# STATS 769 - Lab 04 - bole001

Bernard O'Leary

26 August 2019

## Describe the methods used to import the data to R

Data have been imported into R from XML and JSON sources, using MongoDB for JSON, xml2 for XPath and jsonlite for JSON. Unfortunately I have not got to using BaseX for importing the XML data. However the following three sections provide code that illustrates how the importation procedure worked for each type of data.

For both jsonlite and xml2, data were imported using a file-based approach, looping through files and then combining into a dataframe. For MongoDB, data have been collected by running a single query of the database, which results in a dataframe.

All data sources and methods used resulted in the same dataframe being rendered in R, with two columns and 98817 rows.

### JSON data

Import data and provide dimensions and first 6 rows of the data for validation purposes.

```
library(jsonlite)

## Warning: package 'jsonlite' was built under R version 3.6.1

readJsonFileIntoDataframe <- function(file_name) {
  # Read in data
  fromJSON(readLines(file_name))
}


file_numbers <- c(1:10)
file_names <- paste0("trips-", file_numbers, ".json")
# At home
files <- file.path("./JSON", file_names)
# At uni
#files <- file.path("/course/Labs/Lab04/JSON", filenames)
json_data <- do.call(rbind, lapply(files, readJsonFileIntoDataframe))
json_data <- subset(json_data, year == "2018" & vehicle_type == "scooter",
select = c("trip_duration", "trip_distance"))
head(json_data)

##   trip_duration trip_distance
## 1           358           915
```

```
## 2           226          839
## 3           324         1206
## 4          1096            0
## 5           408         1144
## 6          1094         2631
```

```r
dim(json_data)
```

```
## [1] 98817     2
```

## MongoDB data

Import data and provide dimensions and first 6 rows of the data for validation purposes.

```r
library(mongolite)
```

```
## Warning: package 'mongolite' was built under R version 3.6.1
```

```r
# At home
mongo_db <- mongo("trips", url = "mongodb://localhost:27017/local")
# At uni
#mongo_db <- mongo("trips")
mongo_db_data <- mongo_db$find(
  query = '{"vehicle_type": "scooter", "year": "2018"}',
  fields = '{"trip_duration": true, "trip_distance": true, "_id": false}'
)
head(mongo_db_data)
```

```
##   trip_duration trip_distance
## 1           358          915
## 2           226          839
## 3           324         1206
## 4          1096            0
## 5           408         1144
## 6          1094         2631
```

```r
dim(mongo_db_data)
```

```
## [1] 98817     2
```

## XML data

Import data and provide dimensions and first 6 rows of the data for validation purposes.

```r
library(xml2)
```

```
## Warning: package 'xml2' was built under R version 3.6.1
```

```r
readXmlFileIntoDataframe <- function(file_name) {
  # Read in data
  xml_data <- read_xml(file_name)
  trips <- xml_find_all(xml_data, "//row[vehicle_type = 'scooter'][year =
```

```
2018]")
  trip_distance <- as.numeric(xml_text(xml_find_first(trips,
"trip_distance")))
  trip_duration <- as.numeric(xml_text(xml_find_first(trips,
"trip_duration")))
  as.data.frame(cbind(trip_duration, trip_distance))
}


file_numbers <- c(1:10)
file_names <- paste0("trips-", file_numbers, ".xml")
# At home
files <- file.path("./XML", file_names)
# At uni
#files <- file.path("/course/Labs/Lab04/JSON", filenames)
xml_data <- do.call(rbind, lapply(files, readXmlFileIntoDataframe))
head(xml_data)

##   trip_duration trip_distance
## 1           358           915
## 2           226           839
## 3           324          1206
## 4          1096             0
## 5           408          1144
## 6          1094          2631

dim(xml_data)

## [1] 98817     2
```

## Model the data and derive estimates of test error for 5 polynomial models

Apply 10-fold cross-validation across 5 models with increasing polynomial terms up to the a fifth order polynomial. We find that the error begins to increase after the fourth order polynomial is added (i.e. with a fifth order polynomial term, the error increases).

```
## Cleanse our data
model_trips <- subset(xml_data,
  trip_duration > 0 & trip_distance > 0)
trip_duration <- log(model_trips$trip_duration)
trip_distance <- log(model_trips$trip_distance)
head(model_trips)

##   trip_duration trip_distance
## 1           358           915
## 2           226           839
## 3           324          1206
## 5           408          1144
```

```
## 6          1094         2631
## 7           705         1248
```

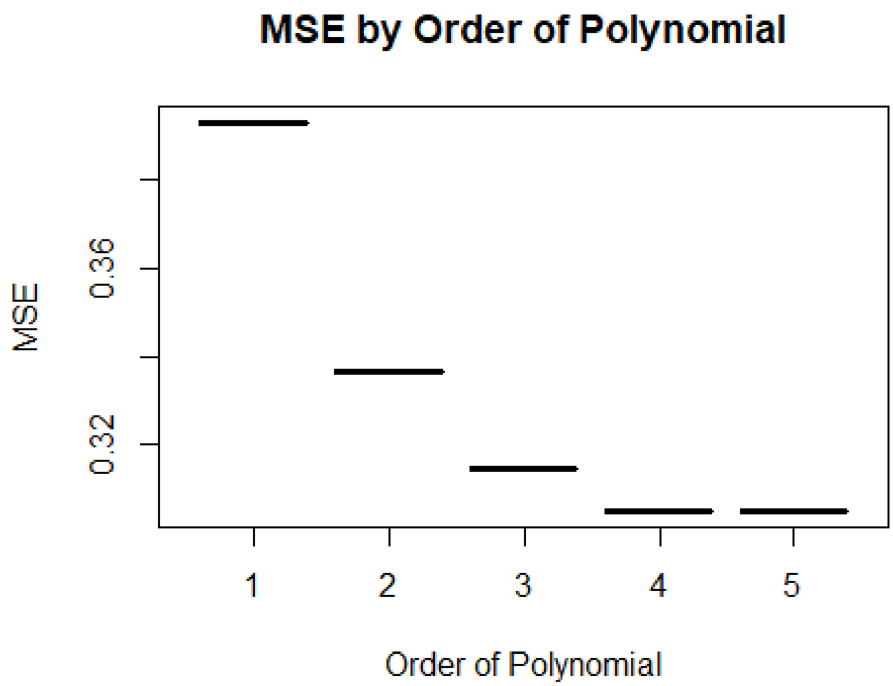
```
## Define 10 splits
labels <- rep(1:10, length.out=nrow(model_trips))
groups <- sample(labels)

## Define our MSE function
mse <- function(i, formula) {
    test_set <- groups == i
    train_set <- groups != i
    fit <- lm(formula,
              data.frame(x=trip_distance[train_set],
y=trip_duration[train_set]))
    pred <- predict(fit, data.frame(x=trip_distance[test_set]))
    mean((pred - trip_duration[test_set])^2)
}

## Train and test five models with increasing polynomial terms, look for
lowest MSE value
mse_polynomial_model <- data.frame("Order" = "1", "MSE" = mean(sapply(1:10,
mse, y ~ x)))
mse_polynomial_model <- rbind(mse_polynomial_model, data.frame("Order" = "2",
"MSE" = mean(sapply(1:10, mse, y ~ x + I(x^2)))))
mse_polynomial_model <- rbind(mse_polynomial_model, data.frame("Order" = "3",
"MSE" = mean(sapply(1:10, mse, y ~ x + I(x^2) + I(x^3)))))
mse_polynomial_model <- rbind(mse_polynomial_model, data.frame("Order" = "4",
"MSE" = mean(sapply(1:10, mse, y ~ x + I(x^2) + I(x^3) + I(x^4)))))
mse_polynomial_model <- rbind(mse_polynomial_model, data.frame("Order" = "5",
"MSE" = mean(sapply(1:10, mse, y ~ x + I(x^2) + I(x^3) + I(x^4) + I(x^5)))))
mse_polynomial_model
```

```
##   Order       MSE
## 1     1 0.3927677
## 2     2 0.3367657
## 3     3 0.3145111
## 4     4 0.3051668
## 5     5 0.3052095
```

## Conclusion summarising analysis

We can see from the following chart that the MSE flattens out at the fourth order polynomial. Although it is not obvious from looking at the chart, there is a slight increase in MSE at the fifth order polynomial.

```
plot(mse_polynomial_model$Order, mse_polynomial_model$MSE, type="o",
xlab="Order of Polynomial", ylab="MSE", main="MSE by Order of Polynomial")
```


MSE by Order of Polynomial
MSE
0.32
0.36
1
2
3
4
5
Order of Polynomial