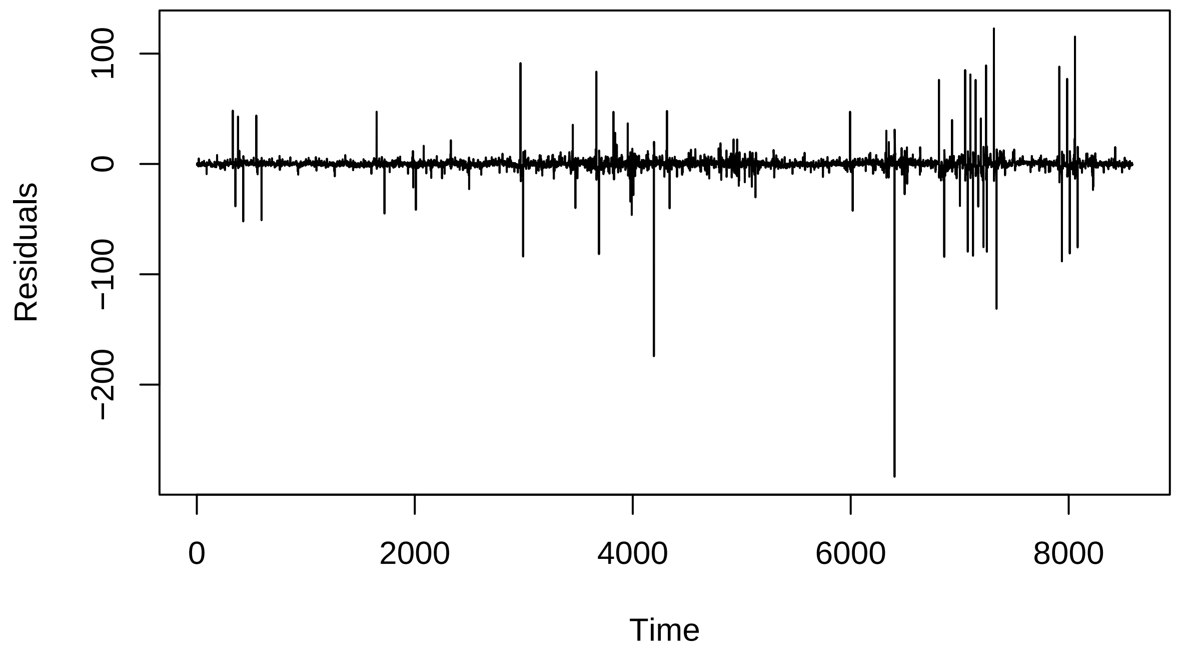


Figure10. Diagnostics for Polynomial Trends & Auto−Corr Errors

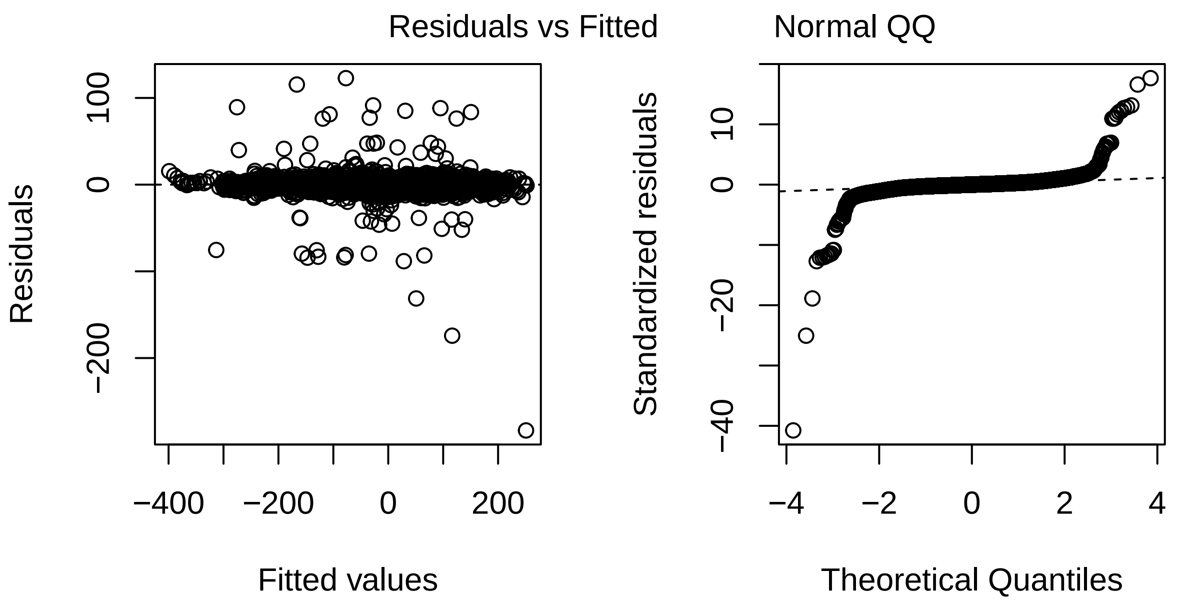
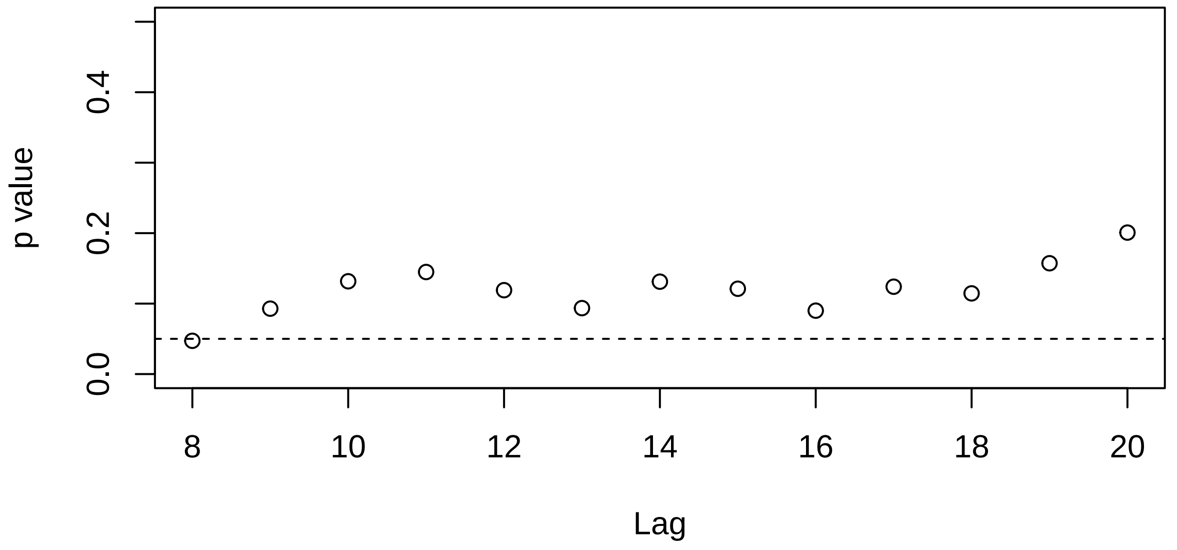


Figure11. P−values for Ljung−Box Statistic



Conclusion

Simply by observing the graphs we have a clear winner between the two approaches: the data is clearly autocorrelated. That means that the number of bikes that are rented each hour is affected by the number of bikes that were rented one hour before, two hours before, one day before... etc. This has several explanations, all of them based on common sense:

- It's very rare to have sudden weather changes. Temperature, rain, humidity follow a temporal series (they are not random). That translates into having similar weather conditions extended over several successive data points. This is something that a "regular" linear regression model would ignore, but a linear regression that takes into account the trends and the autocorrelation would benefit from it.
- Human behavior is not random and it's very predictable. People easily create habits and that is something that is easy to predict, especially with the last types of models that we have used in this project.
- Bikes very often are rented for more than one hour. Therefore, the number of rented bikes in each data point isn't random either, but is just the number of rented bikes from the previous hour plus an additive term (which might be positive if the number of returned bikes is less than the number of newly rented bikes, and negative if the opposite is true.).

Given this, the best model we found to optimize the prediction of the number of rented bikes is a linear regression with polynomial trends and auto-correlated errors. Once this model is trained with data up until the day before you want to predict plus the weather forecast for the next day, it can be used to predict the hourly number of bikes that will be rented the next day.

This isn't a fixed model though. To keep obtaining optimal estimations, the real observed data for each day must be added to the training set on a daily basis, and the estimator parameters must be recomputed frequently. It should also be considered not to include very old data, for example, data from 4 years ago. Even though we have proven the existence of some seasonality and trend, some major factors could come into play and change the "real" distribution of the data (for example, a global pandemic).

Prediction of time series data is a continuous process, but luckily the computation of these models is not very expensive and can be carried out frequently without spending too much money and time. Our final recommendation to the company would be to hire a data scientist to take care of all the modeling and data collection and cleaning.

Reference and Acknowledgement

Data source: https://archive.ics.uci.edu/ml/datasets/Seoul+Bike+Sharing+Demand

Appendix

```r
knitr::opts_chunk$set(warning = FALSE, message = FALSE)

library(zoo)
library(xts)
library(astsa)
library(fastDummies)
library(ggplot2)
```

```r
# generate simulated data
set.seed(2022)

x_ = as.matrix(seq(0,100,0.1))
y_ = 3*x_ + rnorm(length(x_),mean = 0, sd = 15)

x_1 = as.matrix(rep(1,length(x_)))
x_ = cbind(x_1, x_)
y_ = as.matrix(y_)

# lm coefficient function
ols.coef <- function(y, X) {

  #get number of observations and parameters
  n <- nrow(X)
  p <- ncol(X)

  #compute coefficients
  beta <- chol2inv(chol(crossprod(X)))%*%crossprod(X,y)
  colnames(beta) <- "beta"

  #compute sigma
  e = y - X%*%beta
  sigma = sum(e*e)/(n-p)

  #compute se
  V <- chol2inv(chol(crossprod(X))) * sigma
  se <- sqrt(diag(V))

  #compute p-value
  pvalue <- round(2*pt(abs(beta/se), n,lower.tail = FALSE),5)
  colnames(pvalue) <- "p.value"

  #return result
  result <- list(coefs=cbind(X=colnames(X), round(beta,3), se=round(se,3), round(pvalue,3)),
                 sigma=sigma)
  return(result)
}

# simulation test
coef = ols.coef(y_,x_)
coef


ggplot()+
  aes(x=x_[,2],y=y_)+
  geom_jitter()+
  ylab("simulated y") +
  xlab("simulated x") +
  geom_abline(slope=coef$coefs[2,1], intercept = coef$coefs[1,1], color = "red", size=1)
```

```r
# lm residual function: Computes the residual for a given linear regression model
ols.resid <- function(y, X) {

  #compute coefficients
  beta <- chol2inv(chol(crossprod(X)))%*%crossprod(X,y)

  #compute residuals
  e = y - X%*%beta

  #return residuals
  return(e)
}

# lm prediction function: Given y and X, returns the fitted values for the y using the OLS
ols.pred <- function(y, X) {

  #compute coefficients
  beta <- chol2inv(chol(crossprod(X)))%*%crossprod(X,y)

  #compute predicted values
  fit = as.matrix(X %*% beta)

  #return predicted values
  return(fit)
}


setwd("/Users/mingzhao/Desktop")

data <- read.csv("Dataset_withonlynew_var.csv")


# dataset for linear regression
data1 <- subset(data, select=-c(X, Date, Winter, Monday, X0))

data1$avg7days <- stats::filter(data1$Rented.Bike.Count, rep(1/168, 168), sides = 1)

data1$Temperature..C. <- stats::filter(data1$Temperature..C., rep(1/168, 168), sides = 1)
data1$Humidity... <- stats::filter(data1$Humidity..., rep(1/168, 168), sides = 1)
data1$Wind.speed..m.s. <- stats::filter(data1$Wind.speed..m.s., rep(1/168, 168), sides = 1)
data1$Visibility..10m. <- stats::filter(data1$Visibility..10m., rep(1/168, 168), sides = 1)
data1$Dew.point.temperature..C. <- stats::filter(data1$Dew.point.temperature..C., rep(1/168, 168), side
data1$Solar.Radiation..MJ.m2. <- stats::filter(data1$Solar.Radiation..MJ.m2., rep(1/168, 168), sides =
data1$Rainfall.mm. <- stats::filter(data1$Rainfall.mm., rep(1/168, 168), sides = 1)
data1$Snowfall..cm. <- stats::filter(data1$Snowfall..cm., rep(1/168, 168), sides = 1)

data1 <- data1[complete.cases(data1$avg7days),]
data1 <- data1[-1,]
data1 <- data1[,-1]

# dataset for linear regression with trends
data3 <- subset(data, select=-c(X, Date, Winter, Monday, X0))
```

```
data3$avg7days <- stats::filter(data3$Rented.Bike.Count, rep(1/168, 168), sides = 1)

data3$Temperature..C. <- stats::filter(data3$Temperature..C., rep(1/168, 168), sides = 1)
data3$Humidity... <- stats::filter(data3$Humidity..., rep(1/168, 168), sides = 1)
data3$Wind.speed..m.s. <- stats::filter(data3$Wind.speed..m.s., rep(1/168, 168), sides = 1)
data3$Visibility..10m. <- stats::filter(data3$Visibility..10m., rep(1/168, 168), sides = 1)
data3$Dew.point.temperature..C. <- stats::filter(data3$Dew.point.temperature..C., rep(1/168, 168), sides
data3$Solar.Radiation..MJ.m2. <- stats::filter(data3$Solar.Radiation..MJ.m2., rep(1/168, 168), sides =
data3$Rainfall.mm. <- stats::filter(data3$Rainfall.mm., rep(1/168, 168), sides = 1)
data3$Snowfall..cm. <- stats::filter(data3$Snowfall..cm., rep(1/168, 168), sides = 1)

Xt <- ts(data3$avg7days, start = c(1, 0), freq = 24)

data3$trend1 = time(Xt)
data3$trend2 = data3$trend1^2
data3$trend3 = data3$trend1^3
data3$trend4 = data3$trend1^4

data3 <- data3[complete.cases(data3$avg7days),]

data3 <- data3[-1,]
data3 <- data3[,-1]


Xt <- ts(data1$avg7days, start = c(1, 0), freq = 24)
Yt <- ts(data$Rented.Bike.Count, start = c(1, 0), freq = 24)

par(mfrow=c(2,1))
par(mar=c(1, 4.1, 4, 2.1))
plot.ts(Yt, col='black', xlab=NULL, ylab='Original data', lwd=c(1.5,1.5), xaxt="n",
        main="Figure0a. Time Series: Original vs. 7-day Moving Average Data")
par(mar=c(4.1, 4.1, 1, 2.1))
plot.ts(Xt, col='black', ylab='7-day moving average', lwd=c(1.5,1.5))


par(mfrow=c(1,1))
hist(Xt, col = "white", main="Figure0b. Distribution of 7-day Moving Average Data")



y <- as.matrix(data1[, 43])                         # dependent variable
X <- cbind(constant = 1, as.matrix(data1[, -43]))  # design matrix

coef1 <- ols.coef(y, X)
fit1 <- ols.pred(y, X)
e1 <- ols.resid(y, X)


coef1

step0 <- glm(avg7days~1, family = gaussian, data = data1)
step1 <- glm(avg7days~., family = gaussian, data = data1)
```

```r
ols_fit <- step(step0,
                scope = list(lower=step0, upper=step1),
                direction = "both",
                k = 2,
                trace = 0)


par(mfrow=c(1,1))
plot.ts(cbind(y, fit1),plot.type = 'single',
        col=c('black','red'),
        ylab='Xt',
        main="Figure1. Predictions by Linear Regression",
        lwd=c(1.5,1.5))
legend('topleft',legend = c('data','fitted values'),
       col=c('black','red'),
       lty=c(1,1),
       lwd=c(1,1))

plot.ts(e1, ylab = "Residuals", main = "Figure2. Residuals of Linear Regression")

par(mfrow=c(1,2), oma=c(0,0,2,0), mar=c(5.1, 4.1, 3, 2.1))
plot(fit1, e1, xlab = "Fitted values", ylab = "Residuals")
mtext(side=3, line=0.5, at=-0.07, adj=0, cex=1, "Residuals vs Fitted")
abline(h=0, lty=2)
qqnorm(scale(e1), main = NULL, ylab = "Standardized residuals")
qqline(scale(e1), lty=2)
mtext(side=3, line=0.5, at=-0.07, adj=1.2, cex=1, "Normal QQ")
mtext("Figure3. Diagnostics for Linear Regression", cex=1.2, side=3, font=2, line=-0.5, outer=TRUE)


y <- as.matrix(data3[, 43])                        # dependent variable
X <- cbind(constant = 1, as.matrix(data3[, -43]))  # design matrix

coef3 <- ols.coef(y, X)
fit3 <- ols.pred(y, X)
e3 <- ols.resid(y, X)

coef3

par(mfrow=c(1,1))
plot.ts(cbind(y, fit3),plot.type = 'single',
        col=c('black','red'),
        ylab='Xt',
        main="Figure4. Predictions by Polynomial Trends",
        lwd=c(1.5,1.5))
legend('topleft',legend = c('data','fitted values'),
       col=c('black','red'),
       lty=c(1,1),
       lwd=c(1,1))

plot.ts(e3, ylab = "Residuals", main = "Figure5. Residuals of Polynomial Trends")

par(mfrow=c(1,2), oma=c(0,0,2,0), mar=c(5.1, 4.1, 3, 2.1))
```

```r
plot(fit3, e3, xlab = "Fitted values", ylab = "Residuals")
mtext(side=3, line=0.5, at=-0.07, adj=0, cex=1, "Residuals vs Fitted")
abline(h=0, lty=2)
qqnorm(scale(e3), main = NULL, ylab = "Standardized residuals")
qqline(scale(e3), lty=2)
mtext(side=3, line=0.5, at=-0.07, adj=1.2, cex=1, "Normal QQ")
mtext("Figure6. Diagnostics for Polynomial Trends", cex=1.2, side=3, font=2, line=-0.5, outer=TRUE)


acf2(e3, main = "Figure7. ACF: Residuals of Polynomial Trends")

lag1 <- c(rep(NA, 1), head(e3, -1))
lag2 <- c(rep(NA, 2), head(e3, -2))
lag3 <- c(rep(NA, 3), head(e3, -3))
lag4 <- c(rep(NA, 4), head(e3, -4))
lag5 <- c(rep(NA, 5), head(e3, -5))
lag6 <- c(rep(NA, 6), head(e3, -6))
lag7 <- c(rep(NA, 7), head(e3, -7))

lags <- cbind(lag1, lag2, lag3, lag4, lag5, lag6, lag7)[-c(1:7),]


y <- e3[-c(1:7)]                # dependent variable
X <- cbind(constant = 1, lags) # design matrix

coef4 <- ols.coef(y, X)
fit4 <- ols.pred(y, X)
e4 <- ols.resid(y, X)

coef4

fit = fit3[-c(1:7)] + fit4

Y <- as.matrix(data3[, 43])[-c(1:7)]

par(mfrow=c(2,1))
par(mar=c(1, 4.1, 4, 2.1))
plot.ts(Y, col='black', xlab=NULL, ylab='Xt (data)', lwd=c(1.5,1.5), xaxt="n",
        main="Figure8. Predictions by Polynomial Trends & Auto-Corr Errors")
par(mar=c(4.1, 4.1, 1, 2.1))
plot.ts(fit, col='red', ylab='Xt (fitted values)', lwd=c(1.5,1.5))

par(mfrow=c(1,1))
par(mar=c(5.1, 4.1, 2.3, 2.1))
plot.ts(e4, ylab = "Residuals", main = "Figure9. Residuals of Polynomial Trends & Auto-Corr Errors")

par(mfrow=c(1,2), oma=c(0,0,2,0), mar=c(5.1, 4.1, 3, 2.1))
plot(fit4, e4, xlab = "Fitted values", ylab = "Residuals")
mtext(side=3, line=0.5, at=-0.07, adj=0, cex=1, "Residuals vs Fitted")
abline(h=0, lty=2)
qqnorm(scale(e4), main = NULL, ylab = "Standardized residuals")
qqline(scale(e4), lty=2)
mtext(side=3, line=0.5, at=-0.07, adj=1.2, cex=1, "Normal QQ")
mtext("Figure10. Diagnostics for Polynomial Trends & Auto-Corr Errors", cex=1.2, side=3, font=2,
      line=-0.5, outer=TRUE)

p8  <- Box.test(e4, lag = 8, type = "Ljung-Box", fitdf = 7)[3]
p9  <- Box.test(e4, lag = 9, type = "Ljung-Box", fitdf = 7)[3]
p10 <- Box.test(e4, lag = 10, type = "Ljung-Box", fitdf = 7)[3]
p11 <- Box.test(e4, lag = 11, type = "Ljung-Box", fitdf = 7)[3]
p12 <- Box.test(e4, lag = 12, type = "Ljung-Box", fitdf = 7)[3]
p13 <- Box.test(e4, lag = 13, type = "Ljung-Box", fitdf = 7)[3]
p14 <- Box.test(e4, lag = 14, type = "Ljung-Box", fitdf = 7)[3]
p15 <- Box.test(e4, lag = 15, type = "Ljung-Box", fitdf = 7)[3]
p16 <- Box.test(e4, lag = 16, type = "Ljung-Box", fitdf = 7)[3]
p17 <- Box.test(e4, lag = 17, type = "Ljung-Box", fitdf = 7)[3]
p18 <- Box.test(e4, lag = 18, type = "Ljung-Box", fitdf = 7)[3]
p19 <- Box.test(e4, lag = 19, type = "Ljung-Box", fitdf = 7)[3]
p20 <- Box.test(e4, lag = 20, type = "Ljung-Box", fitdf = 7)[3]

p.value <- cbind(p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p19, p20)
lag <- c(8:20)

par(mfrow=c(1,1))
plot(lag, p.value, ylim = c(0, 0.5), ylab = "p value", xlab = "Lag",
     main = "Figure11. P-values for Ljung-Box Statistic")
abline(h=0.05, lty=2)
```