What is Analytics?

The science of using data to build models that lead to better decisions that add value to individuals, companies and institutions.

(Source: Prof. Dimitris Bertsimas, MIT)
Prof. Thomas Davenport, Babson College

The Analytics Edge

1) Analytics provides a competitive edge to individuals, companies and institutions.
2) Analytics is often critical to the success of a company and this is increasingly becoming the norm.

Analytics is often classified into:

1) Descriptive analytics
2) Predictive analytics
3) Prescriptive analytics

Source:
Davenport & Harris.
Competing on Analytics

Companies such as IBM traditionally a PC compar has now invested over $20 billion since 2005 in growing its analytics business.

One of the main reasons for the advanced interest in analytics is that data is often unstructured and comes in various forms. However such data is becoming increasingly available and one tries to develop models that can use such data.

Data might be incomplete and does not come with labels and not directly given as regression or optimization problems.

Analytics is often used in conjunction with Big Data. While more data is better, we can also do analytics with small datasets. In this course, we will work with both small and large datasets.

Predictive analytics is also often used to make predictions at a personal (individual) level. This brings powerful insights but also has privacy issues that need to be dealt with for any organization. While we will not deal with privacy issues in this class, it is important in practice.

Watson & Jeopardy

Jeopardy is a popular TV quiz show launched in 196
The challenge is that the quiz needs Watson to have Natural Language Processing that deals with understanding the human language. The quiz is specifically designed so that the clues are meant to be hard to obtain meaning from. Furthermore the answer must be given in around 5 seconds if the contestant wants to successfully buzz in.

Watson used a fundamental measure accuracy that is the fraction of clues for which it was successfully confident of buzzing in and precision the fraction of times the response was correct.

Watson generated a set of candidate answers using a wide range of encyclopedias, dictionaries, news articles. From identifying relevent articles. For each answer it generated a confidence level to determine the most likely correct ones. By choosing when to buzz using a threshold for the confidence level, it could play more defensively or aggressively. All this needed huge computational memory & power. The Key to Success was to exploit the strength of a computer along with massive parallelization to play a game against the top humans.
Ability of Watson to work with unstructured ~~questions~~ information & answer questions gives it lots of new possibilities

Ensemble methods are an important part of making predictions. Instead of using a single model to make predictions, one can use multiple models to make predictions (ensemble) and then use a majority vote or an average to make better predictions. One has to tradeoff between accuracy and interpretability often in such models.
Watson made extensive use of ensemble methods in predicting the accuracy of answers.

An Introduction to R

www.r-project.org

Why R?

1) R is free and open.

2) R is an integrated suite of software facilities for data manipulation, calculation and graphical display.

3) R provides an environment within which many statistical techniques are implemented and this can be extended easily via packages.

4) R has an extensive online support and discussion forum.

Origins of R (First appeared: 1993)

R is an implementation of the S language developed at Bell Laboratories (formerly AT&T, now Lucent) by John Chambers and colleagues. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland and currently developed by the R Development Core Team of which Chambers is a member.

An Introduction To R Commands

Preliminaries

`getwd( )`	Returns the current working directory
`help(getwd)`	Provides help on a specific function
`?getwd`	Alternative to get help on a function
`Setwd("C:/Users/Ikarthik/Desktop/Analytics")`	Sets the working directory
`dir( )`	lists all files in directory
`ls( )`	Lists (Displays) objects stored within R
`X ← 50`	Assign the number 50 to X
`ls( )`	Display objects stored within R
`50 → X`	Alternate way of assigning 50 to x
`X = 50`	Alternate way of assigning 50 to X
`50* X`	Multiply X with 50
X^2	Square the value in X
`exp(X)`	Compute e^X
`1/X`	Divide 1 by X
`Y ← 1/x`	Assigns y the value of 1/x which is 0.02 here
`rm(Y)`	Removes the variable Y

Numbers & vectors

`x ← c(1, -3, 7, 9.4, pi)` — Concatenates (combines) numbers to form a vector

`y ← c(4)` is equivalent to `y ← 4`

`1/x` — Reciprocal of the numbers are computed in the vector

`y ← c(x, 0, x)` — Combines the vector x, 0 and x to create a vector of length 11

`exp(y)` — Applies the exponentiation operation to each element of y

`y + x` — Creates a new vector of length 11 by recycling the shorter vector (even fractionally) till it matches longer vector

`max(x)`
`min(x)` — Find maximum and minimum elements in x

`which.max(x)` — Determines location (index) of first max

`which(x == max(x))` — Determines all locations where max is obtained

`sum(x)`
`prod(x)` — Sum and product of entries in x

`mean(x)`	Mean of elements in x
`var(x)`	Variance of elements in x Equivalent to: $\text{sum}((x-\text{mean}(x))^2)/(\text{length}(x)-1)$
`max(x,y)`	Returns maximum entry in the vector with entries from x and y
`pmax(x,y)`	Parallel maximum returns a vector (of length equal to their longest argument) that contains in each element the largest element in that position of any vectors in input
`summary(x)`	For a vector, summary provide a six number summary including the min, max, mean, 1st quartile, median and 3rd quartile. This function can also be used with other objects as we will see later
`rm(list=ls())`	Removes all variables from the workspace

In general it is preferred to use ← for assignment instead of =.

The difference for example is clear when you use it inside a function

```
> exp ( X = 1:10)
> X
```

Object x is not found

```
> exp(X <- 1:10)
> X
 1  2  3 ...  10
```

Here x is defined in the user workspace

Note X ← 5 is equivalent to 5 → X

However X = 5 is not equivalent to 5 = X

Note:

X ← 3 (Assigns 3 to x)

X < -3 (Checks if x is less than -3)

`-4:4` Generates the vector C(-4,-3,-2,...,4)

`seq(-4,4, by=0.2)` Generates the vector C(-4,-3.8,...,4)

`rep(X, times=2)` Puts two copies of X end to end

`rep(x, each=2)` Repeats each element of X twice before moving on

`X>1` Returns a logical vector with values TRUE & FALSE by comparing each element of X with 1.

`is.na(X)` Returns a logical vector with values TRUE & FALSE where TRUE is given if the element is not available or a missing value (NA).

`is.na(C(1:3, NA))` Returns the vector FALSE FALSE FALSE TR

`mean(c(1:3,NA), na.rm=TRUE)` Returns mean after removing NA val

`C("a","b")` Returns a character vector

`X[4]` Returns the fourth element of X

`X[c(1,5)]` Returns the first and fifth elements of

`X[!is.na(x)]` Returns a vector containing the non-missing values of X in the same order

`X[4) ← 10` Sets the fourth element of X to 10

```
z ← 0:9
```

`as.integer(as.character(z))` Returns back the integer vector 0 1 2 ... 9

`class(x)` Name of the class of the object x such as numeric, integer, character, logical, list, matrix, data.frame

`as.character(x)` Returns x as a vector of characters

`as.logical(x)` Returns x as a vector of TRUE & FALSE terms where it is FALSE if entry is 0.

`c(1, "a")` Returns the vector "1" "a" by overriding the class of the vector to character

`c(T, F, T)` Returns a logical vector with TRUE, FALSE, TRUE entries

`T` Global variable (logical) whose initial value is set to TRUE

Factors

```
x ← c("yes", "no", "yes", "maybe", "maybe",
      "no", "no", "no")
```

```
class(x)
```
Character

```
factor(x) → y
```
Transforms a vector into a factor (categorical variable

```
levels(y)
```
Returns levels of a factor
Here yes, no, maybe

```
summary(y)
```
Provides a summary of the factor vector

```
table(x)
```
Builds a table from factor

```
income ← c(500, 1200, 4000, 2300, 2300,
           1234, 5439, 432, 4555)
```

```
tapply(income, x, mean)
```
Computes mean of the subvectors in income from the factors in x.

Maybe	No	Yes
33463	1855.25	2250

Matrices, Arrays

`m <- matrix(c(3,4,5,6,7,8), nrow=3, ncol=2)`

Creates a matrix of size 3 x 2 by filling entries by column by default

`dim(m)` Returns dimension of matrix m

`m[4]` Returns fourth entry in matrix counting by columns

`m[1,2]` Returns first row, second column entry in matrix

`m[1, ]` Returns first row in matrix

`class(m)` Returns matrix as class here

`a <- array(c(3,4,5,6,7,8), c(3,2))` Creates an array of size 3 x 2 as the matrix

Arrays can have more than 2 dimensions unlike a matrix

`z <- 1:50` Treats z as an array of

`dim(z) <- c(5,2,5)` dimension 5 x 2 x 5

`z[1,1,1]` Returns value 1

`z[5,2,5]` Returns value 50

`z[5,2,1:5]` Returns vector 10 20 30 40 5

`diag(10)` Returns a diagonal matrix of size 10 x 10 with 1 on diagonal

`cbind(c(1,2,3), c(4,5,6))` Returns matrix

1	4
2	5
3	6

`rbind(c(1,2,3), c(4,5,6))` Returns matrix

1	2	3
4	5	6

```
Z ← matrix (c(5,7,9,6,3,4), nrow=3, ncol=2)
Y ← matrix (c(1,3,0,9,5,-1), nrow=3, ncol=2)
```

```
Z * Y
```

Returns a matrix with componentwise multiplication

```
t(y)
```

Returns transpose of matrix Y

```
Z %*% t(y)
```

Returns a 3×3 matrix using matrix multiplication ZY'

```
help("%*%")
```

```
a ← array(c(2,1,-1,2), c(2,2))
b ← c(4,4)
Solve(a,b)
```

Solves linear equations

$2x - y = 4$

$x + 2y = 4$

to give $x = 2.4$, $Y = 0.8$

```
Solve(a)
```

Returns inverse of matrix a

```
t←eigen(a)
```

Returns eigenvectors and eigenvalues of a with t$values and t$vectors

Lists & data frames

A list is an object consisting of an ordered collection of objects that can be of different or same type

```
karthik <- list(age = 37, sex = "M", child.ages = c(1, 1))
kin <- list(age = 24, sex = "F", child.ages = NA)
```

Creates two lists for karthik & kin respectively

`karthik[[1]]`	Returns number 37
`class(karthik)`	list
`karthik[[2]]`	Returns character "M"
`karthik$age`	Returns number 37
`karthik$child.ages`	Returns vector 1 1
`karthik$child.ages[1]`	Returns number 1
`karthik[1]`	Returns sublist with age and 37
`class(karthik[1])`	Returns "list"
`class(karthik[[1]])`	Returns "numeric"

`t <- c(karthik, kin)` Returns a list with all the objects concatenated into one large vector & loses the dimensions

Data frames are a tightly coupled collection of variables that share many of the properties of matrices & lists, & is often the fundamental data structure used by R modeling software.

A data frame is a list of variables / vectors of the same length

```
t <- data.frame( names = c("Karthik", "Sam", "Jim"),
                 ages = c(36, 34, 40),
                 Sex = c("M", "F", "M"),
                 Children = c(2, 0, 1) )
```

Creates a data frame where columns can be viewed as attributes and rows can be viewed as observations

Often the data will be read from files using the read.csv command or read.table command

`t$names` Lists out "Karthik", "Sam", "Jim"

Any character vector will by default be stored in a data frame as a factor variable.

`t$Spouse <- c("Kit", "Sparsh", "Blake")`

Reading data

`data()`	Lists available datasets in R
`data(faithful)`	Loads the Old Faithful geyser data set in R
`str(faithful)`	Compactly displays internal structure of R object. Here it is a dataframe with 272 observations and 2 variables
`edit(faithful) → new`	Allows you to edit the data and assign it to new
`edit(data.frame()) → new`	Enter new data via spreadsheet interface
`?read.csv`	Read data from a .csv file

Old Faithful Geyser

A geyser is a hot spring that occasionally becomes unstable and erupts hot water and steam in the air. The Old Faithful geyser is at Yellowstone Park, Wyoming.

Simple plots

`plot(faithful)` — Plots a scatter plot of the eruptions & waiting time observations

`hist(faithful$eruptions)` — Plots histogram of the eruptions

`hist(faithful$eruptions, seq(1.6, 5.2, 0.2))` — More detailed histogram

`plot(faithful$eruptions)` — Plots all the observations

`plot(faithful$eruptions, type="l")` — Plots observation with lines

`plot.ecdf(faithful$eruptions)` — Plots the empirical cumulative distribution function

`qqnorm(faithful$eruptions)` — Provides Q-Q plot to create normal quantile plots

`qqnorm(faithful$eruptions[faithful$eruptions > 3])` — Q-Q plot with subset of entries

`qqline(faithful$eruptions[faithful$eruptions > 3])` — Adds a line to data

`boxplot(faithful$eruptions)` — Provides boxplot of data

Old Faithful geyser dateset observations:

1) Eruption times and waiting time between successive eruptions exhibit highly oscillatory behavior: low followed by high and high followed by low

2) Eruption times have a highly bimodal distribution

3) Lower eruption times — Lower waiting times
Higher eruption times — Higher waiting times

This can be used to predict when the next geyser eruption will occur.

During a short eruption, less water and heat are used, so both are restored in shorter time.
During longer eruptions, more time is needed for it to rebuild.

faithful1 ← subset(faithful, faithful$eruptions < 3)

faithful ← subset(faithful, faithful$eruptions ≥ 3)

Creates subset of dataframes with all rows corresponding to cases where the condition is satisfied

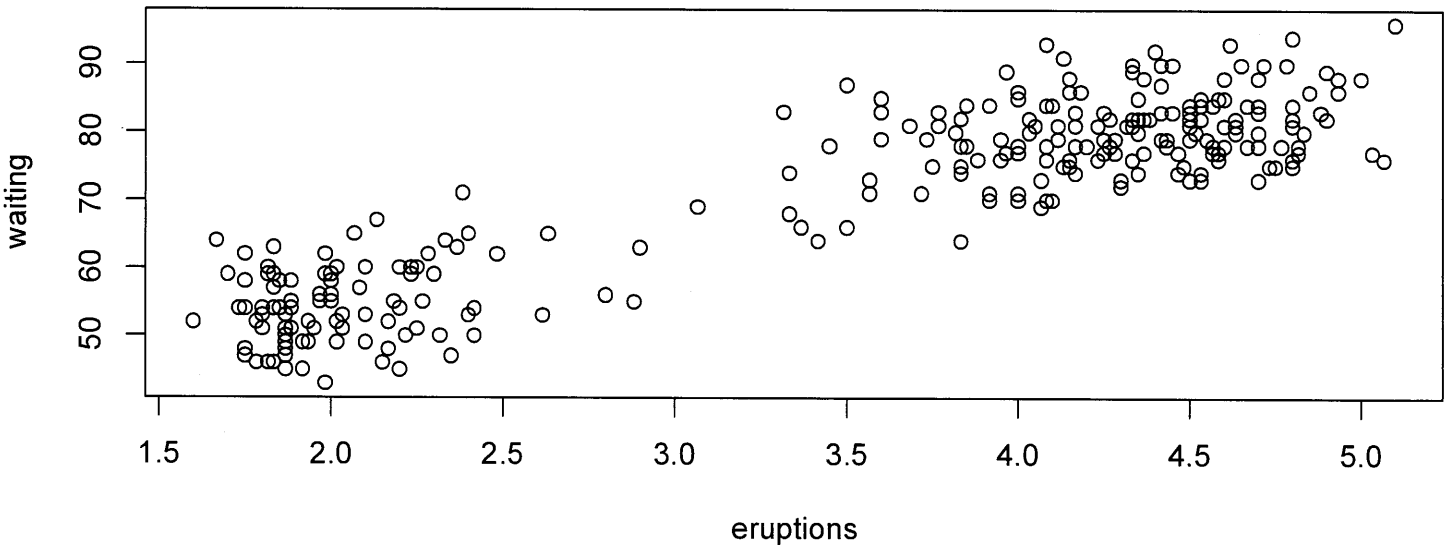
t.test(faithful1$waiting)

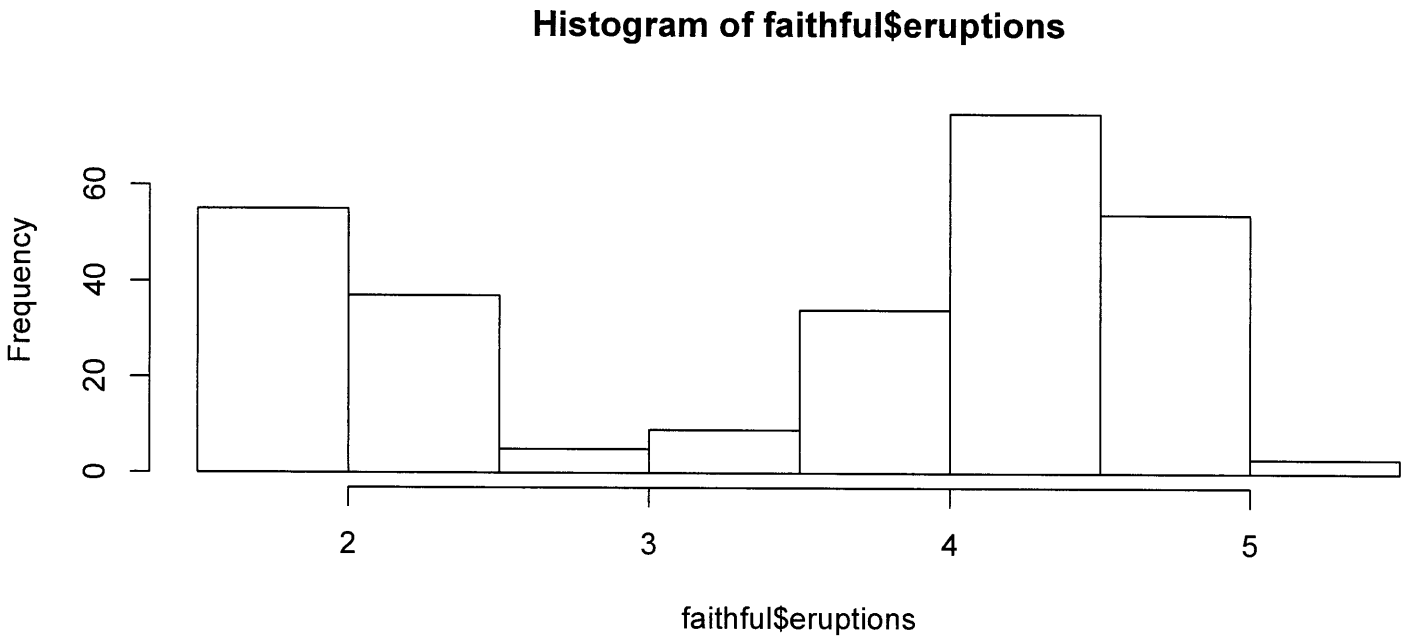
t.test(faithful2$waiting)

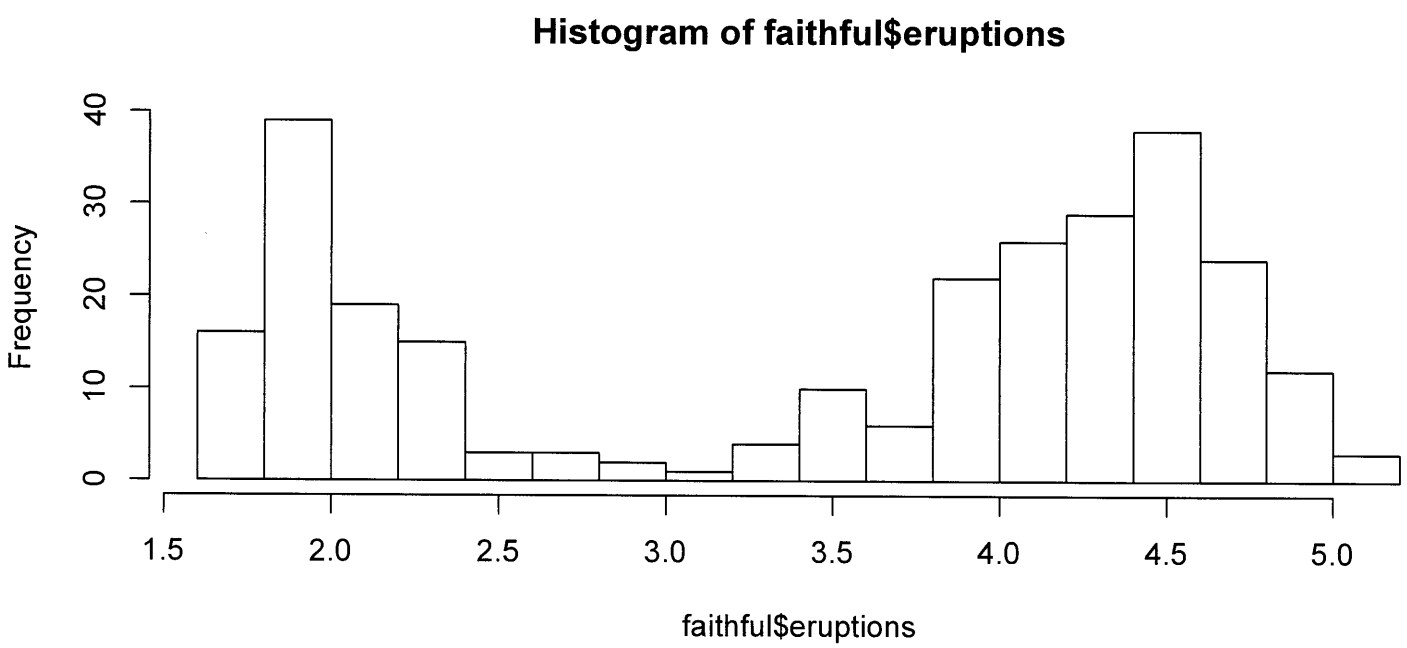
Performs a statistical one sample t-test to test the hypothesis if mean value = 0 and derives 95% confidence interval for mean parameter

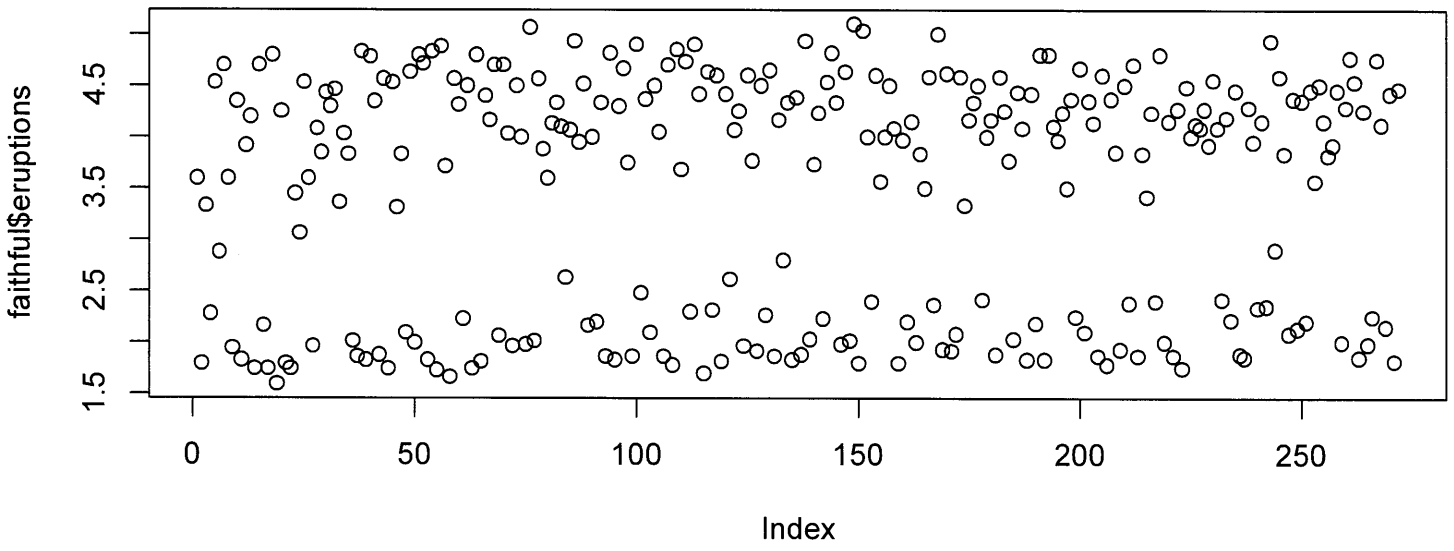
In this case with 95% confidence, if the eruption time < 3, the average waiting time is between 53.3 and 55.67.

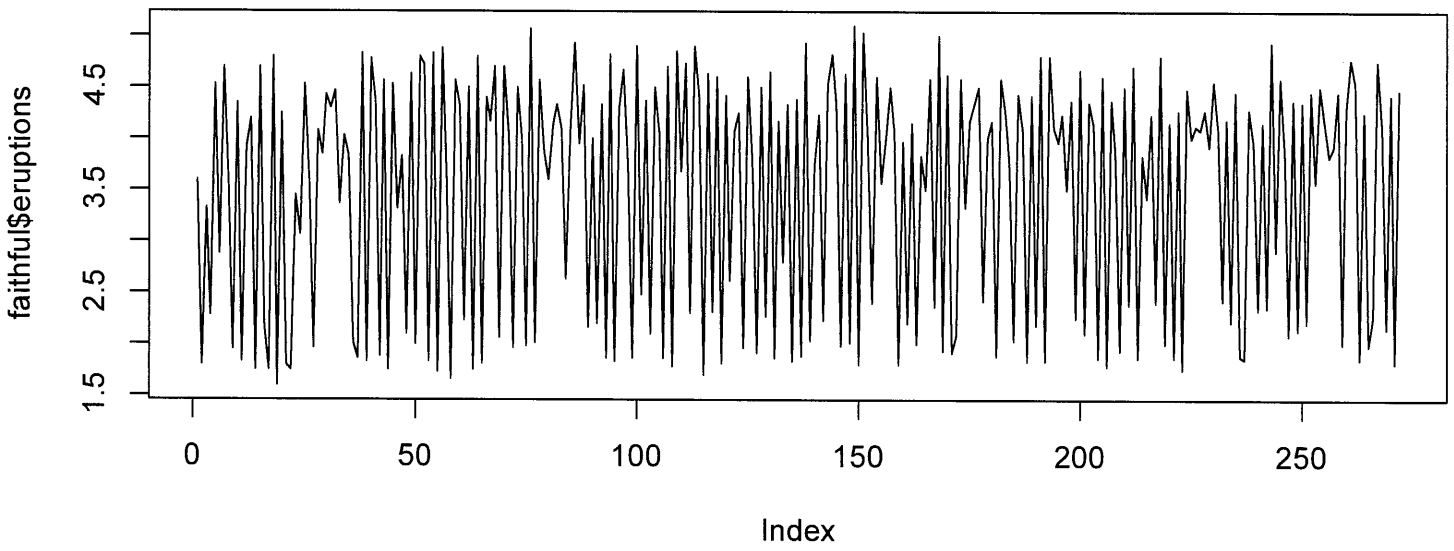
If the eruption time ≥ 3, the average waiting time with 95% confidence is between 79.1 and 80.89

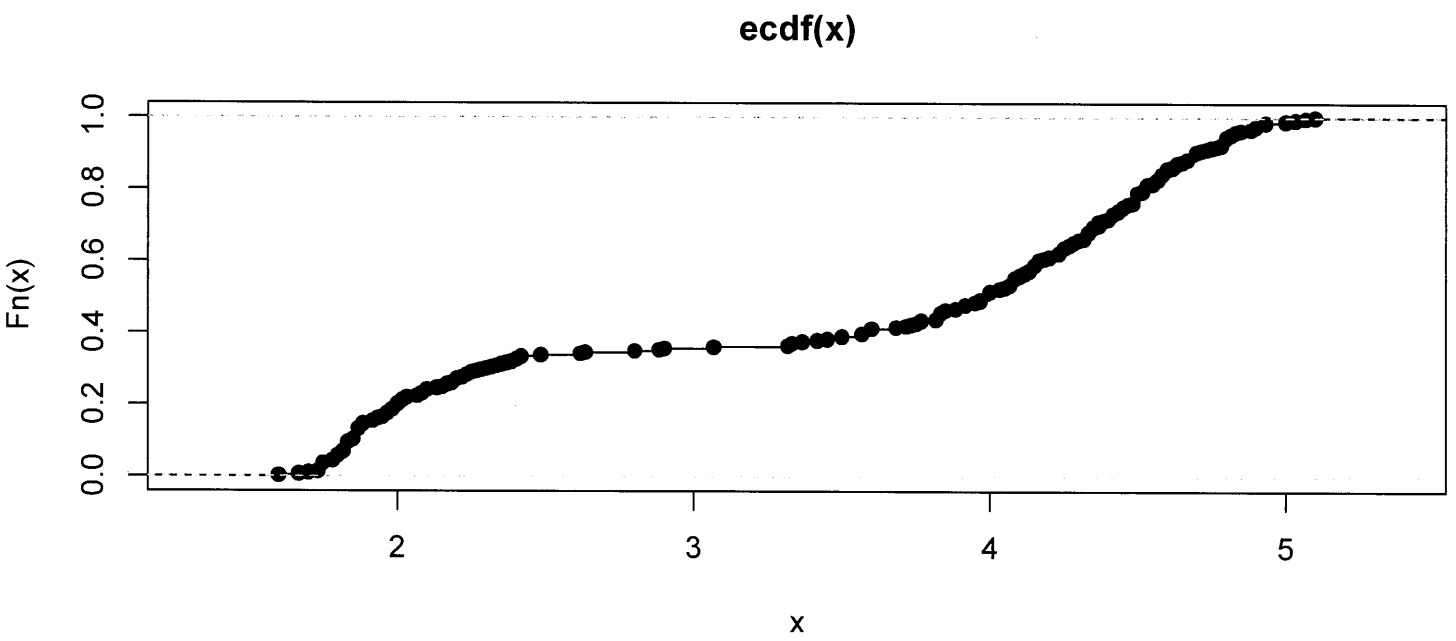

waiting
50
60
70
80
90
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
eruptions

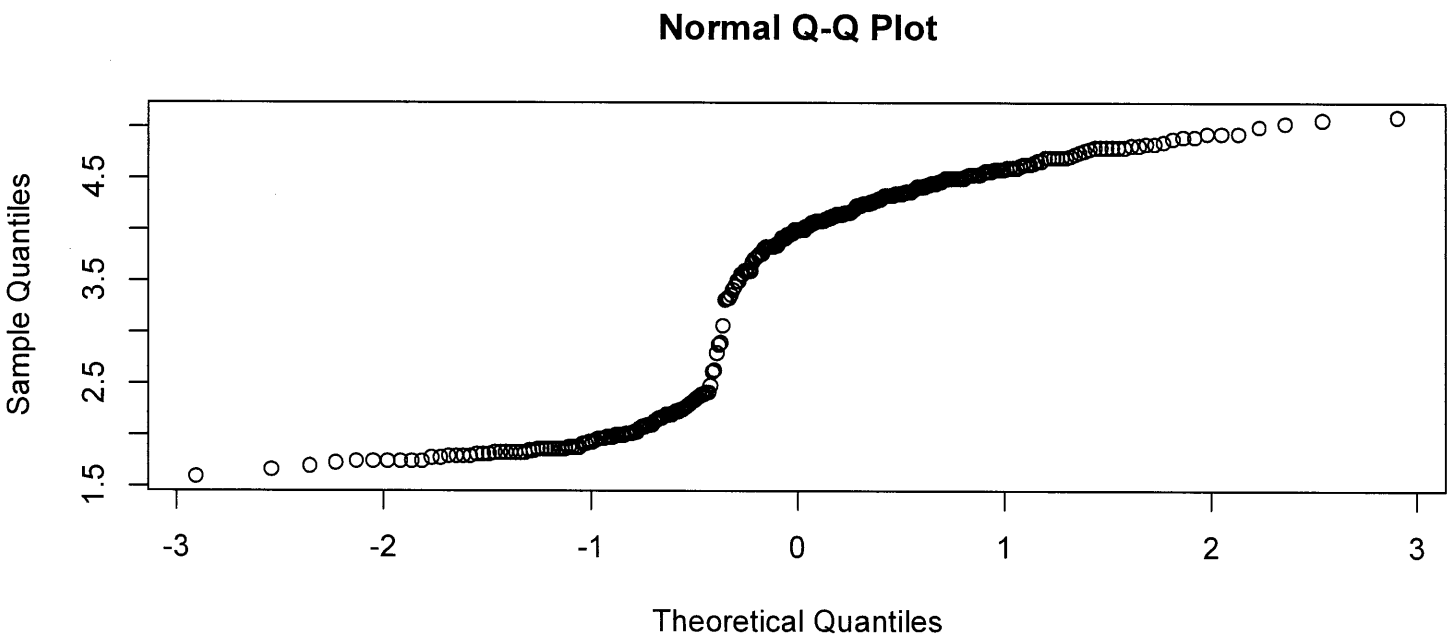

Histogram of faithful$eruptions
Frequency
0
20
40
60
2
3
4
5
faithful$eruptions

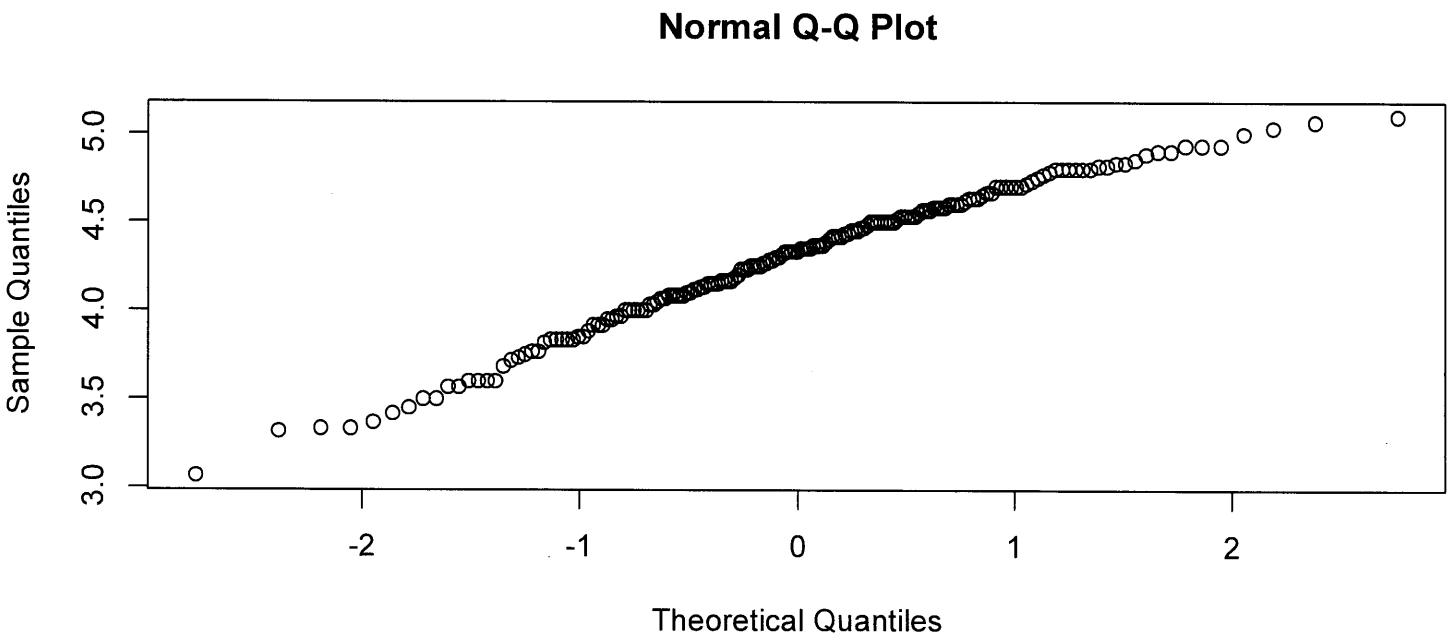

Histogram of faithful$eruptions
Frequency
0
10
20
30
40
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
faithful$eruptions

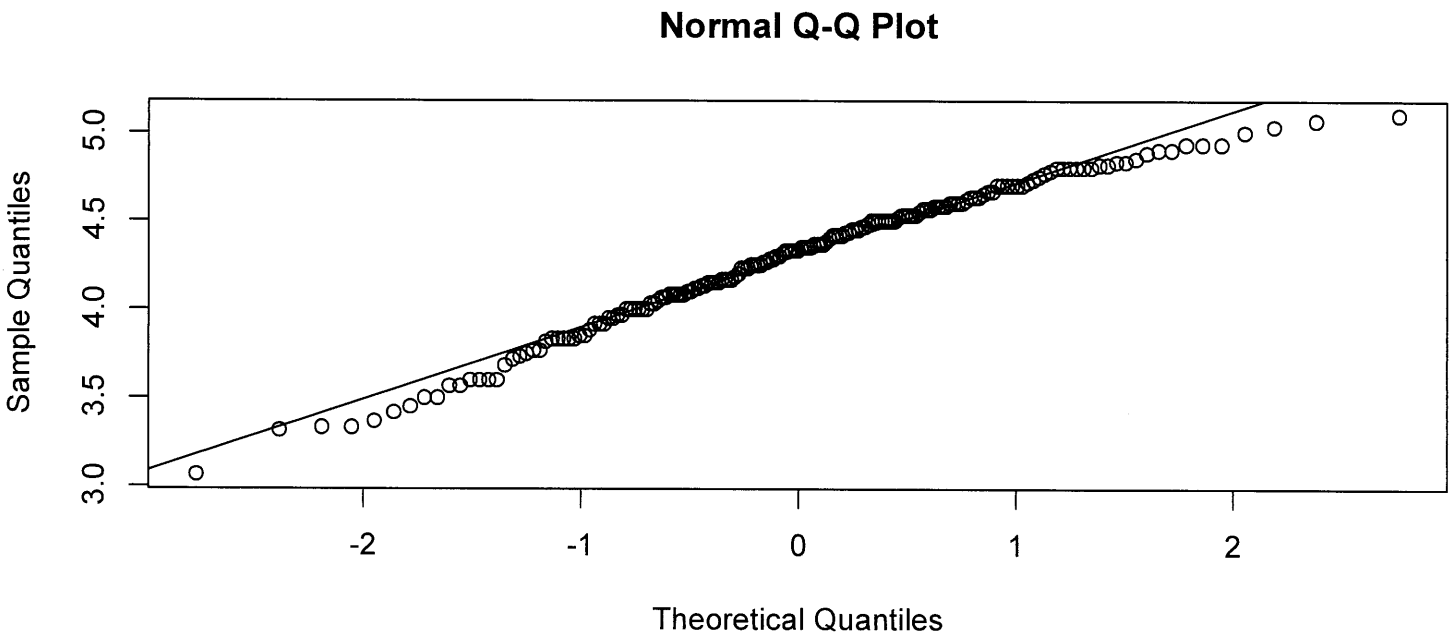

faithful$eruptions
1.5
2.5
3.5
4.5
0
50
100
150
200
250
Index

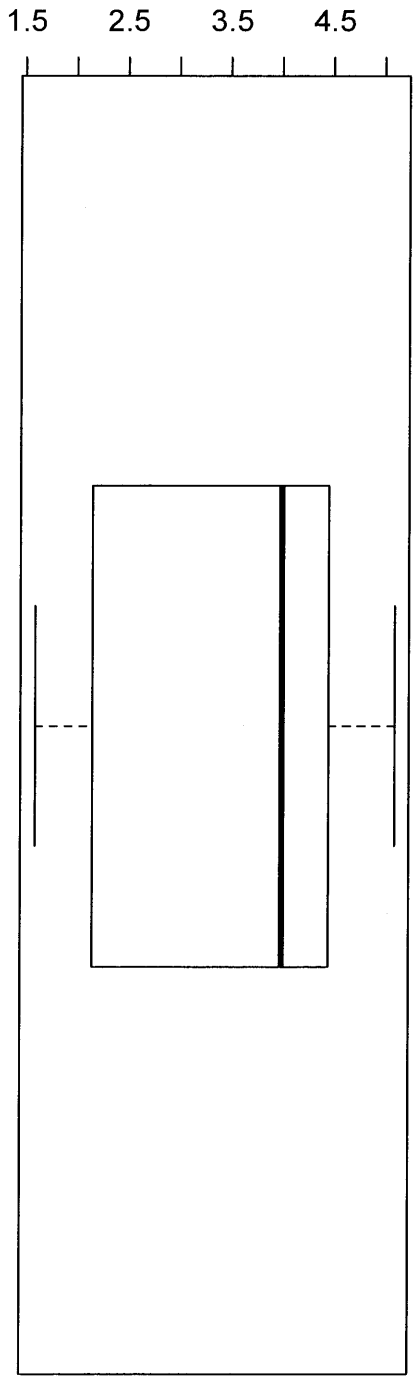

faithful$eruptions
1.5
2.5
3.5
4.5
0
50
100
150
200
250
Index


ecdf(x)
Fn(x)
0.0
0.2
0.4
0.6
0.8
1.0
2
3
4
5
x


Normal Q-Q Plot
Sample Quantiles
1.5
2.5
3.5
4.5
-3
-2
-1
0
1
2
3
Theoretical Quantiles


Normal Q-Q Plot
Sample Quantiles
3.0
3.5
4.0
4.5
5.0
-2
-1
0
1
2
Theoretical Quantiles


Normal Q-Q Plot
Sample Quantiles
3.0
3.5
4.0
4.5
5.0
-2
-1
0
1
2
Theoretical Quantiles


1.5
2.5
3.5
4.5

Data from the World Health Organization

www.who.int/gho/en (Global health observatory data)

```
W <- read.csv("WHO.csv")
```

```
Str(W)
```

This dataset contains data on 194 countries from WHO. The variables are:

Country - name of country

Region - region the country lies in

Population - population in thousands

Under15 - % of population under 15 years of age

Over60 - % of population over 60 years of age

FertilityRate - Average number of children per woman

LifeExpectancy - Life expectancy in years

LiteracyRate - Literacy rate among adults at least 15 years of age

GNI - gross national income per capita

```
which(w$Country == "Singapore")
```

This identifies the row of the dataframe with Singapore

```
Summary(w$Over60)
```

```
Summary(w$Under15)
```

```
w[155,]
```

As this indicates Singapore has a high ratio of population over 60 & low ratio under 15 compared to many countries

plot (wGNI, wFertilityRate)

The figure indicates an inverse relation between income & fertility. This suggests that reproductive restraint arises often as a consequence of economic progress or lower fertility leads to more resources being available per child, making them more productive.

We recreate this with ggplot2 package.

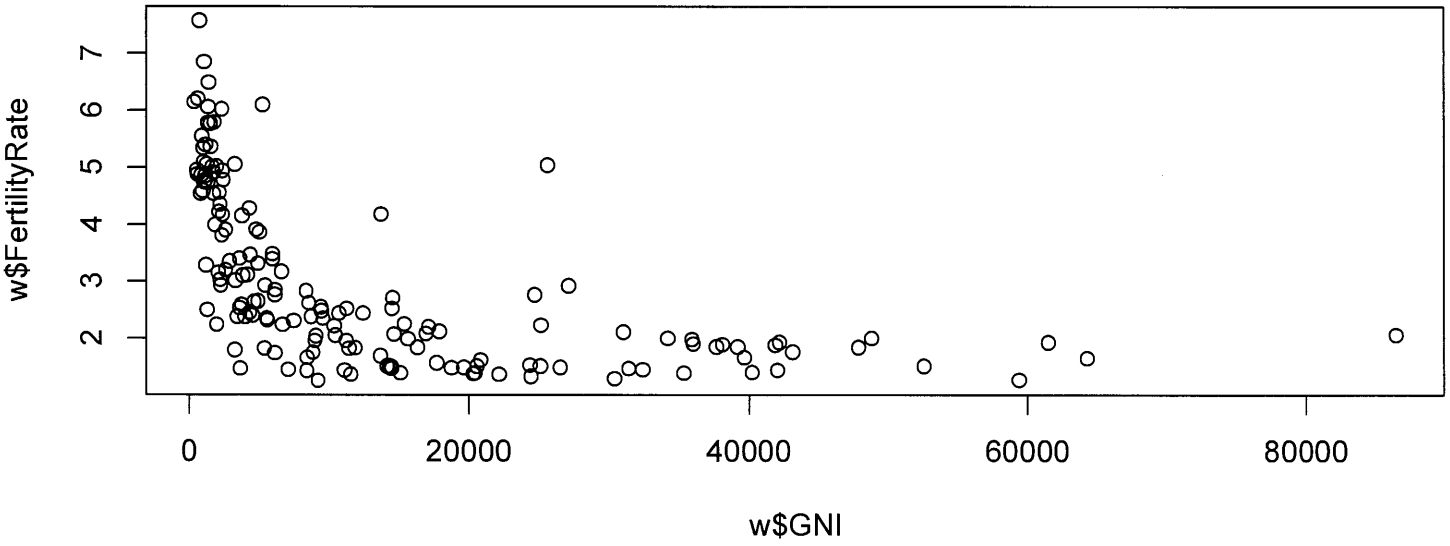
ggplot(w, aes(x= GNI, y= FertilityRate)) + geom_point()

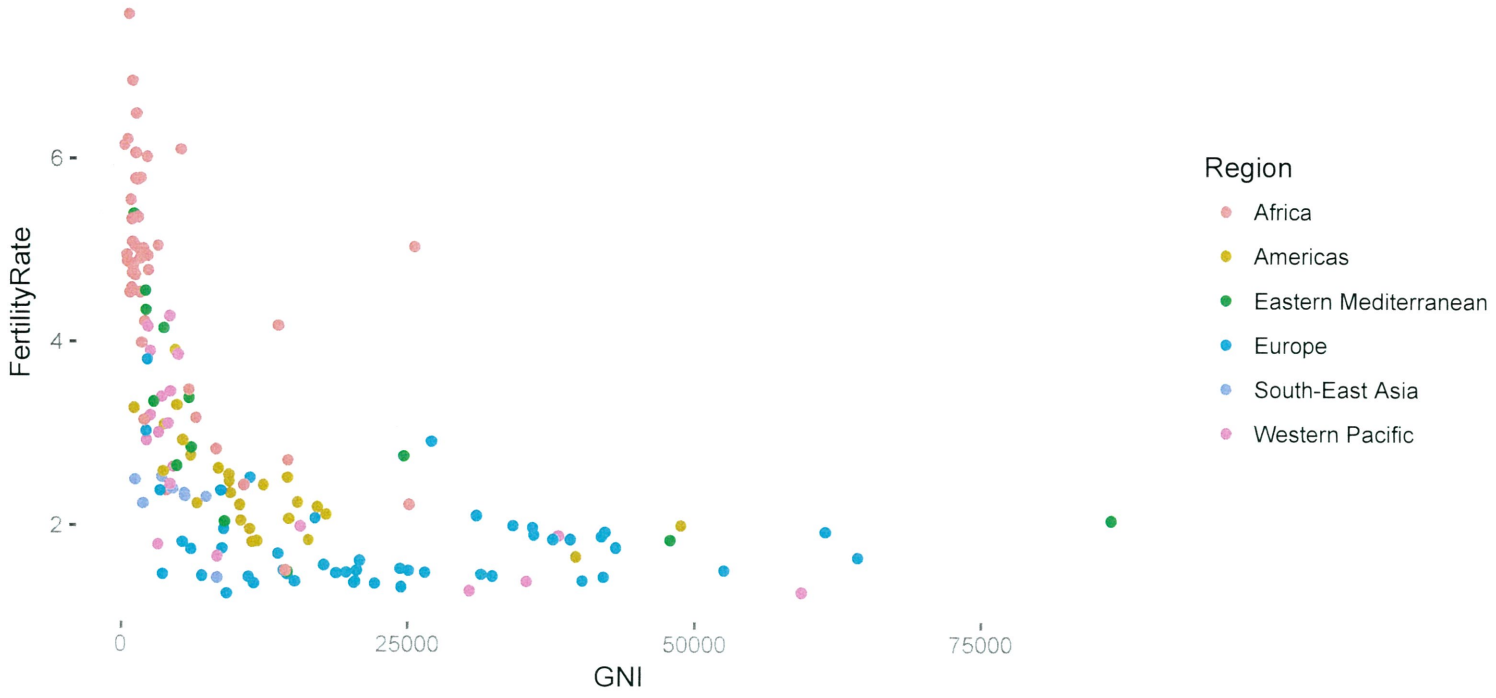
This provides a visual improvement where we add the data and the aesthetic mapping and then the layer with the points.

? ggplot

ggplot (w, aes (x= GNI, y= FertilityRate, color = Region))
+ geom_point ()

This colors the points as per the regions by adding the colors argument to aesthetic option. This helps see that there are trends by regions such as Europe has high GNI, low fertility rates while Africa has low GNI, high fertility rates.


w$FertilityRate
2
3
4
5
6
7
0
20000
40000
60000
80000
w$GNI


FertilityRate
6
4
2
0
25000
50000
75000
GNI
Region
Africa
Americas
Eastern Mediterranean
Europe
South-East Asia
Western Pacific

More advanced visualization in R

One of the advantages of software such as R is that you can access and install packages that provide advanced techniques. The ggplot2 package in R is one such package that improves the basic plotting functionality in R.

In ggplot2, you build a graphic by building layer upon layer where the layers are:

- Data and aesthetic mapping
  Data turns an abstract graphic into a concrete graphic
- Statistical transformation
  Transforms the data by summarizing it such as smooth which calculates the mean of y given x while ensuring smoothness
- Geometric Object
  Controls the type of plot you can create

```
install.packages("ggplot2")
library(ggplot2)
```

Install and load the ggplot2 package before using. The next time you want to run it, you only need to use the library command.

Date from crime location and time in Chicago

Increasingly government organizations are providing access to date to citizens on statistics such as occurrence of crimes. One such website is https://date.cityofchicago.org.

```
Crime ← read.csv("Crime.csv")

Str(Crime)
```

This dataset consists of 191641 observations with date and time, latitude and longitude of location of motor theft crimes.

```
Crime$Date ← strptime(Crime$Date,
                      format = "%m/%d/%y %H %M")

Crime$Weekdays ← weekdays(Crime$Date)

WeekdayCounts ← as.data.frame(table(Crime$Weekdays))
```

The first line converts the date variable to a format that R can work with. The second line extracts the weekday from the date while the third line creates a dataframe that counts the number of crimes on each day of the week. Here WeekdayCounts$Var1 gives the day & WeekdayCounts$Freq gives the number of crimes on the day.

We now create a heatmap to plot the crime data.

Crime$Hour ← Crime$Date$hour

This creates a new variable that captures the hour which is easy to retrieve from the R object now.

WeekDayHourCounts ← as.data.frame(
table(Crime$Weekdays, Crime$Hour))

This creates a data frame with 168 observations where Var1 is the weekday, Var2 is the hour of the crime and Freq is the number of crimes. $(168 = \underbrace{24}_{\text{Hours}} \times \underbrace{7}_{\text{Days}})$.

WeekdayHourCounts$Var1 ←
factor(WeekdayHourCounts$Var1, Ordered = TRUE,
levels = c("Monday", "Tuesday", "wednesday",
"Thursday", "Friday", "Saturday", "Sunday"))

This helps convert it to chronological order rather than alphabetical order.

WeekdayHourCounts$Var2 ←
as.numeric(as.character(WeekdayHourCounts$Var2))

This converts Var2 to a numeric vector from factor. We need to use the intermediate as.character(to deal with how R stores factors.

```
ggplot(WeekdayHourCounts, aes(x=Var2, y=Var1))
+ geom_tile(aes(fill = Freq)) +
  xlab("Hour of day") + ylab("")
```

The figure is a heat map where the lighter color shows more motor vehicle thefts on that day and that hour.

```
ggplot(WeekdayHourCounts, aes(x=Var2, y=Var1)
+ geom_tile(aes(fill = Freq)) + xlab("Hour of day"
+ ylab("")+ scale_fill_gradient(low="white",
                                 high="red")
```

This shows higher frequency values by red and lower by white helping policemen identify hotspots more easily in practice.

As the figure shows, there are more crimes for example Friday nights around 22:00.

We can also overlay such that maps on geographical maps.

```
Install.packages("maps")
library(maps)
Install.packages("ggmap")
library(ggmap)
```

} Use packages maps and ggmap for this display of maps and to combine with Google Maps and other such information

```
Chicago ← get_map(location = "chicago")
ggmap(chicago)
```

This gets a map of Chicago and plots it

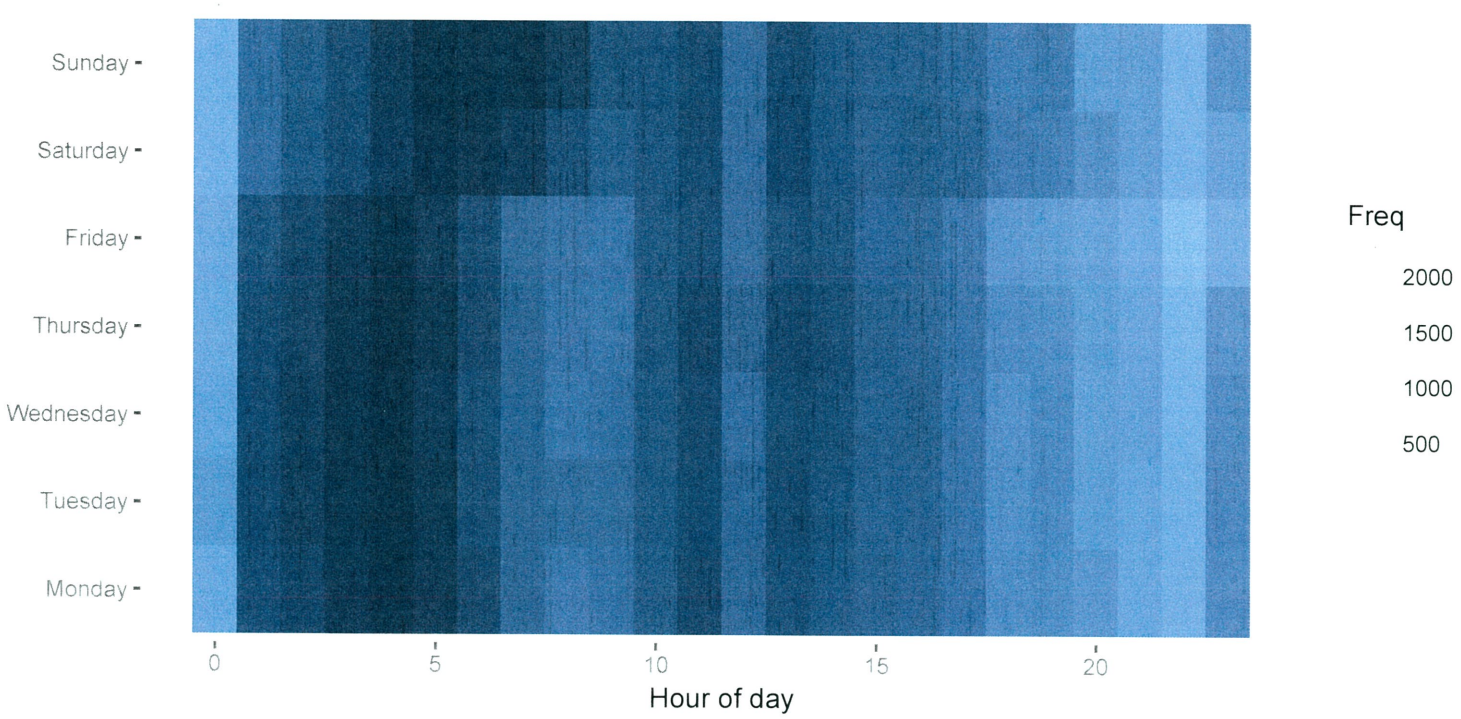
```
LatLongCounts ← as.data.frame(table(round(Crime$
                Longitude,2), round(Crime$Latitude,2)))
```

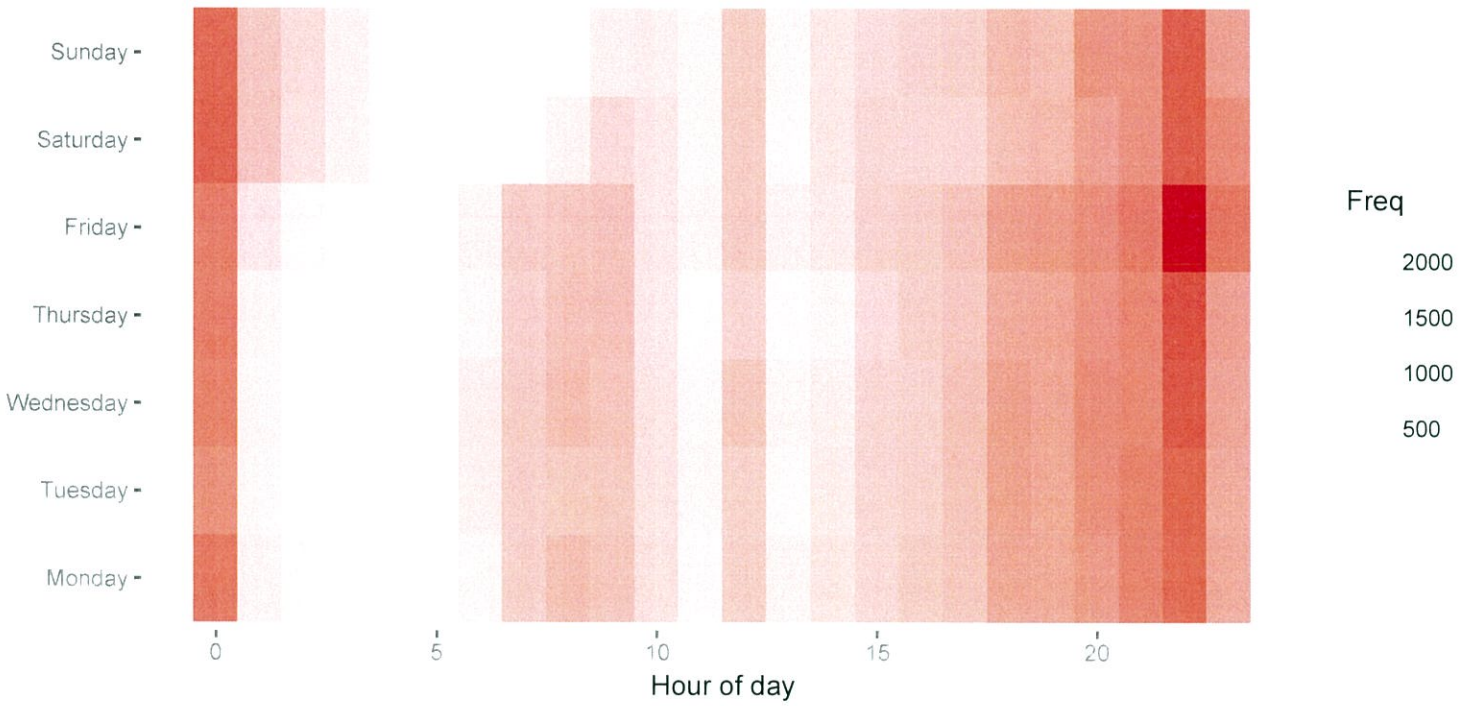
This creates a new data frame that captures latitude & longitude coordinates but rounds it so that the number of parts reduce. We have 1638 observations where Var1 is longitude, Var2 is latitude & Freq is number of thefts in that area.

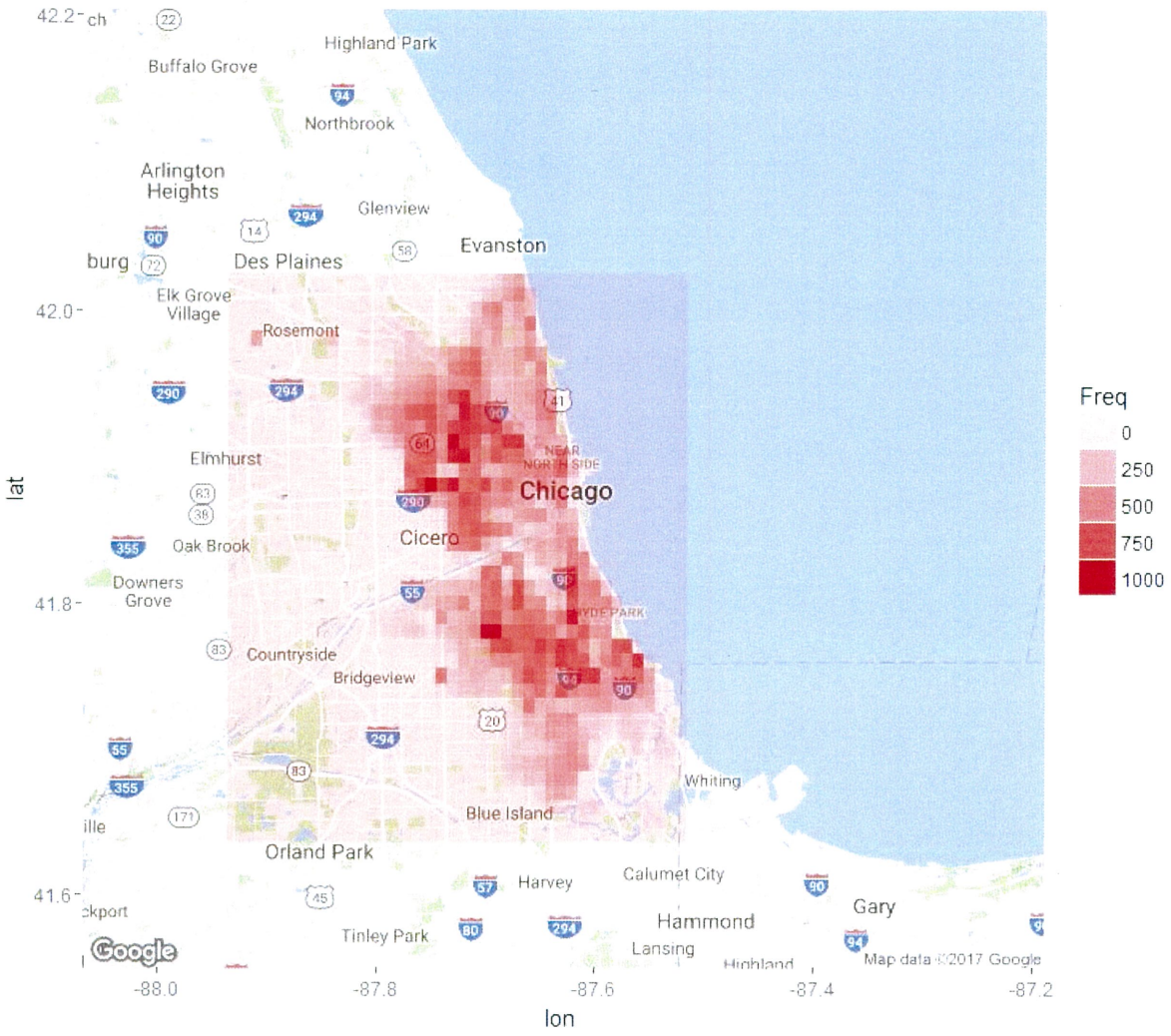
```
LatLongCounts$Var1 ← as.numeric(as.character(
                     LatLongCounts$Var1))
LatLongCounts$Var2 ← as.numeric(as.character(
                     LatLongCounts$Var2))
```

This helps convert factor to numeric.

```
ggmap(chicago) + geom_tile(data = LatLongCounts, aes(x = Var1,
  y = Var2, alpha = Freq), fill = "red")
```


Sunday
Saturday
Friday
Thursday
Wednesday
Tuesday
Monday
0
5
10
15
20
Hour of day
Freq
2000
1500
1000
500


Sunday
Saturday
Friday
Thursday
Wednesday
Tuesday
Monday
0
5
10
15
20
Hour of day
Freq
2000
1500
1000
500


Highland Park
Buffalo Grove
Northbrook
Arlington Heights
Glenview
Evanston
Des Plaines
Elk Grove Village
Rosemont
Elmhurst
Chicago
Cicero
Oak Brook
Downers Grove
Countryside
Bridgeview
Whiting
Blue Island
Orland Park
Harvey
Calumet City
Gary
Hammond
Lansing
Tinley Park
Map data ©2017 Google
42.2
42.0
41.8
41.6
lat
-88.0
-87.8
-87.6
-87.4
-87.2
lon
Freq
0
250
500
750
1000