Lab_3_K-means_and_random_forests

Daniel Martin-Williams

2023-12-29

Q1

Consider a random variable y with expected value $\mathbb{E}(y) = \mu$ and variance $V(y) = \sigma^2$. Assume we have a collection of observations $\{y_i\}_{i=1}^n$ that constitute an independent and identically distributed (iid) sample of size n from y.

1. Derive the variance of the sample mean $\overline{y_n}$, expressed as $\overline{y_n} = \frac{1}{n}\sum_{i=1}^n y_i$. Discuss the relationship between $V(y)$ and $V(\overline{y_n})$.

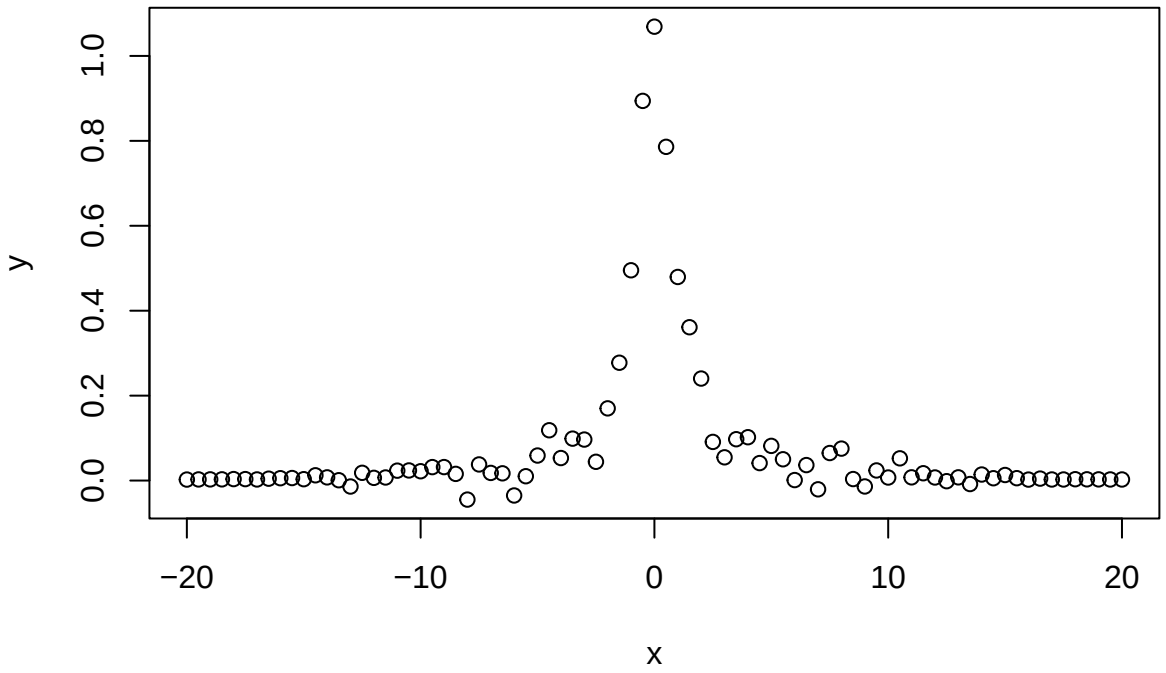
Solution: Given that the observations are iid, the variance of the sample mean is computed as follows:

$$V(\overline{y_n}) = V\left(\frac{1}{n}\sum_{i=1}^n y_i\right) = \frac{1}{n^2}V\left(\sum_{i=1}^n y_i\right) = \frac{1}{n^2}\sum_{i=1}^n V(y_i) = \frac{n\sigma^2}{n^2} = \frac{\sigma^2}{n}.$$

Thus, the variance of $\overline{y_n}$ is a fraction $\frac{1}{n}$ of the variance of the individual observations $V(y)$. This implies that taking the average of iid observations leads to a reduction in variance, which is a fundamental principle behind the efficacy of ensemble methods like bagging and random forests over single-predictor models.

Q2

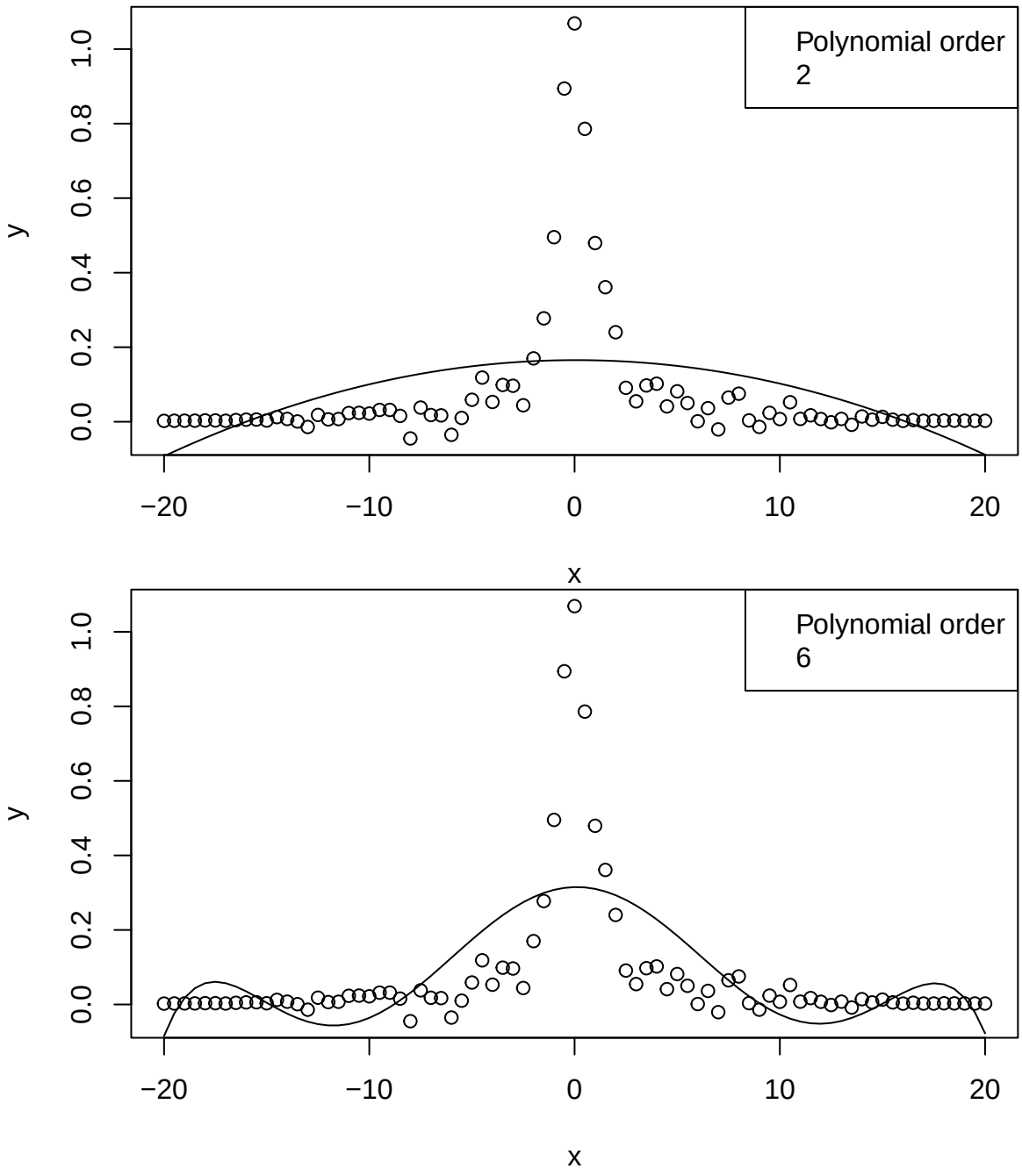
```
df <- read.csv("rfexample.csv")
plot(df$x, df$y, xlab = 'x', ylab = 'y')
```

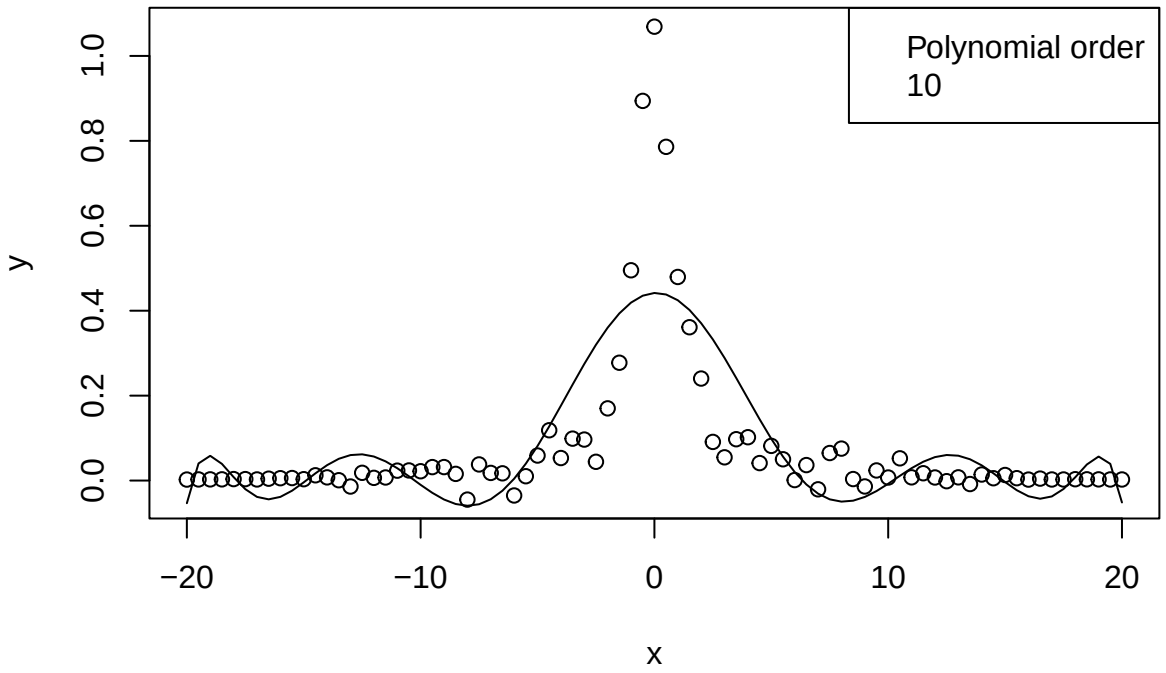


This shows a non-linear relationship.

Q3

```
polys = c(2,6,10)
for (i in polys){
  polyreg = glm(y ~ poly(x, i, raw = T), data = df)
  plot(df$x, df$y, xlab = 'x', ylab = 'y')
  lines(df$x, polyreg$fitted.values)
  legend (x = 'topright', legend = c('Polynomial order', i))
  }
```

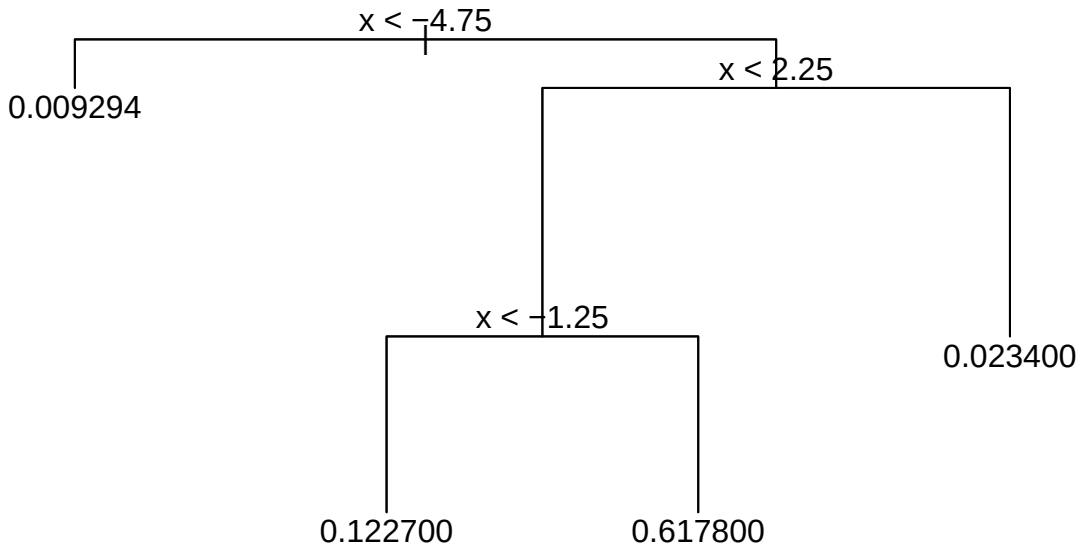

Polynomial order 2
y
0.0
0.2
0.4
0.6
0.8
1.0
−20
−10
0
10
20
x
Polynomial order 6
y
0.0
0.2
0.4
0.6
0.8
1.0
−20
−10
0
10
20
x



Polynomials do not seem to provide an accurate fit to this data, missing out on the more extreme values around 0.

Q4

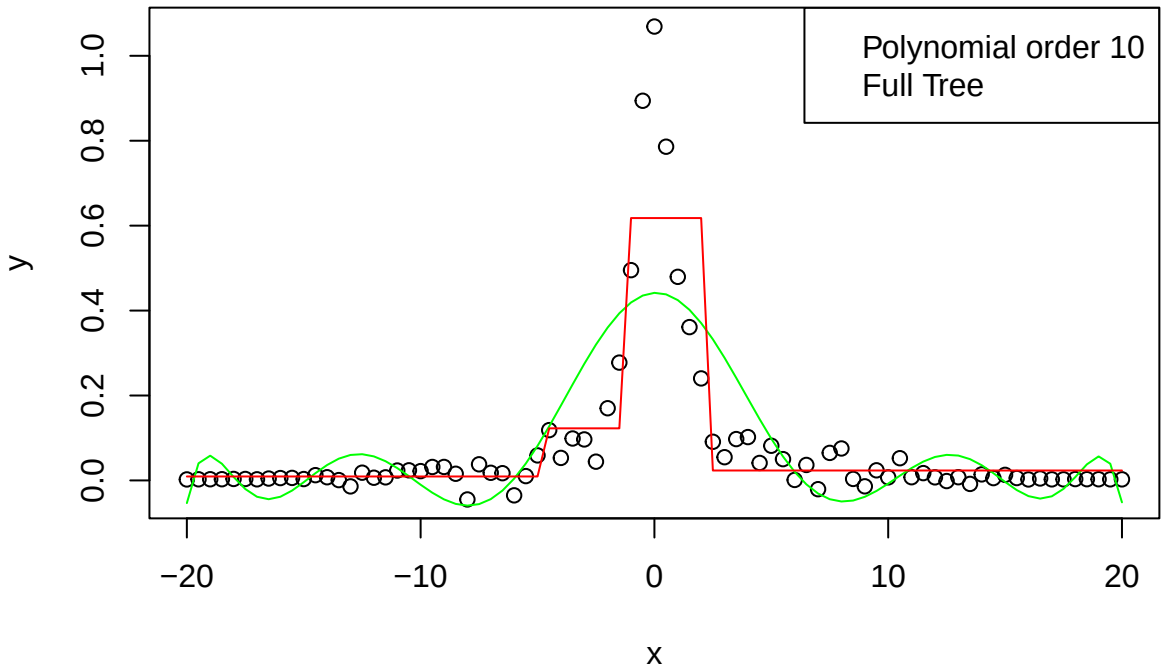
```
library(tree)
decisiontree <- tree(y~x, data = df, mindev = 0.01)
plot(decisiontree)
text(decisiontree)
```



This is the decision tree associated with the tree created by the tree function. The numbers at each leaf represent the average Y values at each leaf, or final node. The inequalities at each decision node, or split, is the rule in splitting the dataset at each split.

Q5

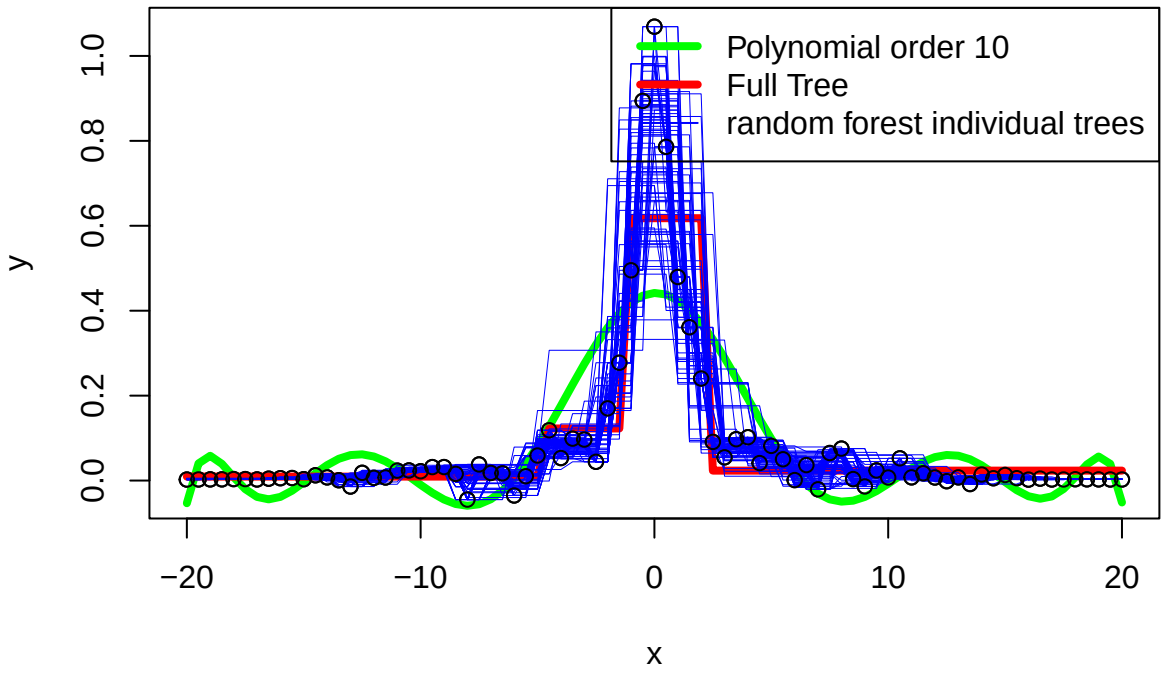
```
polyreg = glm(y ~ poly(x, 10, raw = T), data = df)
decisiontree2 <- tree(y~x, data = df, mindev = 0)

predvals = predict(decisiontree, newdata = df)
plot(df$x, df$y, xlab = 'x', ylab = 'y')
lines(df$x, polyreg$fitted.values, col = 'green')
lines(df$x, predvals, col = 'red')
legend (x = 'topright', legend = c('Polynomial order 10','Full Tree' ), col = c('green','red'))
```



Here, we see that the decision tree has more accurately represented the data compared to polynomials, but still not optimally, missing out the largest values of y. It should be noted that in individual, fully grown tree like this is likely very over fitted, and would not give useful predictive OOS values.

Q6

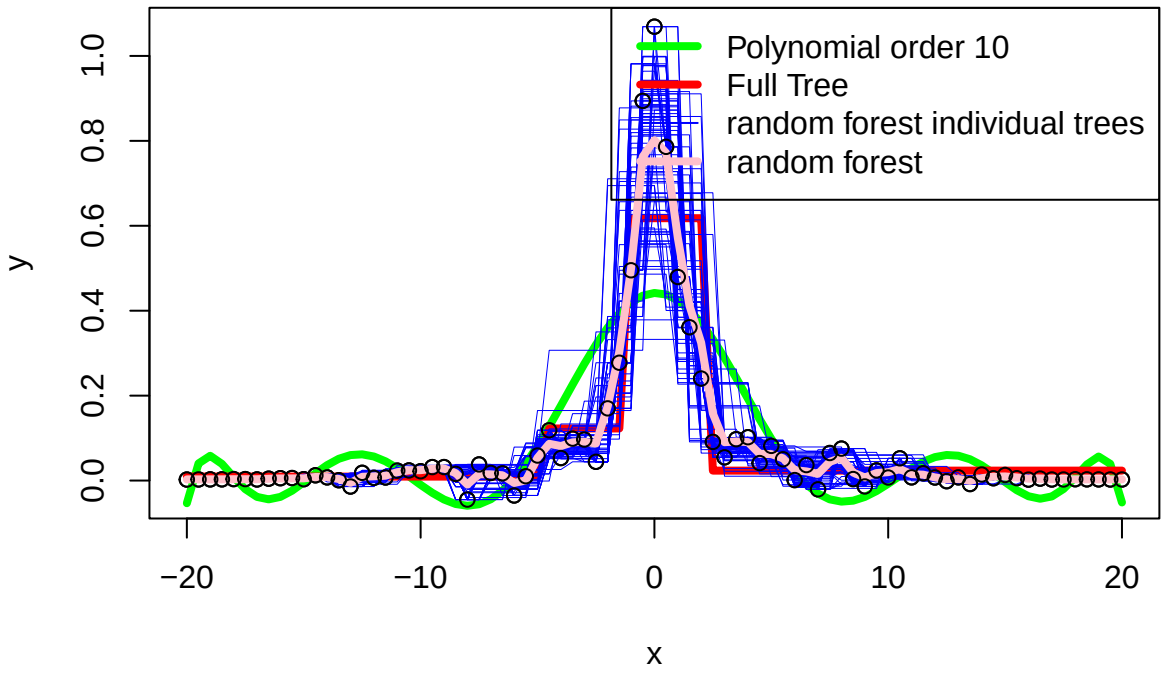
```
library(ranger)
randomforest <- ranger(y ~ x, data = df, max.depth = 0, num.trees = 100)
allrfpred <- predict(randomforest, data = df, predict.all = T)
plot(df$x, df$y, xlab = 'x', ylab = 'y')
lines(df$x, polyreg$fitted.values, col = 'green', lwd = 4)
lines(df$x, predvals, col = 'red', lwd = 4)
matlines(replicate(100,df$x),allrfpred$predictions,col = "blue",lwd = 0.5,lty=1)
points(df$x,df$y)
legend (x = 'topright', legend = c('Polynomial order 10','Full Tree',
                                   'random forest individual trees' ), col = c
        ('green','red','blue'), lty = c(1,1,1), lwd = c(4,4,1))
```



Observing the performance of each tree within the random forest, we can see that they do a decent job of capturing the nuances of the data, including the sharp peak at $x = 0$ and the more level sections at higher absolute values of x. Yet, these trees exhibit a considerable amount of variability. They are more noisy. For instance, while some trees drastically underestimate the peak—sometimes even more so than our polynomial model—others almost hit the mark on the highest y-value present in the dataset. This inconsistency can be attributed to the tendency of individual trees to overfit their specific random samples. Consequently, depending on whether the sample includes the maximum y-value or not, the trees will vary in their predictions. This is the precise reason why relying on a single tree often leads to disappointing predictions with new, unseen data; they simply don't generalize well.

Q7

```
rfpred <- predict(randomforest, data = df, predict.all = F)
plot(df$x, df$y, xlab = 'x', ylab = 'y')
lines(df$x, polyreg$fitted.values, col = 'green', lwd = 4)
lines(df$x, predvals, col = 'red', lwd = 4)
matlines(replicate(100,df$x),allrfpred$predictions,col = "blue",lwd = 0.5,lty=1)
lines(df$x, rfpred$predictions, col = 'pink', lwd = 4)

points(df$x,df$y)
legend (x = 'topright', legend = c('Polynomial order 10','Full Tree',
                                   'random forest individual trees' , 'random forest'), col = c
        ('green','red','blue', 'pink'), lty = c(1,1,1), lwd = c(4,4,1))
```



The random forest gives a smoother, lower variance set of predictions to individual trees. It also provides a better fit than the polynomial, capturing the extreme values around 0 to an extent, as well as the smoother y values at the high absolute values of x. Essentially, the tendency of individual trees to overfit is mitigated when their predictions are aggregated through the bagging process used in creating the random forest. This collective approach helps in smoothing out the overfitting issue. Random forest provides the closest predictive fit to the data.

Q8 - bind the data into 1 df

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
uber.apr <- read.csv("uber-raw-data-apr14.csv")
uber.may <- read.csv("uber-raw-data-may14.csv")
uber.jun <- read.csv("uber-raw-data-jun14.csv")
uber.jul <- read.csv("uber-raw-data-jul14.csv")
uber.aug <- read.csv("uber-raw-data-aug14.csv")
uber.sep <- read.csv("uber-raw-data-sep14.csv")
uber <- rbind(uber.apr,uber.may,uber.jun,uber.jul,uber.aug,uber.sep)

uber$Date.Time <- mdy_hms(uber$Date.Time)
uber$Year <- factor(year(uber$Date.Time))
uber$Month <- factor(month(uber$Date.Time))
uber$Day <- factor(day(uber$Date.Time))
uber$Weekday <- factor(wday(uber$Date.Time))
uber$Hour <- factor(hour(uber$Date.Time))
uber$Minute <- factor(minute(uber$Date.Time))
```

```
uber$Second <- factor(second(uber$Date.Time))
```

Q9

Explain why standardization of the data that feeds into the K-means algorithm is not required (in fact,may not even be desirable!) for this particular application of K-means

In this specific case, the 'x' variables represent coordinate points, both expressed in decimal degrees. The centroid results from the K-means algorithm are essentially the mean of these coordinate values. Given that our scope is confined to a relatively compact region (New York City), it's reasonable to regard these centroids as indicative of actual locations within NYC, denoted in decimal degrees.

Q10

Keeping the geography of NYC in mind, make a suggestion for the number of clusters in your analysis.Briefly substantiate your answer.

We could include 5 clusters, as an example, to represent each of the 5 boroughs of NYC. There is no definitive way to pick clusters, so it could make sense to try a few and see which give us the best insights.

Q11,12,13

```
set.seed(1)
kmclusters <- kmeans(uber[,c("Lat","Lon")], 5, nstart = 10)
```

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 226716350)
```

```
kmclusters$size
```

```
## [1]   45632 2068387  238327 2034312  147669
```

```
kmclusters$betweenss/kmclusters$totss
```

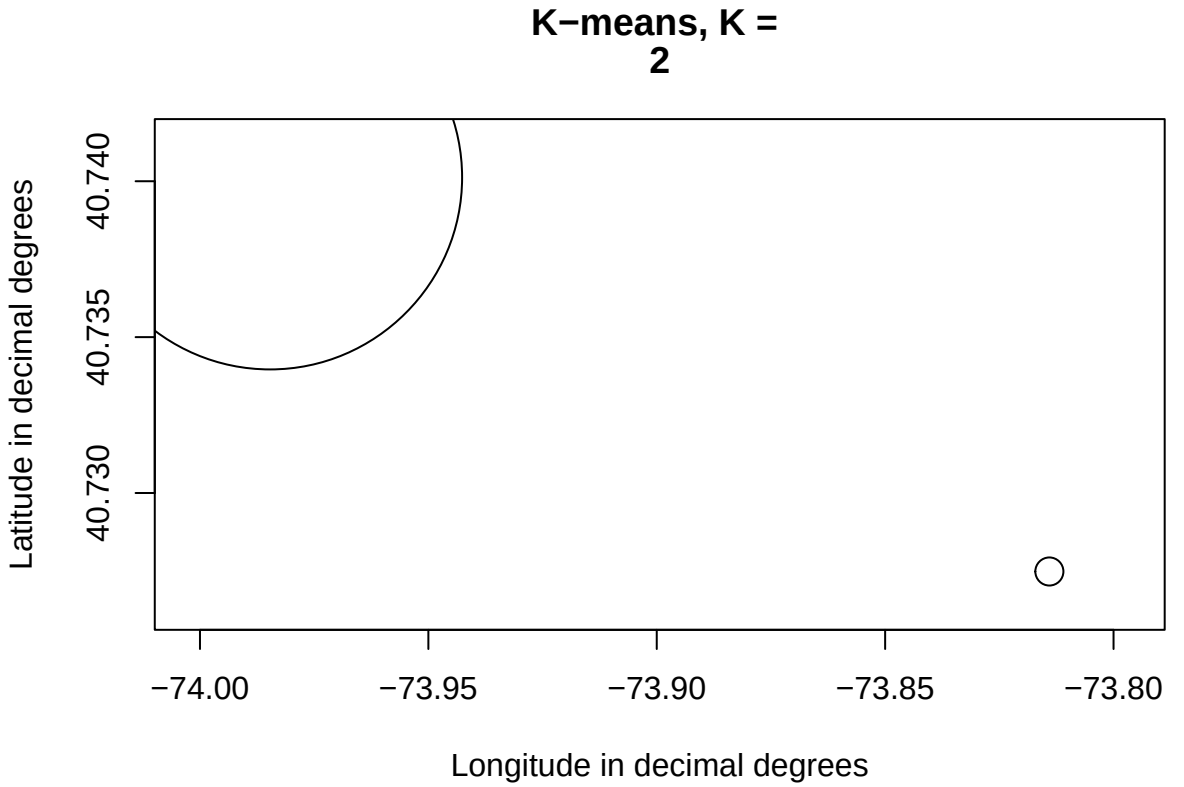
```
## [1] 0.6937608
```

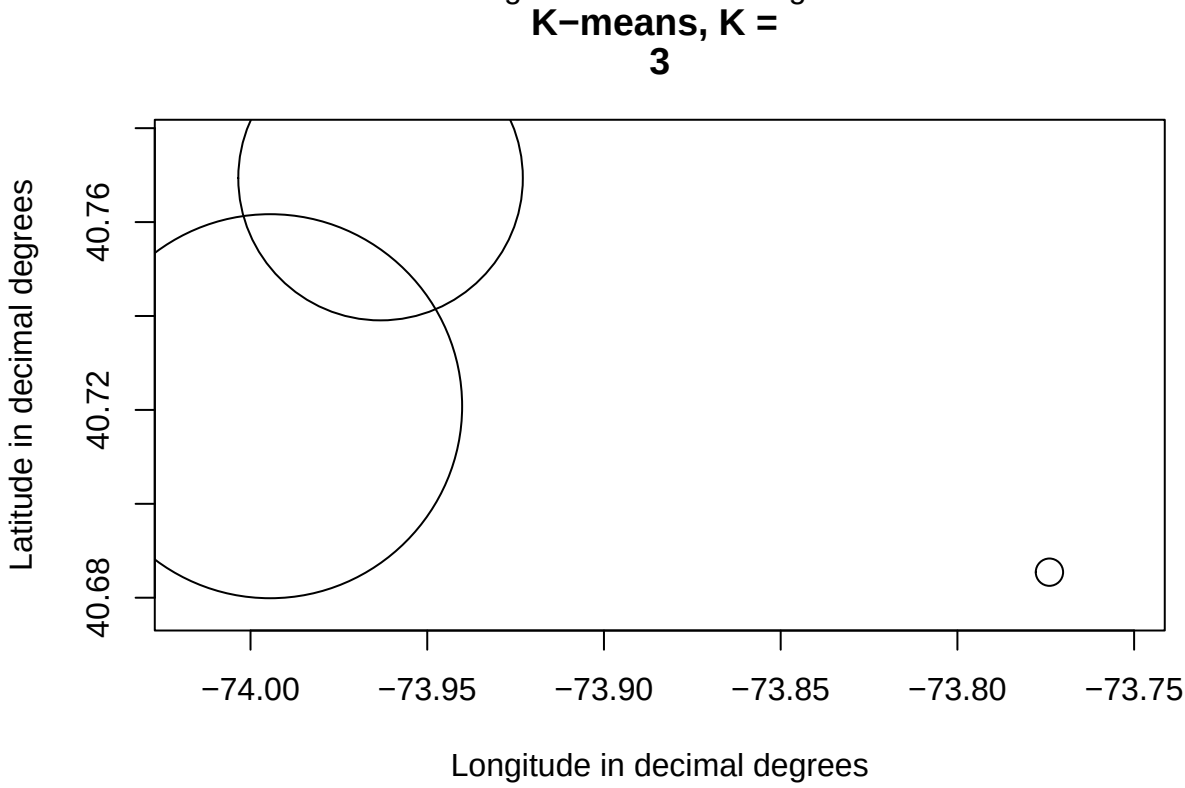
```
kmclusters$centers
```

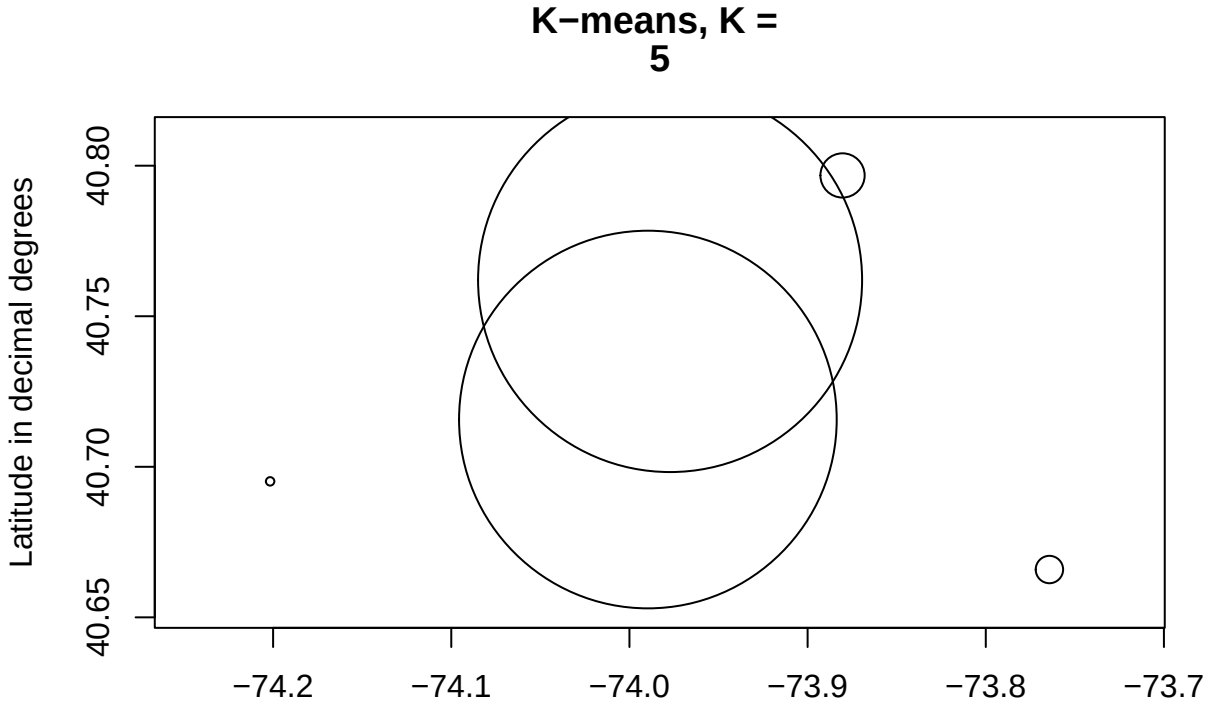
```
##        Lat       Lon
## 1 40.69513 -74.20168
## 2 40.76202 -73.97719
## 3 40.79677 -73.88040
## 4 40.71570 -73.98965
## 5 40.66588 -73.76428
```

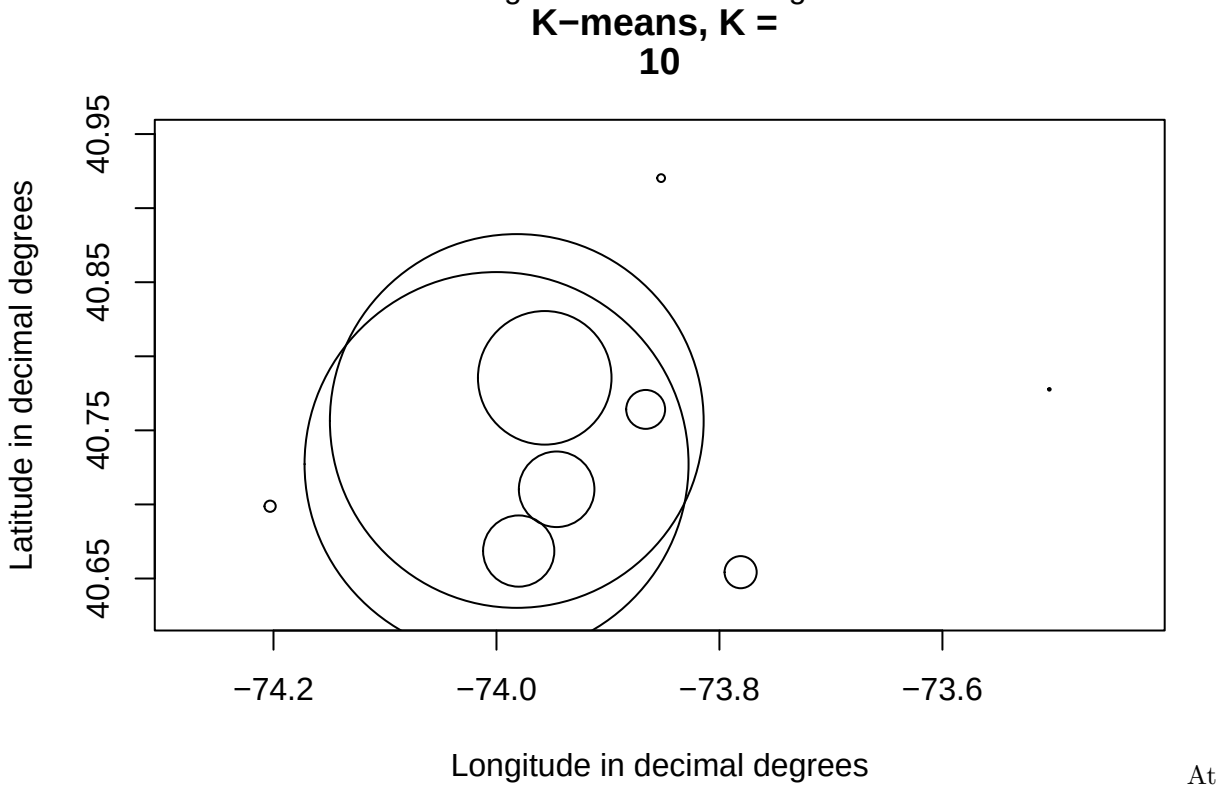
The share of SS accounted for by the clusters is 69%. The centers are at the above coordinates, representing areas on a map. In order to better visualise and see what the data represents, it could be useful to overlay it onto a map.

```
cnums <- c(2,3,5,10)
for (j in cnums){
  kmclusters <- kmeans(uber[,c("Lat","Lon")], j, nstart = 10)
  symbols(kmclusters$centers[,2], kmclusters$centers[,1], circles=kmclusters$size,
          xlab = "Longitude in decimal degrees",
          ylab = "Latitude in decimal degrees",
          main = c("K-means, K = ", j))}
```


K−means, K =
2
Latitude in decimal degrees
40.740
40.735
40.730
−74.00
−73.95
−73.90
−73.85
−73.80
Longitude in decimal degrees


K−means, K =
3
Latitude in decimal degrees
40.76
40.72
40.68
−74.00
−73.95
−73.90
−73.85
−73.80
−73.75
Longitude in decimal degrees





At K = 5, the clusters do indeed seem to represent the boroughs of NYC.