Predictive Models of COVID-19 for Confirmed Cases in New York, New Jersey, Illinois, Massachusetts and California States

COVID-19 Code

Insert this libraries in order to run the code

```
library(gdata)
library(dplyr)
library(plyr)
library(ggplot2)
library(tibble)
library(treemap)
library(fiftystater)
library(tidyverse)
library(splitstackshape)
library(openintro)
library(caret)
library(caTools)
library(mgcv)
```

Data preprocessing

Concatenation operator definition

```
'%+%' <- function(a,b) {paste(a,b,sep='')}
```

http source

```
http = 'https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_
```

Download all the daily files from its repository

```
m1 = read.csv(http%+% "03-01-2020.csv", header = TRUE)
m2  = read.csv(http%+%"03-02-2020.csv", header = TRUE)
m3 = read.csv(http%+%"03-03-2020.csv", header = TRUE)
m4 = read.csv(http%+%"03-04-2020.csv", header = TRUE)
m5 =read.csv(http%+%"03-05-2020.csv", header = TRUE)
m6 = read.csv(http%+%"03-06-2020.csv", header = TRUE)
m7 = read.csv(http%+%"03-07-2020.csv", header = TRUE)
m8 = read.csv(http%+%"03-08-2020.csv", header = TRUE)
m9 = read.csv(http%+%"03-09-2020.csv", header = TRUE)
m10 = read.csv(http%+%"03-10-2020.csv", header = TRUE)
m11 = read.csv(http%+%"03-11-2020.csv", header = TRUE)
m12= read.csv(http%+%"03-12-2020.csv", header = TRUE)
m13 = read.csv(http%+%"03-13-2020.csv", header = TRUE)
m14 = read.csv(http%+%"03-14-2020.csv", header = TRUE)
m15 = read.csv(http%+%"03-15-2020.csv", header = TRUE)
m16 = read.csv(http%+%"03-16-2020.csv", header = TRUE)
```

```r
m17 = read.csv(http%+%"03-17-2020.csv", header = TRUE)
m18 = read.csv(http%+%"03-18-2020.csv", header = TRUE)
m19 = read.csv(http%+%"03-19-2020.csv", header = TRUE)
m20 = read.csv(http%+%"03-20-2020.csv", header = TRUE)
m21 = read.csv(http%+%"03-21-2020.csv", header = TRUE)
m22 = read.csv(http%+%"03-22-2020.csv", header = TRUE)
m23 = read.csv(http%+%"03-23-2020.csv", header = TRUE)
m24 = read.csv(http%+%"03-24-2020.csv", header = TRUE)
m25 = read.csv(http%+%"03-25-2020.csv", header = TRUE)
m26 = read.csv(http%+%"03-26-2020.csv", header = TRUE)
m27 = read.csv(http%+%"03-27-2020.csv", header = TRUE)
m28 = read.csv(http%+%"03-28-2020.csv", header = TRUE)
m29 =read.csv(http%+%"03-29-2020.csv", header = TRUE)
m30 =read.csv(http%+%"03-30-2020.csv", header = TRUE)
m31 =read.csv(http%+%"03-30-2020.csv", header = TRUE)
a1 = read.csv(http%+%"04-01-2020.csv", header = TRUE)
a2  = read.csv(http%+%"04-02-2020.csv", header = TRUE)
a3 = read.csv(http%+%"04-03-2020.csv", header = TRUE)
a4 = read.csv(http%+%"04-04-2020.csv", header = TRUE)
a5 =read.csv(http%+%"04-05-2020.csv", header = TRUE)
a6 = read.csv(http%+%"04-06-2020.csv", header = TRUE)
a7 = read.csv(http%+%"04-07-2020.csv", header = TRUE)
a8 = read.csv(http%+%"04-08-2020.csv", header = TRUE)
a9 = read.csv(http%+%"04-09-2020.csv", header = TRUE)
a10 = read.csv(http%+%"04-10-2020.csv", header = TRUE)
a11 = read.csv(http%+%"04-11-2020.csv", header = TRUE)
a12= read.csv(http%+%"04-12-2020.csv", header = TRUE)
a13 = read.csv(http%+%"04-13-2020.csv", header = TRUE)
a14 = read.csv(http%+%"04-14-2020.csv", header = TRUE)
a15 = read.csv(http%+%"04-15-2020.csv", header = TRUE)
a16 = read.csv(http%+%"04-16-2020.csv", header = TRUE)
a17 = read.csv(http%+%"04-17-2020.csv", header = TRUE)
a18 = read.csv(http%+%"04-18-2020.csv", header = TRUE)
a19 = read.csv(http%+%"04-19-2020.csv", header = TRUE)
a20 = read.csv(http%+%"04-20-2020.csv", header = TRUE)
a21 = read.csv(http%+%"04-21-2020.csv", header = TRUE)
a22 = read.csv(http%+%"04-22-2020.csv", header = TRUE)
a23 = read.csv(http%+%"04-23-2020.csv", header = TRUE)
a24 = read.csv(http%+%"04-24-2020.csv", header = TRUE)
a25 = read.csv(http%+%"04-25-2020.csv", header = TRUE)
a26 = read.csv(http%+%"04-26-2020.csv", header = TRUE)
a27 = read.csv(http%+%"04-27-2020.csv", header = TRUE)
a28 = read.csv(http%+%"04-28-2020.csv", header = TRUE)
a29 =read.csv(http%+%"04-29-2020.csv", header = TRUE)
a30 =read.csv(http%+%"04-30-2020.csv", header = TRUE)
ma1 = read.csv(http%+%"05-01-2020.csv", header = TRUE)
ma2  = read.csv(http%+%"05-02-2020.csv", header = TRUE)
ma3 = read.csv(http%+%"05-03-2020.csv", header = TRUE)
ma4 = read.csv(http%+%"05-04-2020.csv", header = TRUE)
ma5 =read.csv(http%+%"05-05-2020.csv", header = TRUE)
ma6 = read.csv(http%+%"05-06-2020.csv", header = TRUE)
ma7 = read.csv(http%+%"05-07-2020.csv", header = TRUE)
ma8 = read.csv(http%+%"05-08-2020.csv", header = TRUE)
```

```r
ma9 = read.csv(http%+%"05-09-2020.csv", header = TRUE)
ma10 = read.csv(http%+%"05-10-2020.csv", header = TRUE)
ma11 = read.csv(http%+%"05-11-2020.csv", header = TRUE)
ma12= read.csv(http%+%"05-12-2020.csv", header = TRUE)
ma13 = read.csv(http%+%"05-13-2020.csv", header = TRUE)
ma14 = read.csv(http%+%"05-14-2020.csv", header = TRUE)
ma15 = read.csv(http%+%"05-15-2020.csv", header = TRUE)
ma16 = read.csv(http%+%"05-16-2020.csv", header = TRUE)
ma17 = read.csv(http%+%"05-17-2020.csv", header = TRUE)
ma18 = read.csv(http%+%"05-18-2020.csv", header = TRUE)
ma19 = read.csv(http%+%"05-19-2020.csv", header = TRUE)
ma20 = read.csv(http%+%"05-20-2020.csv", header = TRUE)
```

Add date column

```r
m1 = cbind(m1, c(as.Date("2020-03-01") ))
m2 =cbind(m2, c(as.Date("2020-03-02") ))
m3 =cbind(m3, c(as.Date("2020-03-03") ))
m4 =cbind(m4, c(as.Date("2020-03-04") ))
m5 =cbind(m5, c(as.Date("2020-03-05") ))
m6 = cbind(m6, c(as.Date("2020-03-06") ))
m7 = cbind(m7, c(as.Date("2020-03-07") ))
m8= cbind(m8, c(as.Date("2020-03-08") ))
m9 =cbind(m9, c(as.Date("2020-03-09") ))
m10 = cbind(m10, c(as.Date("2020-03-10") ))
m11 = cbind(m11, c(as.Date("2020-03-11") ))
m12  = cbind(m12, c(as.Date("2020-03-12") ))
m13 = cbind(m13, c(as.Date("2020-03-13") ))
m14 =cbind(m14, c(as.Date("2020-03-14") ))
m15 =cbind(m15, c(as.Date("2020-03-15") ))
m16 = cbind(m16, c(as.Date("2020-03-16") ))
m17 = cbind(m17, c(as.Date("2020-03-17") ))
m18 = cbind(m18, c(as.Date("2020-03-18") ))
m19 = cbind(m19, c(as.Date("2020-03-19") ))
m20 = cbind(m20, c(as.Date("2020-03-20") ))
m21 = cbind(m21, c(as.Date("2020-03-21") ))
m22  = cbind(m22, c(as.Date("2020-03-22") ))
m23 = cbind(m23, c(as.Date("2020-03-23") ))
m24 = cbind(m24, c(as.Date("2020-03-24") ))
m25 = cbind(m25, c(as.Date("2020-03-25") ))
m26 = cbind(m26, c(as.Date("2020-03-26") ))
m27 = cbind(m27, c(as.Date("2020-03-27") ))
m28 = cbind(m28, c(as.Date("2020-03-28") ))
m29= cbind(m29, c(as.Date("2020-03-29") ))
m30 = cbind(m30, c(as.Date("2020-03-30") ))
m31 = cbind(m31, c(as.Date("2020-03-31") ))
a1 = cbind(a1, c(as.Date("2020-04-01") ))
a2 =cbind(a2, c(as.Date("2020-04-02") ))
a3 =cbind(a3, c(as.Date("2020-04-03") ))
a4 =cbind(a4, c(as.Date("2020-04-04") ))
a5 =cbind(a5, c(as.Date("2020-04-05") ))
a6 = cbind(a6, c(as.Date("2020-04-06") ))
a7 = cbind(a7, c(as.Date("2020-04-07") ))
a8= cbind(a8, c(as.Date("2020-04-08") ))
```

```
a9 =cbind(a9, c(as.Date("2020-04-09") ))
a10 = cbind(a10, c(as.Date("2020-04-10") ))
a11 = cbind(a11, c(as.Date("2020-04-11") ))
a12  = cbind(a12, c(as.Date("2020-04-12") ))
a13 = cbind(a13, c(as.Date("2020-04-13") ))
a14 =cbind(a14, c(as.Date("2020-04-14") ))
a15 =cbind(a15, c(as.Date("2020-04-15") ))
a16 = cbind(a16, c(as.Date("2020-04-16") ))
a17 = cbind(a17, c(as.Date("2020-04-17") ))
a18 = cbind(a18, c(as.Date("2020-04-18") ))
a19 = cbind(a19, c(as.Date("2020-04-19") ))
a20 = cbind(a20, c(as.Date("2020-04-20") ))
a21 = cbind(a21, c(as.Date("2020-04-21") ))
a22  = cbind(a22, c(as.Date("2020-04-22") ))
a23 = cbind(a23, c(as.Date("2020-04-23") ))
a24 = cbind(a24, c(as.Date("2020-04-24") ))
a25 = cbind(a25, c(as.Date("2020-04-25") ))
a26 = cbind(a26, c(as.Date("2020-04-26") ))
a27 = cbind(a27, c(as.Date("2020-04-27") ))
a28 = cbind(a28, c(as.Date("2020-04-28") ))
a29= cbind(a29, c(as.Date("2020-04-29") ))
a30 = cbind(a30, c(as.Date("2020-04-30") ))
ma1 = cbind(ma1, c(as.Date("2020-05-01") ))
ma2 =cbind(ma2, c(as.Date("2020-05-02") ))
ma3 =cbind(ma3, c(as.Date("2020-05-03") ))
ma4 =cbind(ma4, c(as.Date("2020-05-04") ))
ma5 =cbind(ma5, c(as.Date("2020-05-05") ))
ma6 = cbind(ma6, c(as.Date("2020-05-06") ))
ma7 = cbind(ma7, c(as.Date("2020-05-07") ))
ma8= cbind(ma8, c(as.Date("2020-05-08") ))
ma9 =cbind(ma9, c(as.Date("2020-05-09") ))
ma10 = cbind(ma10, c(as.Date("2020-05-10") ))
ma11 = cbind(ma11, c(as.Date("2020-05-11") ))
ma12  = cbind(ma12, c(as.Date("2020-05-12") ))
ma13 = cbind(ma13, c(as.Date("2020-05-13") ))
ma14 =cbind(ma14, c(as.Date("2020-05-14") ))
ma15 =cbind(ma15, c(as.Date("2020-05-15") ))
ma16 =cbind(ma16, c(as.Date("2020-05-16") ))
ma17 =cbind(ma17, c(as.Date("2020-05-17") ))
ma18 =cbind(ma18, c(as.Date("2020-05-18") ))
ma19 =cbind(ma19, c(as.Date("2020-05-19") ))
ma20 =cbind(ma20, c(as.Date("2020-05-20") ))
```

Change a column for compatibility with packages

```
names(m1)[9] <- "date"
names(m2)[9] <- "date"
names(m3)[9] <- "date"
names(m4)[9] <- "date"
names(m5)[9] <- "date"
names(m6)[9] <- "date"
names(m7)[9] <- "date"
names(m8)[9] <- "date"
names(m9)[9] <- "date"
```

```
names(m10)[9] <- "date"
names(m11)[9] <- "date"
names(m12)[9] <- "date"
names(m13)[9] <- "date"
names(m14)[9] <- "date"
names(m15)[9] <- "date"
names(m16)[9] <- "date"
names(m17)[9] <- "date"
names(m18)[9] <- "date"
names(m19)[9] <- "date"
names(m20)[9] <- "date"
names(m21)[9] <- "date"
names(m22)[13] <- "date"
names(m23)[13] <- "date"
names(m24)[13] <- "date"
names(m25)[13] <- "date"
names(m26)[13] <- "date"
names(m27)[13] <- "date"
names(m28)[13] <- "date"
names(m29)[13] <- "date"
names(m30)[13] <- "date"
names(m31)[13] <- "date"
names(a1)[13] <- "date"
names(a2)[13] <- "date"
names(a3)[13] <- "date"
names(a4)[13] <- "date"
names(a5)[13] <- "date"
names(a6)[13] <- "date"
names(a7)[13] <- "date"
names(a8)[13] <- "date"
names(a9)[13] <- "date"
names(a10)[13] <- "date"
names(a11)[13] <- "date"
names(a12)[13] <- "date"
names(a13)[13] <- "date"
names(a14)[13] <- "date"
names(a15)[13] <- "date"
names(a16)[13] <- "date"
names(a17)[13] <- "date"
names(a18)[13] <- "date"
names(a19)[13] <- "date"
names(a20)[13] <- "date"
names(a21)[13] <- "date"
names(a22)[13] <- "date"
names(a23)[13] <- "date"
names(a24)[13] <- "date"
names(a25)[13] <- "date"
names(a26)[13] <- "date"
names(a27)[13] <- "date"
names(a28)[13] <- "date"
names(a29)[13] <- "date"
names(a30)[13] <- "date"
names(ma1)[13] <- "date"
```

```r
names(ma2)[13] <- "date"
names(ma3)[13] <- "date"
names(ma4)[13] <- "date"
names(ma5)[13] <- "date"
names(ma6)[13] <- "date"
names(ma7)[13] <- "date"
names(ma8)[13] <- "date"
names(ma9)[13] <- "date"
names(ma10)[13] <- "date"
names(ma11)[13] <- "date"
names(ma12)[13] <- "date"
names(ma13)[13] <- "date"
names(ma14)[13] <- "date"
names(ma15)[13] <- "date"
names(ma16)[13] <- "date"
names(ma17)[13] <- "date"
names(ma18)[13] <- "date"
names(ma19)[13] <- "date"
names(ma20)[13] <- "date"
names(m13)[1] <- "Province.State"
names(m22)[1] <- "FIPS"
```

Concatenate the daily files that have a consistent registry

```r
p1= rbind(m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11,m12,m13,m14,m15,m16,m17,m18,m19,m20,m21)
p2= rbind(m22,m23,m24,m25,m26,m27,m28,m29,m30,m31,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a1
```

Store a day dataset for later use in coordinates

```r
coord <- p1
```

Delete unused columns

```r
p2$FIPS <- p2$Last_Update <- p2$Active <- p2$Combined_Key <- p2$Lat <-p2$Long_ <- NULL
p1$Last.Update <- p1$Latitude <- p1$Longitude <- NULL
p2$Admin2 <- NULL
```

Change column heads

```r
names(p2)[1] <- "Province.State"
names(p2)[2] <- "Country.Region"
```

Filter US only regions

```r
p1 = filter(p1, Country.Region == "US")
p2 = filter(p2, Country.Region == "US")
```

Delete regions that are not mainland

```r
p1<-p1 %>%
  filter(Province.State!="Grand Princess",Province.State!="Guam" , Province.State!="Northern Mariana Isl

p2<-p2 %>%
  filter(Province.State!="Grand Princess",Province.State!="Guam" , Province.State!="Northern Mariana Isl
```

Here we sum the numbers from the same state, to avoid the dates being summed We extracted and re-added them

March

```r
dm22= filter(p2, date == "2020-03-22")
dm22$Country.Region <- dm22$date <- NULL
dm22 <- dm22 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once per session.
```

```r
dm22= cbind(dm22, c(as.Date("2020-03-22") ))
names(dm22)[5] <- "date"

dm23= filter(p2, date == "2020-03-23")
dm23$Country.Region <- dm23$date <- NULL
dm23 <- dm23 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dm23= cbind(dm23, c(as.Date("2020-03-23") ))
names(dm23)[5] <- "date"

dm24= filter(p2, date == "2020-03-24")
dm24$Country.Region <- dm24$date <- NULL
dm24 <- dm24 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dm24= cbind(dm24, c(as.Date("2020-03-24") ))
names(dm24)[5] <- "date"

dm25= filter(p2, date == "2020-03-25")
dm25$Country.Region <- dm25$date <- NULL
dm25 <- dm25 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dm25= cbind(dm25, c(as.Date("2020-03-25") ))
names(dm25)[5] <- "date"

dm26= filter(p2, date == "2020-03-26")
dm26$Country.Region <- dm26$date <- NULL
dm26 <- dm26 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dm26= cbind(dm26, c(as.Date("2020-03-26") ))
names(dm26)[5] <- "date"
```

```r
dm27= filter(p2, date == "2020-03-27")
dm27$Country.Region <- dm27$date <- NULL
dm27 <- dm27 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dm27= cbind(dm27, c(as.Date("2020-03-27") ))
names(dm27)[5] <- "date"

dm28= filter(p2, date == "2020-03-28")
dm28$Country.Region <- dm28$date <- NULL
dm28 <- dm28 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dm28= cbind(dm28, c(as.Date("2020-03-28") ))
names(dm28)[5] <- "date"

dm29= filter(p2, date == "2020-03-29")
dm29$Country.Region <- dm29$date <- NULL
dm29 <- dm29 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dm29= cbind(dm29, c(as.Date("2020-03-29") ))
names(dm29)[5] <- "date"

dm30= filter(p2, date == "2020-03-30")
dm30$Country.Region <- dm30$date <- NULL
dm30 <- dm30 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dm30= cbind(dm30, c(as.Date("2020-03-30") ))
names(dm30)[5] <- "date"

dm31= filter(p2, date == "2020-03-31")
dm31$Country.Region <- dm31$date <- NULL
dm31 <- dm31 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dm31= cbind(dm31, c(as.Date("2020-03-31") ))
names(dm31)[5] <- "date"
```

April

```r
 da1= filter(p2, date == "2020-04-01")
da1$Country.Region <- da1$date <- NULL
da1 <- da1 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da1= cbind(da1, c(as.Date("2020-04-01") ))
names(da1)[5] <- "date"

da2= filter(p2, date == "2020-04-02")
da2$Country.Region <- da2$date <- NULL
da2 <- da2 %>%
  group_by(Province.State) %>%
```

```
  summarise_all(funs(sum))
da2= cbind(da2, c(as.Date("2020-04-02") ))
names(da2)[5] <- "date"

da3= filter(p2, date == "2020-04-03")
da3$Country.Region <- da3$date <- NULL
da3 <- da3 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da3= cbind(da3, c(as.Date("2020-04-03") ))
names(da3)[5] <- "date"

da4= filter(p2, date == "2020-04-04")
da4$Country.Region <- da4$date <- NULL
da4 <- da4 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da4= cbind(da4, c(as.Date("2020-04-04") ))
names(da4)[5] <- "date"

da5= filter(p2, date == "2020-04-05")
da5$Country.Region <- da5$date <- NULL
da5 <- da5 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da5= cbind(da5, c(as.Date("2020-04-05") ))
names(da5)[5] <- "date"

da6= filter(p2, date == "2020-04-06")
da6$Country.Region <- da6$date <- NULL
da6 <- da6 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da6= cbind(da6, c(as.Date("2020-04-06") ))
names(da6)[5] <- "date"

da7= filter(p2, date == "2020-04-07")
da7$Country.Region <- da7$date <- NULL
da7 <- da7 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da7= cbind(da7, c(as.Date("2020-04-07") ))
names(da7)[5] <- "date"

da8= filter(p2, date == "2020-04-08")
da8$Country.Region <- da8$date <- NULL
da8 <- da8 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da8= cbind(da8, c(as.Date("2020-04-08") ))
names(da8)[5] <- "date"

da9= filter(p2, date == "2020-04-09")
```

```
da9$Country.Region <- da9$date <- NULL
da9 <- da9 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da9= cbind(da9, c(as.Date("2020-04-09") ))
names(da9)[5] <- "date"

da10= filter(p2, date == "2020-04-10")
da10$Country.Region <- da10$date <- NULL
da10 <- da10 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da10= cbind(da10, c(as.Date("2020-04-10") ))
names(da10)[5] <- "date"

da11= filter(p2, date == "2020-04-11")
da11$Country.Region <- da11$date <- NULL
da11 <- da11 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da11= cbind(da11, c(as.Date("2020-04-11") ))
names(da11)[5] <- "date"

da12= filter(p2, date == "2020-04-12")
da12$Country.Region <- da12$date <- NULL
da12 <- da12 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da12= cbind(da12, c(as.Date("2020-04-12") ))
names(da12)[5] <- "date"

da13= filter(p2, date == "2020-04-13")
da13$Country.Region <- da13$date <- NULL
da13 <- da13 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da13= cbind(da13, c(as.Date("2020-04-13") ))
names(da13)[5] <- "date"

da14= filter(p2, date == "2020-04-14")
da14$Country.Region <- da14$date <- NULL
da14 <- da14 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da14= cbind(da14, c(as.Date("2020-04-14") ))
names(da14)[5] <- "date"

da15= filter(p2, date == "2020-04-15")
da15$Country.Region <- da15$date <- NULL
da15 <- da15 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da15= cbind(da15, c(as.Date("2020-04-15") ))
```

```
names(da15)[5] <- "date"

da16= filter(p2, date == "2020-04-16")
da16$Country.Region <- da16$date <- NULL
da16 <- da16 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da16= cbind(da16, c(as.Date("2020-04-16") ))
names(da16)[5] <- "date"

da17= filter(p2, date == "2020-04-17")
da17$Country.Region <- da17$date <- NULL
da17 <- da17 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da17= cbind(da17, c(as.Date("2020-04-17") ))
names(da17)[5] <- "date"

da18= filter(p2, date == "2020-04-18")
da18$Country.Region <- da18$date <- NULL
da18 <- da18 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da18= cbind(da18, c(as.Date("2020-04-18") ))
names(da18)[5] <- "date"

da19= filter(p2, date == "2020-04-19")
da19$Country.Region <- da19$date <- NULL
da19 <- da19 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da19= cbind(da19, c(as.Date("2020-04-19") ))
names(da19)[5] <- "date"

da20= filter(p2, date == "2020-04-20")
da20$Country.Region <- da20$date <- NULL
da20 <- da20 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da20= cbind(da20, c(as.Date("2020-04-20") ))
names(da20)[5] <- "date"

da21= filter(p2, date == "2020-04-21")
da21$Country.Region <- da21$date <- NULL
da21 <- da21 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da21= cbind(da21, c(as.Date("2020-04-21") ))
names(da21)[5] <- "date"

da22= filter(p2, date == "2020-04-22")
da22$Country.Region <- da22$date <- NULL
da22 <- da22 %>%
```

```r
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da22= cbind(da22, c(as.Date("2020-04-22") ))
names(da22)[5] <- "date"

da23= filter(p2, date == "2020-04-23")
da23$Country.Region <- da23$date <- NULL
da23 <- da23 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da23= cbind(da23, c(as.Date("2020-04-23") ))
names(da23)[5] <- "date"

da24= filter(p2, date == "2020-04-24")
da24$Country.Region <- da24$date <- NULL
da24 <- da24 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da24= cbind(da24, c(as.Date("2020-04-24") ))
names(da24)[5] <- "date"

da25= filter(p2, date == "2020-04-25")
da25$Country.Region <- da25$date <- NULL
da25 <- da25 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da25= cbind(da25, c(as.Date("2020-04-25") ))
names(da25)[5] <- "date"

da26= filter(p2, date == "2020-04-26")
da26$Country.Region <- da26$date <- NULL
da26 <- da26 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da26= cbind(da26, c(as.Date("2020-04-26") ))
names(da26)[5] <- "date"

da27= filter(p2, date == "2020-04-27")
da27$Country.Region <- da27$date <- NULL
da27 <- da27 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da27= cbind(da27, c(as.Date("2020-04-27") ))
names(da27)[5] <- "date"

da28= filter(p2, date == "2020-04-28")
da28$Country.Region <- da28$date <- NULL
da28 <- da28 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da28= cbind(da28, c(as.Date("2020-04-28") ))
names(da28)[5] <- "date"
```

```
da29= filter(p2, date == "2020-04-29")
da29$Country.Region <- da29$date <- NULL
da29 <- da29 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da29= cbind(da29, c(as.Date("2020-04-29") ))
names(da29)[5] <- "date"

da30= filter(p2, date == "2020-04-30")
da30$Country.Region <- da30$date <- NULL
da30 <- da30 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
da30= cbind(da30, c(as.Date("2020-04-30") ))
names(da30)[5] <- "date"
```

May

```
dma1= filter(p2, date == "2020-05-01")
dma1$Country.Region <- dma1$date <- NULL
dma1 <- dma1 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma1= cbind(dma1, c(as.Date("2020-05-01") ))
names(dma1)[5] <- "date"

dma2= filter(p2, date == "2020-05-02")
dma2$Country.Region <- dma2$date <- NULL
dma2 <- dma2 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma2= cbind(dma2, c(as.Date("2020-05-02") ))
names(dma2)[5] <- "date"

dma3= filter(p2, date == "2020-05-03")
dma3$Country.Region <- dma3$date <- NULL
dma3 <- dma3 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma3= cbind(dma3, c(as.Date("2020-05-03") ))
names(dma3)[5] <- "date"

dma4= filter(p2, date == "2020-05-04")
dma4$Country.Region <- dma4$date <- NULL
dma4 <- dma4 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma4= cbind(dma4, c(as.Date("2020-05-04") ))
names(dma4)[5] <- "date"

dma5= filter(p2, date == "2020-05-05")
dma5$Country.Region <- dma5$date <- NULL
dma5 <- dma5 %>%
  group_by(Province.State) %>%
```

```r
  summarise_all(funs(sum))
dma5= cbind(dma5, c(as.Date("2020-05-05") ))
names(dma5)[5] <- "date"

dma6= filter(p2, date == "2020-05-06")
dma6$Country.Region <- dma6$date <- NULL
dma6 <- dma6 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma6= cbind(dma6, c(as.Date("2020-05-06") ))
names(dma6)[5] <- "date"

dma7= filter(p2, date == "2020-05-07")
dma7$Country.Region <- dma7$date <- NULL
dma7 <- dma7 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma7= cbind(dma7, c(as.Date("2020-05-07") ))
names(dma7)[5] <- "date"

dma8 = filter(p2, date == "2020-05-08")
dma8$Country.Region <- dma8$date <- NULL
dma8 <- dma8 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma8= cbind(dma8, c(as.Date("2020-05-08") ))
names(dma8)[5] <- "date"

dma9= filter(p2, date == "2020-05-09")
dma9$Country.Region <- dma9$date <- NULL
dma9<- dma9 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma9= cbind(dma9, c(as.Date("2020-05-09") ))
names(dma9)[5] <- "date"

dma10= filter(p2, date == "2020-05-10")
dma10$Country.Region <- dma10$date <- NULL
dma10 <- dma10 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma10= cbind(dma10, c(as.Date("2020-05-10") ))
names(dma10)[5] <- "date"

dma11= filter(p2, date == "2020-05-11")
dma11$Country.Region <- dma11$date <- NULL
dma11 <- dma11 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma11= cbind(dma11, c(as.Date("2020-05-11") ))
names(dma11)[5] <- "date"

dma12= filter(p2, date == "2020-05-12")
```

```r
dma12$Country.Region <- dma12$date <- NULL
dma12 <- dma12 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma12= cbind(dma12, c(as.Date("2020-05-12") ))
names(dma12)[5] <- "date"

dma13= filter(p2, date == "2020-05-13")
dma13$Country.Region <- dma13$date <- NULL
dma13 <- dma13 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma13= cbind(dma13, c(as.Date("2020-05-13") ))
names(dma13)[5] <- "date"

dma14= filter(p2, date == "2020-05-14")
dma14$Country.Region <- dma14$date <- NULL
dma14 <- dma14 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma14= cbind(dma14, c(as.Date("2020-05-14") ))
names(dma14)[5] <- "date"

dma15= filter(p2, date == "2020-05-15")
dma15$Country.Region <- dma15$date <- NULL
dma15 <- dma15 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma15= cbind(dma15, c(as.Date("2020-05-15") ))
names(dma15)[5] <- "date"

dma16= filter(p2, date == "2020-05-16")
dma16$Country.Region <- dma16$date <- NULL
dma16 <- dma16 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma16= cbind(dma16, c(as.Date("2020-05-16") ))
names(dma16)[5] <- "date"

dma17= filter(p2, date == "2020-05-17")
dma17$Country.Region <- dma17$date <- NULL
dma17 <- dma17 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma17= cbind(dma17, c(as.Date("2020-05-17") ))
names(dma17)[5] <- "date"

dma18= filter(p2, date == "2020-05-18")
dma18$Country.Region <- dma18$date <- NULL
dma18 <- dma18 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma18= cbind(dma18, c(as.Date("2020-05-18") ))
```

```r
names(dma18)[5] <- "date"

dma19= filter(p2, date == "2020-05-19")
dma19$Country.Region <- dma19$date <- NULL
dma19 <- dma19 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma19= cbind(dma19, c(as.Date("2020-05-19") ))
names(dma19)[5] <- "date"

dma20= filter(p2, date == "2020-05-20")
dma20$Country.Region <- dma20$date <- NULL
dma20 <- dma20 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
dma20= cbind(dma20, c(as.Date("2020-05-20") ))
names(dma20)[5] <- "date"
```

Concatenate already processed data from March 22 through May 22

```r
ds2= rbind(dm22,dm23,dm24,dm25,dm26,dm27,dm28,dm29,dm30,dm31,da1,da2,da3,da4,da5,da6,da7,da8,da9,da10,d
ds2$Recovered <- NULL
```

Given that data from March 10 to 21 is only aggregated by state it does not need transformation, except for April 11

```r
p1 <- cSplit(p1, "Province.State", ",")
p12 <- p1[c(526:1133),c(1:7)]
p12$Province.State_2 <-p12$Country.Region <- p12$Recovered <- NULL
p11 <- p1[c(1:525),c(1:7)]
p12 <- p12[, c(4,1, 2,3)]
names(p12)[1] <- "Province.State"
```

Here we deal with another inconsistency on state labeling and aggregation

```r
p6 <- cSplit(p11, "Province.State_2", " ")
p6 <- na.omit(p6, cols = c("Province.State_2_1"))
pfrom <- p6[c(1:513),c(8:10)]
pfrom <- pfrom %>%
  mutate_all(funs(ifelse(is.na(.), 0, .)))
pn= cbind(p6,pfrom)
p6 <- pn[,c(-8:-10)]
p6<-p6 %>%
  filter(Province.State_2_3!="1",Province.State_2_2!="1" , Province.State_2_4!="1")
p6$Province.State_2_2 <- p6$Province.State_2_3 <- p6$Province.State_2_4 <- NULL
```

Change from D.C to DC for the mapping library abbr2state

```r
p6 <- p6 %>%
  mutate(Province.State_2_1 = str_replace(Province.State_2_1, "D.C.", "DC"))
```

Change state abbreviations to full names

```r
ab <- p6$Province.State_2_1
p6$Province.State_2_1 <- abbr2state(ab)
p6[c(429),c(7)] <- "District of Columbia"
```

Delete and change order of dates

```r
p6$Province.State_1 <- p6$Country.Region <- p6$Recovered <- NULL
names(p6)[4] <- "Province.State"
p6 <- p6[, c(4,1,2,3)]

d1= filter(p6, date == "2020-03-01")
d1$date <- NULL
d1 <- d1 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
d1= cbind(d1, c(as.Date("2020-03-01") ))
names(d1)[4] <- "date"

d2= filter(p6, date == "2020-03-02")
d2$date <- NULL
d2 <- d2 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
d2= cbind(d2, c(as.Date("2020-03-02") ))
names(d2)[4] <- "date"

d3= filter(p6, date == "2020-03-03")
d3$date <- NULL
d3 <- d3 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
d3= cbind(d3, c(as.Date("2020-03-03") ))
names(d3)[4] <- "date"

d4= filter(p6, date == "2020-03-04")
d4$date <- NULL
d4 <- d4 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
d4= cbind(d4, c(as.Date("2020-03-04") ))
names(d4)[4] <- "date"

d5= filter(p6, date == "2020-03-05")
d5$date <- NULL
d5 <- d5 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
d5= cbind(d5, c(as.Date("2020-03-05") ))
names(d5)[4] <- "date"

d6= filter(p6, date == "2020-03-06")
d6$date <- NULL
d6 <- d6 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
d6= cbind(d6, c(as.Date("2020-03-06") ))
names(d6)[4] <- "date"

d7= filter(p6, date == "2020-03-07")
```

```r
d7$date <- NULL
d7 <- d7 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
d7= cbind(d7, c(as.Date("2020-03-07") ))
names(d7)[4] <- "date"

d8= filter(p6, date == "2020-03-08")
d8$date <- NULL
d8 <- d8 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
d8= cbind(d8, c(as.Date("2020-03-08") ))
names(d8)[4] <- "date"

d9= filter(p6, date == "2020-03-09")
d9$date <- NULL
d9 <- d9 %>%
  group_by(Province.State) %>%
  summarise_all(funs(sum))
d9= cbind(d9, c(as.Date("2020-03-09") ))
names(d9)[4] <- "date"

p11 = rbind(d1,d2,d3,d4,d5,d6,d7,d8,d9)
```

Concatenate all three subgroups of information we made previously

```r
ds2 <- rbind(p11,p12,ds2)
```

Add Latitude and Longitude of full dataset

```r
ds2 <- cbind(ds2,1,1)
names(ds2)[5] <- "Latitude"
names(ds2)[6] <- "Longitude"
```

Generate coordinates per state

```r
p1= filter(coord,Country.Region == "US",date == "2020-03-20" )
```

Assign the state coordinates to its respective datapoint

```r
Alabama= filter(p1,Province.State == "Alabama" )
Alabama1 <- Alabama$Latitude[c(1)]
Alabama2 <- Alabama$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Alabama" ] <-Alabama1
ds2$Longitude[ ds2$Province.State== "Alabama" ] <- Alabama2

Alaska= filter(p1,Province.State == "Alaska" )
Alaska1 <- Alaska$Latitude[c(1)]
Alaska2 <- Alaska$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Alaska" ] <-Alaska1
ds2$Longitude[ ds2$Province.State== "Alaska" ] <- Alaska2

Arizona= filter(p1,Province.State == "Arizona" )
Arizona1 <- Arizona$Latitude[c(1)]
Arizona2 <- Arizona$Longitude[c(1)]
```

```
ds2$Latitude[ ds2$Province.State== "Arizona" ] <-Arizona1
ds2$Longitude[ ds2$Province.State== "Arizona" ] <- Arizona2

Arkansas= filter(p1,Province.State == "Arkansas" )
Arkansas1 <- Arkansas$Latitude[c(1)]
Arkansas2 <- Arkansas$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Arkansas" ] <-Arkansas1
ds2$Longitude[ ds2$Province.State== "Arkansas" ] <- Arkansas2

California= filter(p1,Province.State == "California" )
California1 <- California$Latitude[c(1)]
California2 <- California$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "California" ] <-California1
ds2$Longitude[ ds2$Province.State== "California" ] <- California2

Colorado= filter(p1,Province.State == "Colorado" )
Colorado1 <- Colorado$Latitude[c(1)]
Colorado2 <- Colorado$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Colorado" ] <-Colorado1
ds2$Longitude[ ds2$Province.State== "Colorado" ] <- Colorado2

Connecticut= filter(p1,Province.State == "Connecticut" )
Connecticut1 <- Connecticut$Latitude[c(1)]
Connecticut2 <- Connecticut$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Connecticut" ] <-Connecticut1
ds2$Longitude[ ds2$Province.State== "Connecticut" ] <- Connecticut2

Delaware= filter(p1,Province.State == "Delaware" )
Delaware1 <- Delaware$Latitude[c(1)]
Delaware2 <- Delaware$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Delaware" ] <-Delaware1
ds2$Longitude[ ds2$Province.State== "Delaware" ] <- Delaware2

DC= filter(p1,Province.State == "District of Columbia" )
DC1 <- DC$Latitude[c(1)]
DC2 <- DC$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "District of Columbia" ] <-DC1
ds2$Longitude[ ds2$Province.State== "District of Columbia" ] <- DC2

Florida= filter(p1,Province.State == "Florida" )
Florida1 <- Florida$Latitude[c(1)]
Florida2 <- Florida$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Florida" ] <-Florida1
ds2$Longitude[ ds2$Province.State== "Florida" ] <- Florida2

Georgia= filter(p1,Province.State == "Georgia" )
Georgia1 <- Georgia$Latitude[c(1)]
Georgia2 <- Georgia$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Georgia" ] <-Georgia1
ds2$Longitude[ ds2$Province.State== "Georgia" ] <- Georgia2

Hawaii= filter(p1,Province.State == "Hawaii" )
Hawaii1 <- Hawaii$Latitude[c(1)]
```

```r
Hawaii2 <- Hawaii$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Hawaii" ] <-Hawaii1
ds2$Longitude[ ds2$Province.State== "Hawaii" ] <- Hawaii2

Idaho= filter(p1,Province.State == "Idaho" )
Idaho1 <- Idaho$Latitude[c(1)]
Idaho2 <- Idaho$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Idaho" ] <-Idaho1
ds2$Longitude[ ds2$Province.State== "Idaho" ] <- Idaho2

Illinois= filter(p1,Province.State == "Illinois" )
Illinois1 <- Illinois$Latitude[c(1)]
Illinois2 <- Illinois$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Illinois" ] <-Illinois1
ds2$Longitude[ ds2$Province.State== "Illinois" ] <- Illinois2

Indiana= filter(p1,Province.State == "Indiana" )
Indiana1 <- Indiana$Latitude[c(1)]
Indiana2 <- Indiana$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Indiana" ] <-Indiana1
ds2$Longitude[ ds2$Province.State== "Indiana" ] <- Indiana2

Iowa= filter(p1,Province.State == "Iowa" )
Iowa1 <- Iowa$Latitude[c(1)]
Iowa2 <- Iowa$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Iowa" ] <-Iowa1
ds2$Longitude[ ds2$Province.State== "Iowa" ] <- Iowa2

Kansas= filter(p1,Province.State == "Kansas" )
Kansas1 <- Kansas$Latitude[c(1)]
Kansas2 <- Kansas$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Kansas" ] <-Kansas1
ds2$Longitude[ ds2$Province.State== "Kansas" ] <- Kansas2

Kentucky= filter(p1,Province.State == "Kentucky" )
Kentucky1 <- Kentucky$Latitude[c(1)]
Kentucky2 <- Kentucky$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Kentucky" ] <-Kentucky1
ds2$Longitude[ ds2$Province.State== "Kentucky" ] <- Kentucky2

Louisiana= filter(p1,Province.State == "Louisiana" )
Louisiana1 <- Louisiana$Latitude[c(1)]
Louisiana2 <- Louisiana$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Louisiana" ] <-Louisiana1
ds2$Longitude[ ds2$Province.State== "Louisiana" ] <- Louisiana2

Maine= filter(p1,Province.State == "Maine" )
Maine1 <- Maine$Latitude[c(1)]
Maine2 <- Maine$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Maine" ] <-Maine1
ds2$Longitude[ ds2$Province.State== "Maine" ] <- Maine2

Maryland= filter(p1,Province.State == "Maryland" )
```

```
Maryland1 <- Maryland$Latitude[c(1)]
Maryland2 <- Maryland$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Maryland" ] <-Maryland1
ds2$Longitude[ ds2$Province.State== "Maryland" ] <- Maryland2

Massachusetts= filter(p1,Province.State == "Massachusetts" )
Massachusetts1 <- Massachusetts$Latitude[c(1)]
Massachusetts2 <- Massachusetts$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Massachusetts" ] <-Massachusetts1
ds2$Longitude[ ds2$Province.State== "Massachusetts" ] <- Massachusetts2

Michigan= filter(p1,Province.State == "Michigan" )
Michigan1 <- Michigan$Latitude[c(1)]
Michigan2 <- Michigan$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Michigan" ] <-Michigan1
ds2$Longitude[ ds2$Province.State== "Michigan" ] <- Michigan2

Minnesota= filter(p1,Province.State == "Minnesota" )
Minnesota1 <- Minnesota$Latitude[c(1)]
Minnesota2 <- Minnesota$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Minnesota" ] <-Minnesota1
ds2$Longitude[ ds2$Province.State== "Minnesota" ] <- Minnesota2

Mississippi= filter(p1,Province.State == "Mississippi" )
Mississippi1 <- Mississippi$Latitude[c(1)]
Mississippi2 <- Mississippi$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Mississippi" ] <-Mississippi1
ds2$Longitude[ ds2$Province.State== "Mississippi" ] <- Mississippi2

Missouri= filter(p1,Province.State == "Missouri" )
Missouri1 <- Missouri$Latitude[c(1)]
Missouri2 <- Missouri$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Missouri" ] <-Missouri1
ds2$Longitude[ ds2$Province.State== "Missouri" ] <- Missouri2

Montana= filter(p1,Province.State == "Montana" )
Montana1 <- Montana$Latitude[c(1)]
Montana2 <- Montana$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Montana" ] <-Montana1
ds2$Longitude[ ds2$Province.State== "Montana" ] <- Montana2

Nebraska= filter(p1,Province.State == "Nebraska" )
Nebraska1 <- Nebraska$Latitude[c(1)]
Nebraska2 <- Nebraska$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Nebraska" ] <-Nebraska1
ds2$Longitude[ ds2$Province.State== "Nebraska" ] <- Nebraska2

Nevada= filter(p1,Province.State == "Nevada" )
Nevada1 <- Nevada$Latitude[c(1)]
Nevada2 <- Nevada$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Nevada" ] <-Nevada1
ds2$Longitude[ ds2$Province.State== "Nevada" ] <- Nevada2
```

```
NewHa = filter(p1,Province.State == "New Hampshire" )
NewHa1 <- NewHa$Latitude[c(1)]
NewHa2 <- NewHa$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "New Hampshire" ] <-NewHa1
ds2$Longitude[ ds2$Province.State== "New Hampshire" ] <- NewHa2

ny = filter(p1,Province.State == "New York" )
ny1 <- ny$Latitude[c(1)]
ny2 <- ny$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "New York" ] <- ny1
ds2$Longitude[ ds2$Province.State== "New York" ] <- ny2

njer= filter(p1,Province.State == "New Jersey" )
njer1 <- njer$Latitude[c(1)]
njer2 <- njer$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "New Jersey" ] <- njer1
ds2$Longitude[ ds2$Province.State== "New Jersey" ] <- njer2

nmex= filter(p1,Province.State == "New Mexico" )
nmex1 <- nmex$Latitude[c(1)]
nmex2 <- nmex$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "New Mexico" ] <- nmex1
ds2$Longitude[ ds2$Province.State== "New Mexico" ] <- nmex2

ncar= filter(p1,Province.State == "North Carolina" )
ncar1 <- ncar$Latitude[c(1)]
ncar2 <- ncar$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "North Carolina" ] <- ncar1
ds2$Longitude[ ds2$Province.State== "North Carolina" ] <- ncar2

nda= filter(p1,Province.State == "North Dakota" )
nda1 <- nda$Latitude[c(1)]
nda2 <- nda$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "North Dakota" ] <- nda1
ds2$Longitude[ ds2$Province.State== "North Dakota" ] <- nda2

Ohio= filter(p1,Province.State == "Ohio" )
Ohio1 <- Ohio$Latitude[c(1)]
Ohio2 <- Ohio$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Ohio" ] <- Ohio1
ds2$Longitude[ ds2$Province.State== "Ohio" ] <- Ohio2

Oklahoma= filter(p1,Province.State == "Oklahoma" )
Oklahoma1 <- Oklahoma$Latitude[c(1)]
Oklahoma2 <- Oklahoma$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Oklahoma" ] <-Oklahoma1
ds2$Longitude[ ds2$Province.State== "Oklahoma" ] <- Oklahoma2

Oregon= filter(p1,Province.State == "Oregon" )
Oregon1 <- Oregon$Latitude[c(1)]
Oregon2 <- Oregon$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Oregon" ] <-Oregon1
ds2$Longitude[ ds2$Province.State== "Oregon" ] <- Oregon2
```

```
Pennsylvania= filter(p1,Province.State == "Pennsylvania" )
Pennsylvania1 <- Pennsylvania$Latitude[c(1)]
Pennsylvania2 <- Pennsylvania$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Pennsylvania" ] <-Pennsylvania1
ds2$Longitude[ ds2$Province.State== "Pennsylvania" ] <- Pennsylvania2

RhodeI= filter(p1,Province.State == "Rhode Island" )
RhodeI1 <- RhodeI$Latitude[c(1)]
RhodeI2 <- RhodeI$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Rhode Island" ] <-RhodeI1
ds2$Longitude[ ds2$Province.State== "Rhode Island" ] <- RhodeI2

SouthCa= filter(p1,Province.State == "South Carolina" )
SouthCa1 <- SouthCa$Latitude[c(1)]
SouthCa2 <- SouthCa$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "South Carolina" ] <-SouthCa1
ds2$Longitude[ ds2$Province.State== "South Carolina" ] <- SouthCa2

SouthDa= filter(p1,Province.State == "South Dakota" )
SouthDa1 <- SouthDa$Latitude[c(1)]
SouthDa2 <- SouthDa$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "South Dakota" ] <-SouthDa1
ds2$Longitude[ ds2$Province.State== "South Dakota" ] <- SouthDa2

Tennessee= filter(p1,Province.State == "Tennessee" )
Tennessee1 <- Tennessee$Latitude[c(1)]
Tennessee2 <- Tennessee$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Tennessee" ] <-Tennessee1
ds2$Longitude[ ds2$Province.State== "Tennessee" ] <- Tennessee2

Texas= filter(p1,Province.State == "Texas" )
Texas1 <- Texas$Latitude[c(1)]
Texas2 <- Texas$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Texas" ] <-Texas1
ds2$Longitude[ ds2$Province.State== "Texas" ] <- Texas2

Utah= filter(p1,Province.State == "Utah" )
Utah1 <- Utah$Latitude[c(1)]
Utah2 <- Utah$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Utah" ] <-Utah1
ds2$Longitude[ ds2$Province.State== "Utah" ] <- Utah2

Vermont= filter(p1,Province.State == "Vermont" )
Vermont1 <- Vermont$Latitude[c(1)]
Vermont2 <- Vermont$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Vermont" ] <-Vermont1
ds2$Longitude[ ds2$Province.State== "Vermont" ] <- Vermont2

Virginia= filter(p1,Province.State == "Virginia" )
Virginia1 <- Virginia$Latitude[c(1)]
Virginia2 <- Virginia$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Virginia" ] <-Virginia1
ds2$Longitude[ ds2$Province.State== "Virginia" ] <- Virginia2
```

```r
Washington= filter(p1,Province.State == "Washington" )
Washington1 <- Washington$Latitude[c(1)]
Washington2 <- Washington$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Washington" ] <-Washington1
ds2$Longitude[ ds2$Province.State== "Washington" ] <- Washington2

WVirginia= filter(p1,Province.State == "West Virginia" )
WVirginia1 <- WVirginia$Latitude[c(1)]
WVirginia2 <- WVirginia$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "West Virginia" ] <-WVirginia1
ds2$Longitude[ ds2$Province.State== "West Virginia" ] <- WVirginia2

Wisconsin= filter(p1,Province.State == "Wisconsin" )
Wisconsin1 <- Wisconsin$Latitude[c(1)]
Wisconsin2 <- Wisconsin$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Wisconsin" ] <-Wisconsin1
ds2$Longitude[ ds2$Province.State== "Wisconsin" ] <- Wisconsin2

Wyoming= filter(p1,Province.State == "Wyoming" )
Wyoming1 <- Wyoming$Latitude[c(1)]
Wyoming2 <- Wyoming$Longitude[c(1)]
ds2$Latitude[ ds2$Province.State== "Wyoming" ] <-Wyoming1
ds2$Longitude[ ds2$Province.State== "Wyoming" ] <- Wyoming2
```

Change date format

```r
ds2$date <- format(as.Date(ds2$date, format="%Y/%m/%d"),"%d-%m-%y")
ds <- ds2
```

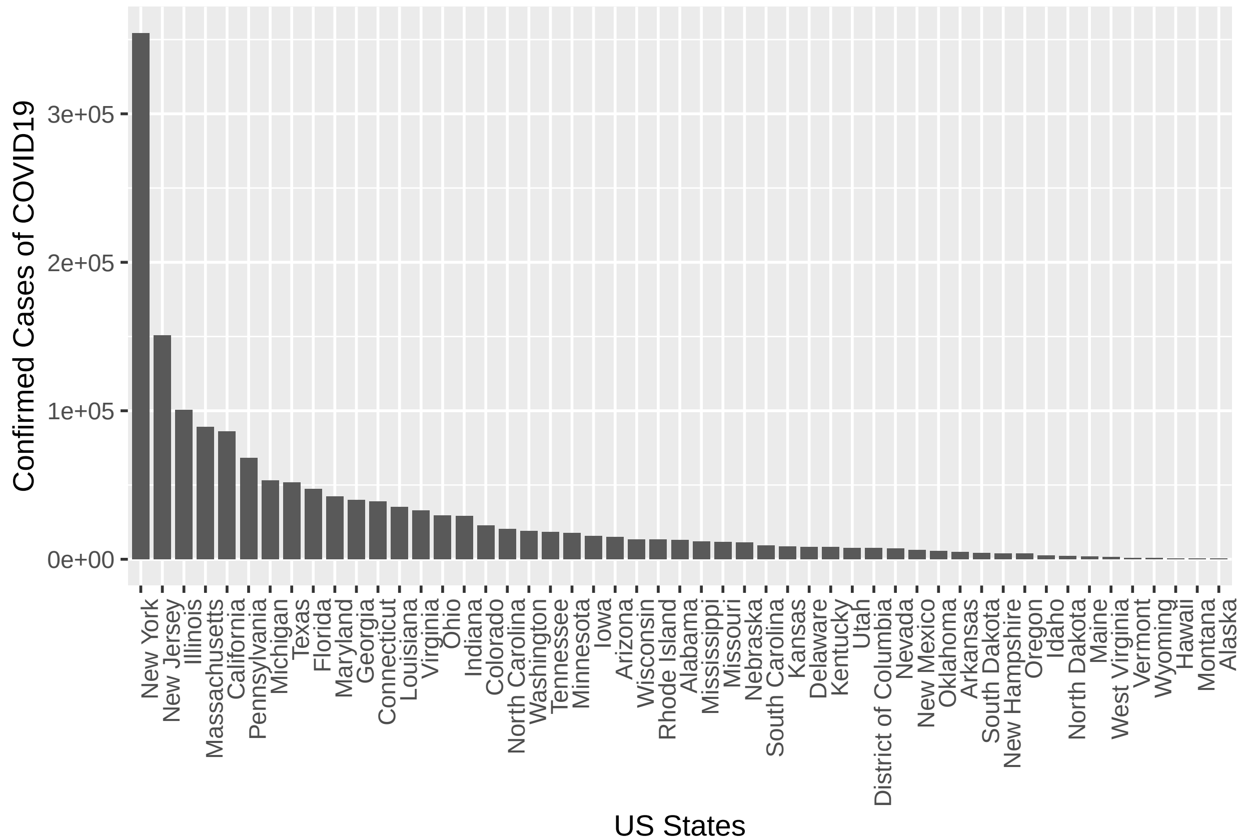
Filter last day

```r
LastDay <- filter(ds, date == "20-05-20")
```

Visualization

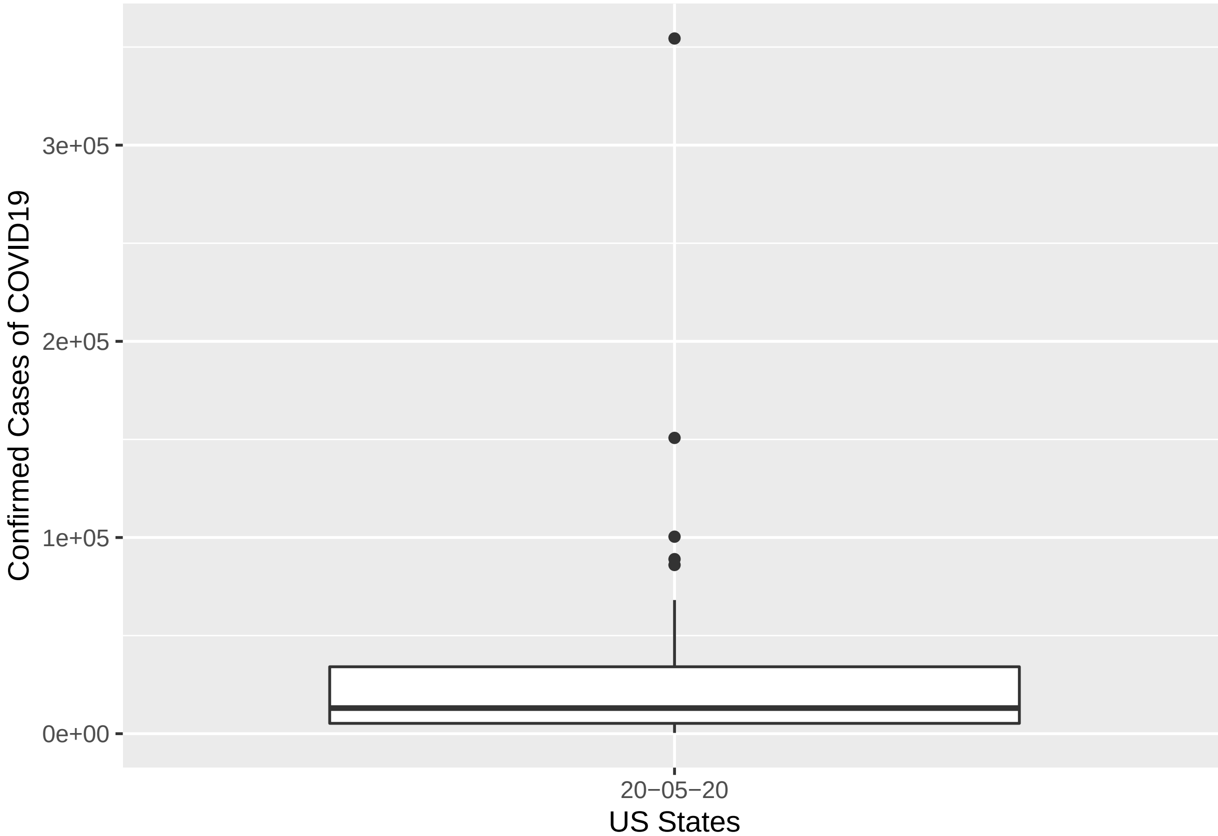
Graph Confirmed vs State

```r
plot1 <- ggplot(LastDay, aes(x = reorder(Province.State, -Confirmed), y = Confirmed)) +
  scale_x_discrete("US States") +
  scale_y_continuous("Confirmed Cases of COVID19")+
  geom_bar(stat = "identity",width=.8)
aa <- plot1 + theme(axis.text.x=element_text(angle=90, hjust=1))
print(aa)
```



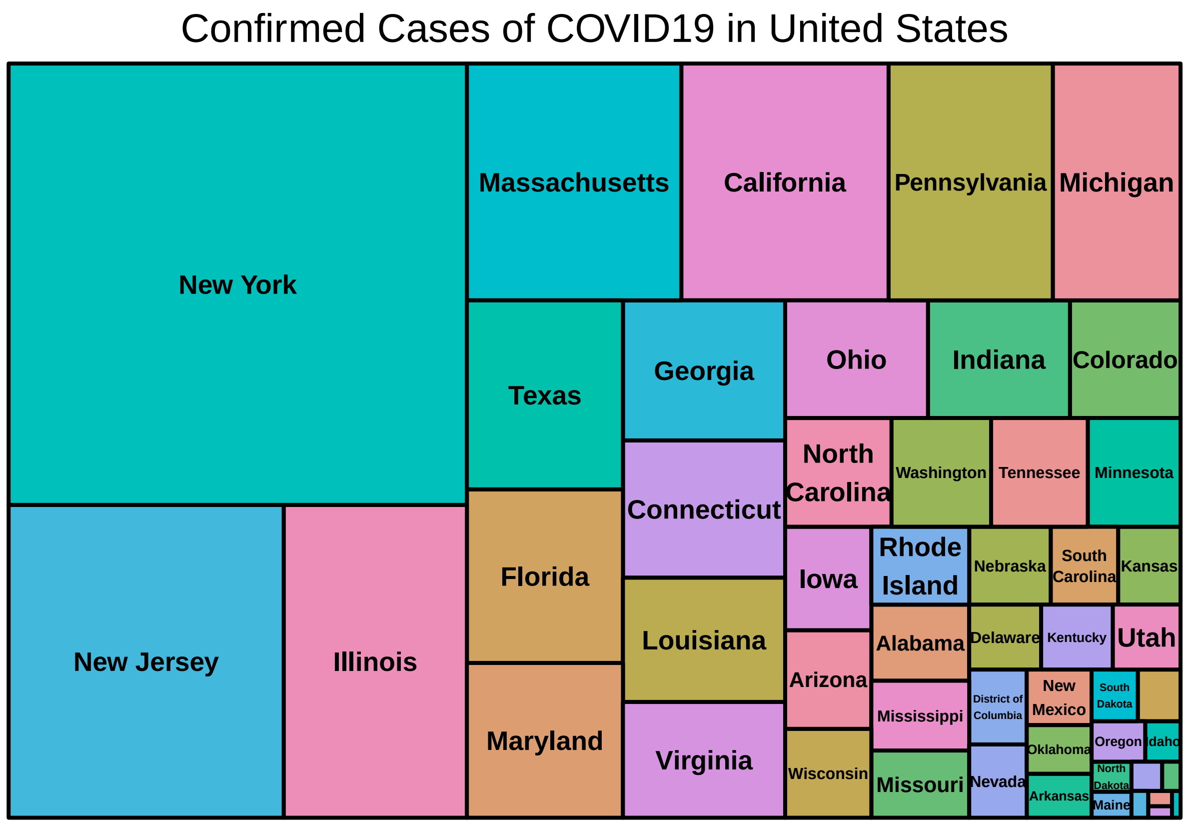
Box Plot

```
p10 <- ggplot(LastDay, aes(x = date, y = Confirmed)) +
  scale_y_continuous("Confirmed Cases of COVID19") +
  scale_x_discrete("US States") +
  geom_boxplot()
print(p10)
```



Tree Map

```
tree <- treemap(LastDay,index = c("Province.State"),vSize ="Confirmed",title = "Confirmed Cases of COVI
```



Add other column in the Last day dataset with the percentage of confirmed cases by state

```
total <- sum(LastDay$Confirmed)
percentage <- (LastDay$Confirmed/total)*100
LastDay= cbind(LastDay,percentage)

summary(LastDay)
```

```
##        Province.State   Confirmed          Deaths          date
##  Arizona       : 1     Min.   :   401   Min.   :   10   Length:51
##  California    : 1     1st Qu.:  5268   1st Qu.:  141   Class :character
##  Illinois      : 1     Median : 13052   Median :  481   Mode  :character
##  Massachusetts: 1     Mean   : 30364   Mean   : 1829
##  Oregon        : 1     3rd Qu.: 34112   3rd Qu.: 1739
##  Rhode Island  : 1     Max.   :354370   Max.   :28636
##  (Other)       :45
##     Latitude        Longitude          percentage
##  Min.   :21.09   Min.   :-157.50   Min.   : 0.0259
##  1st Qu.:35.69   1st Qu.:-102.55   1st Qu.: 0.3401
##  Median :39.85   Median : -89.62   Median : 0.8428
##  Mean   :39.46   Mean   : -93.34   Mean   : 1.9608
##  3rd Qu.:43.04   3rd Qu.: -78.99   3rd Qu.: 2.2028
##  Max.   :61.37   Max.   : -69.38   Max.   :22.8834
##
```

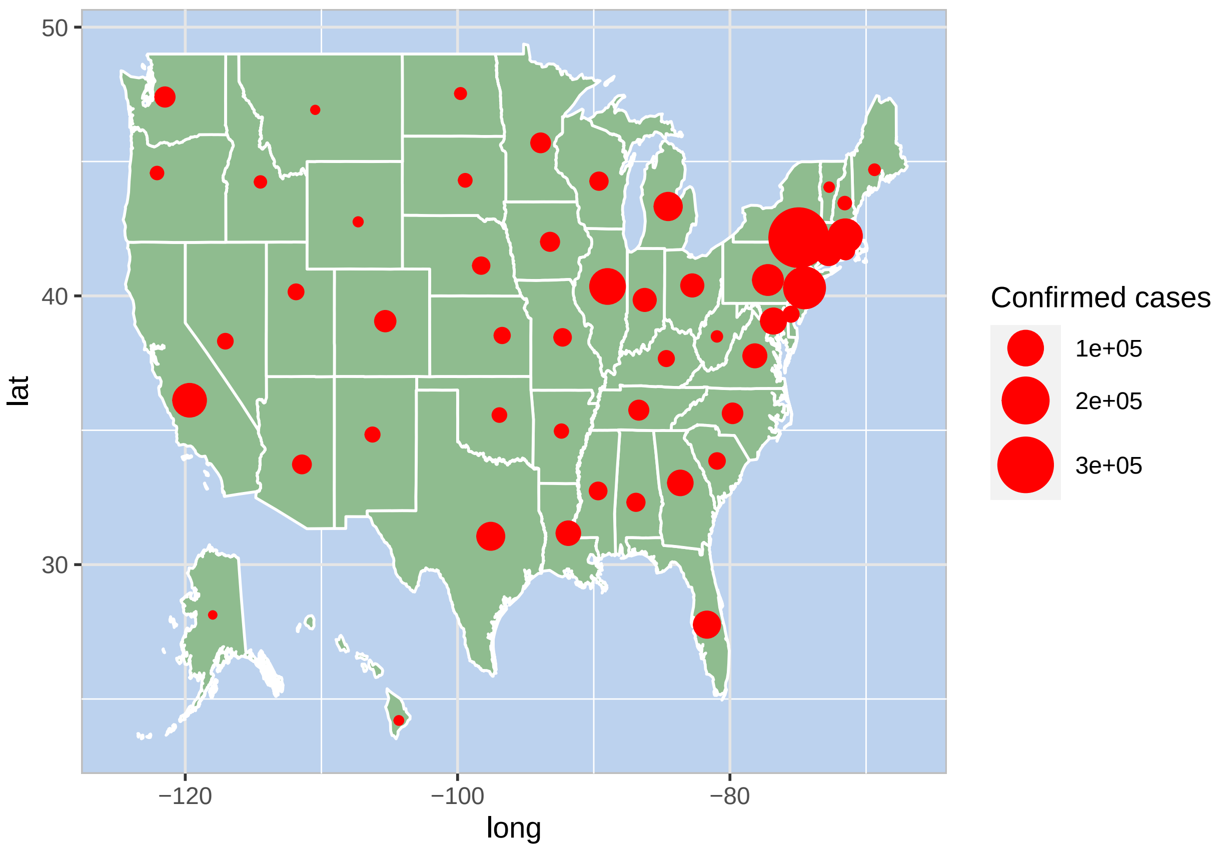
Change Alaska and Hawaii coordinates for proper visualization

```
LastDay$Latitude[ LastDay$Province.State== "Alaska" ] <-28.12768
LastDay$Longitude[ LastDay$Province.State== "Alaska" ] <- -117.981766
```

```r
LastDay$Latitude[ LastDay$Province.State== "Hawaii" ] <-24.200987
LastDay$Longitude[ LastDay$Province.State== "Hawaii" ] <- -104.313994
```

Map of Confirmed Cases

```r
ggplot() + geom_polygon( data=fifty_states, aes(x=long, y=lat, group = group),color="white", fill="dark
  geom_point(data=LastDay,color = "red", aes(x=Longitude, y=Latitude, size =Confirmed))+
  geom_path(colour = "grey40") +
  theme(panel.background = element_rect(fill = "lightsteelblue2", colour = "grey"),
        panel.grid.major = element_line(colour = "grey90")) +
  scale_size(name="", range = c(1, 10)) +
  guides(size=guide_legend("Confirmed cases"))
```



Models

New York

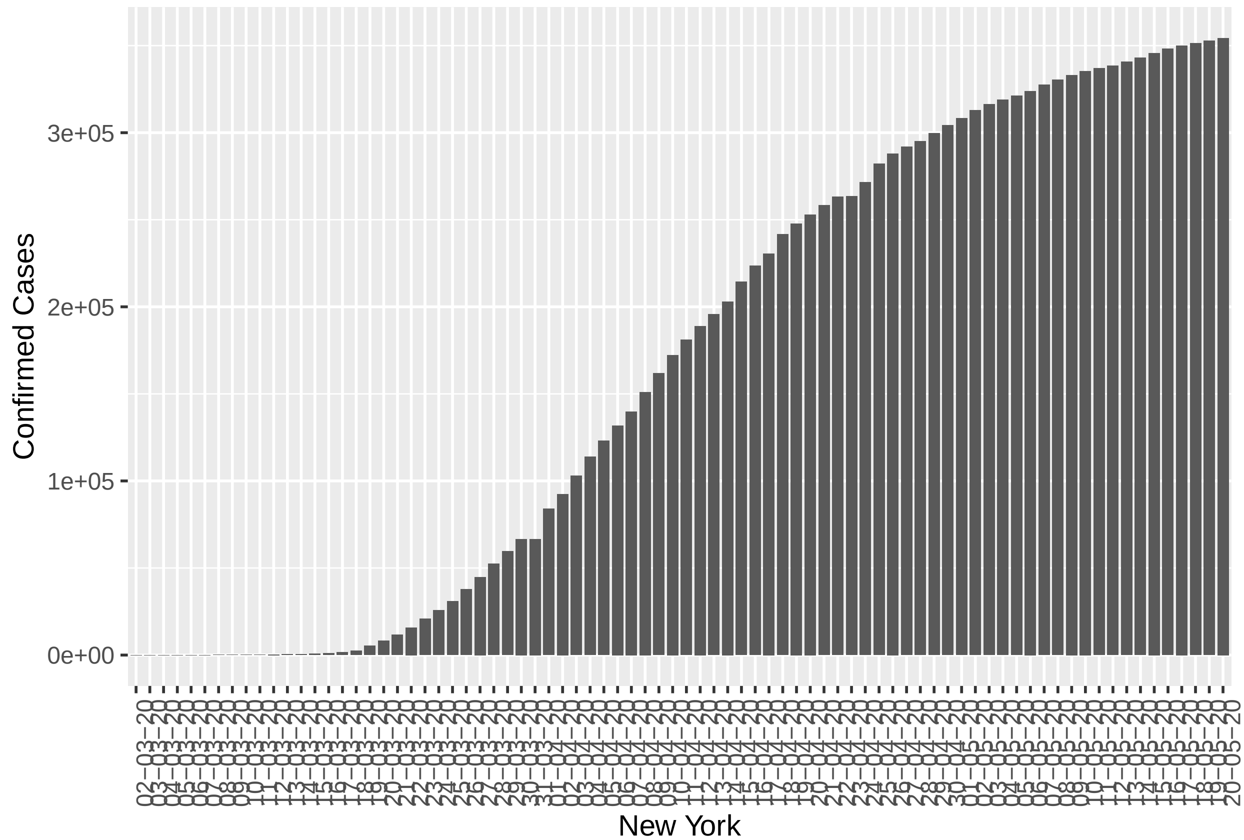
Dataset of New York

```r
NYds <- filter(ds, Province.State == "New York")
```

New York confirmed cases plot

```r
plotNY <- ggplot(NYds, aes(x = reorder(date, Confirmed), y = Confirmed)) +
  scale_x_discrete("New York") +
  scale_y_continuous("Confirmed Cases") +
  geom_bar(stat = "identity",width=.8)
```

```r
plotNY<- plotNY + theme(axis.text.x=element_text(angle=90, hjust=1))
print(plotNY)
```



Create NY subdataset to change date of incidence to number of incidence

```r
NYds <- cbind(NYds,(1:nrow(NYds)))
names(NYds)[7] <- "Day"
x <- NYds$Day
y <- NYds$Confirmed
```

Create the dataset of train set and test set

```r
TrainP=nrow(NYds)*.75
TestP =nrow(NYds)*.25
train = NYds[c(1:TrainP),c(1:7)]
test = NYds[-c(1:TrainP),c(1:7)]
```

NY linear Regression

```r
model1 <- lm(Confirmed ~ poly(Day, 1, raw = TRUE), data = train)
predictions1 <- model1 %>% predict(test)
good1 = fitted.values(model1)
graph1 = c(good1,predictions1)
tra = test$Confirmed
rs <- as.integer(predictions1)
summary(model1)
```

```
##
## Call:
```

```
## lm(formula = Confirmed ~ poly(Day, 1, raw = TRUE), data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -42399 -27802   5944  19235  67329
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -73410       7610  -9.647 1.16e-13 ***
## poly(Day, 1, raw = TRUE)     6082        217  28.035  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 29100 on 58 degrees of freedom
## Multiple R-squared:  0.9313, Adjusted R-squared:  0.9301
## F-statistic: 785.9 on 1 and 58 DF,  p-value: < 2.2e-16
```

Evaluation

```
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model1, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)
sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.9884289809915"
```

```
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 31558.8627634665"
```

```
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.094343188895163"
```

Degree 2 polynomial

```
model2 <- lm(Confirmed ~ poly(Day, 2, raw = TRUE), data = train)
predictions2 <- model2 %>% predict(test)
good2 = fitted.values(model2)
graph2 = c(good2,predictions2)
tra = test$Confirmed
rs <- as.integer(predictions2)

summary(model2)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 2, raw = TRUE), data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -37846 -14427    733  14714  22612
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -17394.831   6511.311  -2.671  0.00983 **
```

```
## poly(Day, 2, raw = TRUE)1    661.586    492.527   1.343  0.18452
## poly(Day, 2, raw = TRUE)2     88.866      7.826  11.355 2.91e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16250 on 57 degrees of freedom
## Multiple R-squared:  0.9789, Adjusted R-squared:  0.9782
## F-statistic:  1324 on 2 and 57 DF,  p-value: < 2.2e-16
```

Model evaluation

```
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model2, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.980467355179391"
```

```
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 161020.645785103"
```

```
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.481360856225493"
```

Degree 3 polynomial

```
model3 <- lm(Confirmed ~ poly(Day, 3, raw = TRUE), data = train)
predictions3 <- model3 %>% predict(test)
good3 = fitted.values(model3)
graph3 = c(good3,predictions3)
tra = test$Confirmed
rs <- as.integer(predictions3)
summary(model3)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 3, raw = TRUE), data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18116.9  -4895.7    -90.7   5032.4  10836.9
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               25068.6237  3609.4350   6.945 4.26e-09 ***
## poly(Day, 3, raw = TRUE)1 -7362.1760   508.2139 -14.486  < 2e-16 ***
## poly(Day, 3, raw = TRUE)2   415.0063    19.2734  21.533  < 2e-16 ***
## poly(Day, 3, raw = TRUE)3    -3.5644     0.2078 -17.153  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6557 on 56 degrees of freedom
## Multiple R-squared:  0.9966, Adjusted R-squared:  0.9965
```

```
## F-statistic:  5523 on 3 and 56 DF,  p-value: < 2.2e-16
```

Evaluation

```
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model3, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.517413565785085"
```

```
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 40143.0315571798"
```

```
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.120005008970339"
```

Degree 4 polynomial

```
model4 <- lm(Confirmed ~ poly(Day, 4, raw = TRUE), data = train)
predictions4 <- model4 %>% predict(test)
good4 = fitted.values(model4)
graph4 = c(good4,predictions4)
tra = test$Confirmed
rs <- as.integer(predictions4)
summary(model4)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 4, raw = TRUE), data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11605.4  -2971.3     89.6   3542.4   9229.1
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                7.879e+03  3.048e+03   2.585  0.01241 *
## poly(Day, 4, raw = TRUE)1 -2.116e+03  6.801e+02  -3.111  0.00296 **
## poly(Day, 4, raw = TRUE)2  3.538e+01  4.481e+01   0.790  0.43317
## poly(Day, 4, raw = TRUE)3  6.062e+00  1.099e+00   5.514 9.67e-07 ***
## poly(Day, 4, raw = TRUE)4 -7.891e-02  8.944e-03  -8.823 4.06e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4258 on 55 degrees of freedom
## Multiple R-squared:  0.9986, Adjusted R-squared:  0.9985
## F-statistic:  9845 on 4 and 55 DF,  p-value: < 2.2e-16
```

Evaluation

```
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model4, test)-test$Confirmed)^2)/
```

```
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.858348259048219"
```

```
sprintf("RSE value is : %s", RSE)
```

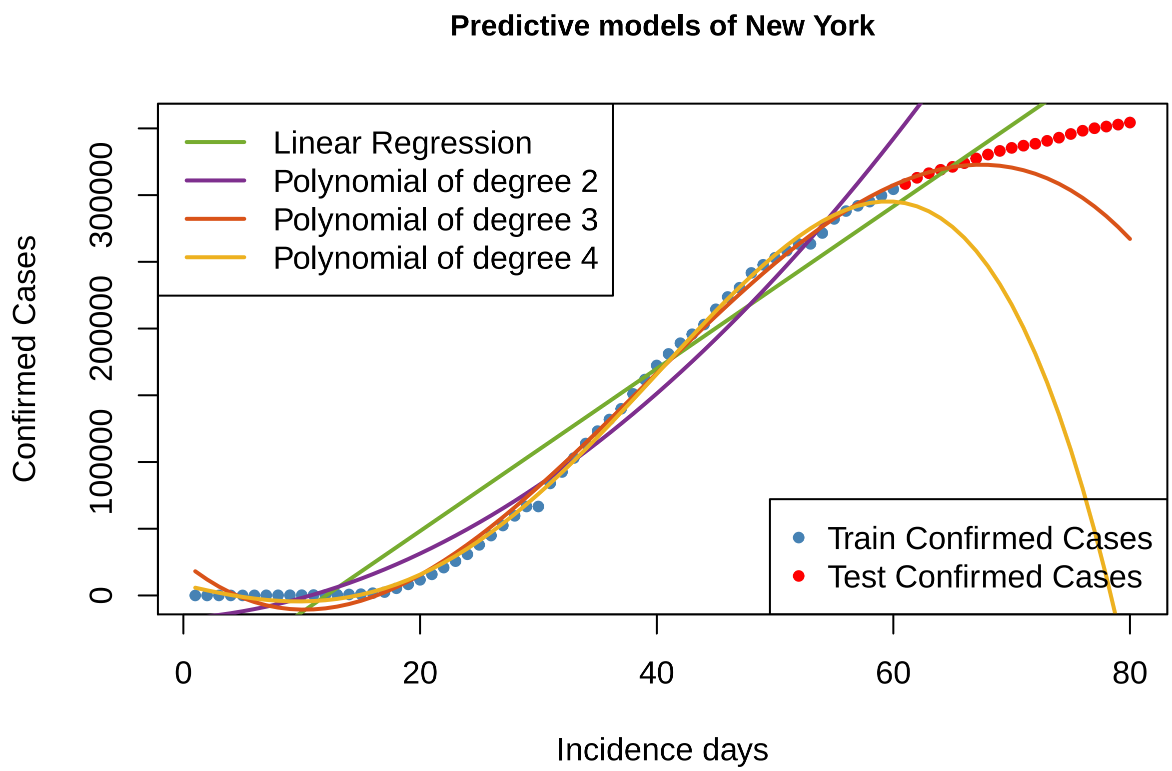
```
## [1] "RSE value is : 212218.743232924"
```

```
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.634414273099066"
```

Multimodel plot

```
plot(x,y,col  = "steelblue",
     pch = 20,xlab = "Incidence days",
     ylab = "Confirmed Cases",
     main="Predictive models of New York",
     cex.main = 0.9)
points(test$Day,test$Confirmed, col="red", pch = 20 )
lines(x,graph1, col='#77AC30', lwd =2)
lines(x,graph2, col='#7E2F8E', lwd =2)
lines(x,graph3, col='#D95319', lwd =2)
lines(x,graph4, col='#EDB120', lwd =2)
legend("topleft", legend=c("Linear Regression", "Polynomial of degree 2","Polynomial of degree 3","Polyn
legend("bottomright",legend=c("Train Confirmed Cases","Test Confirmed Cases"), col=c('steelblue','red')
```



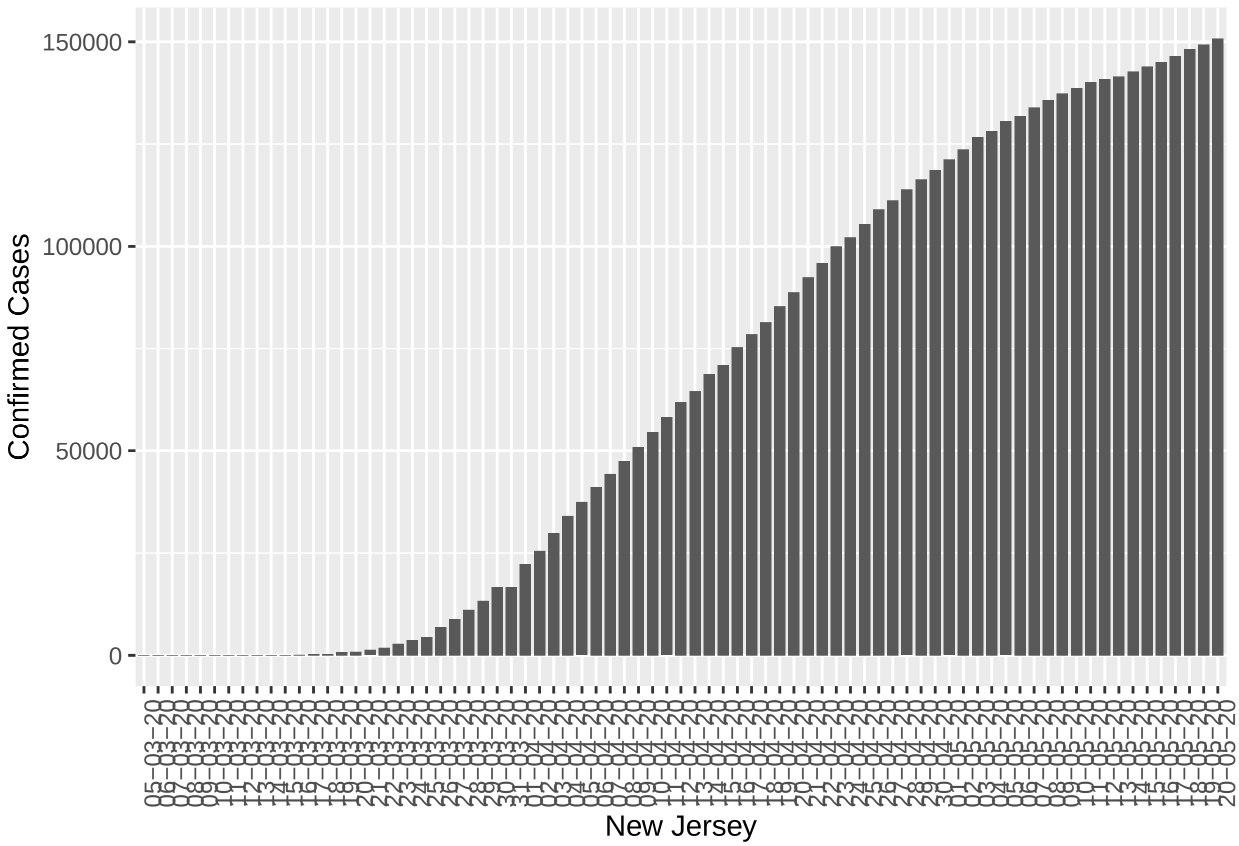
New jersey

Dataset of New Jersey

```
NJds <- filter(ds, Province.State == "New Jersey")
```

New York confirmed cases plot

```
plotNJ <- ggplot(NJds, aes(x = reorder(date, Confirmed), y = Confirmed)) +
  scale_x_discrete("New Jersey") +
  scale_y_continuous("Confirmed Cases") +
  geom_bar(stat = "identity",width=.8)
plotNJ<- plotNJ + theme(axis.text.x=element_text(angle=90, hjust=1))
print(plotNJ)
```



Create New jersey subdataset to change date of incidence to number of incidence

```
NJds <- cbind(NJds,(1:nrow(NJds)))
names(NJds)[7] <- "Day"
```

Create the dataset of train set and test set

```
TrainP=nrow(NJds)*.75
TestP =nrow(NJds)*.25
train = NJds[c(1:TrainP),c(1:7)]
test = NJds[-c(1:TrainP),c(1:7)]
x <- NJds$Day
y <- NJds$Confirmed
```

Linear Regression

```r
model1 <- lm(Confirmed ~ poly(Day, 1, raw = TRUE), data = train)
predictions1 <- model1 %>% predict(test)
good1 = fitted.values(model1)
graph1 = c(good1,predictions1)
tra = test$Confirmed
rs <- as.integer(predictions1)
summary(model1)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 1, raw = TRUE), data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18561.2 -10121.1    209.1  10000.9  26552.5
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -28925.37    3253.27  -8.891 3.16e-12 ***
## poly(Day, 1, raw = TRUE)   2374.91      97.57  24.340  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 12120 on 55 degrees of freedom
## Multiple R-squared:  0.915,  Adjusted R-squared:  0.9135
## F-statistic: 592.4 on 1 and 55 DF,  p-value: < 2.2e-16
```

Evaluation

```r
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model1, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.979922612347053"
```

```r
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 8959.14625328179"
```

```r
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.064975001120721"
```

Degree 2 polynomial

```r
model2 <- lm(Confirmed ~ poly(Day, 2, raw = TRUE), data = train)
predictions2 <- model2 %>% predict(test)
good2 = fitted.values(model2)
graph2 = c(good2,predictions2)
tra = test$Confirmed
rs <- as.integer(predictions2)
summary(model2)
```

```
##
```

```
## Call:
## lm(formula = Confirmed ~ poly(Day, 2, raw = TRUE), data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -11237  -4236   1606   3293   6163
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -2877.691   1872.518  -1.537   0.1302
## poly(Day, 2, raw = TRUE)1  -274.005    148.959  -1.839   0.0713 .
## poly(Day, 2, raw = TRUE)2    45.671      2.489  18.346   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4548 on 54 degrees of freedom
## Multiple R-squared:  0.9883, Adjusted R-squared:  0.9878
## F-statistic:  2272 on 2 and 54 DF,  p-value: < 2.2e-16
```

Evaluation

```
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model2, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)


sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.968985510655857"
```

```
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 59601.290621913"
```

```
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.432250330050887"
```

Degree 3 polynomial

```
model3 <- lm(Confirmed ~ poly(Day, 3, raw = TRUE), data = train)
predictions3 <- model3 %>% predict(test)
good3 = fitted.values(model3)
graph3 = c(good3,predictions3)
tra = test$Confirmed
rs <- as.integer(predictions3)
summary(model3)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 3, raw = TRUE), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5992.8 -1486.6   485.8  1368.4  3739.6
##
```

```
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)    
## (Intercept)              8.357e+03  1.313e+03   6.367   4.7e-08 ***
## poly(Day, 3, raw = TRUE)1 -2.502e+03  1.943e+02 -12.878  < 2e-16 ***
## poly(Day, 3, raw = TRUE)2  1.409e+02  7.748e+00  18.183  < 2e-16 ***
## poly(Day, 3, raw = TRUE)3 -1.094e+00  8.786e-02 -12.456  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2316 on 53 degrees of freedom
## Multiple R-squared:  0.997,  Adjusted R-squared:  0.9968 
## F-statistic:  5890 on 3 and 53 DF,  p-value: < 2.2e-16
```

Evaluation

```r
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model3, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.969003222144401"
```

```r
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 4754.01501628347"
```

```r
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.0344778533889648"
```

Degree 4 polynomial

```r
model4 <- lm(Confirmed ~ poly(Day, 4, raw = TRUE), data = train)
predictions4 <- model4 %>% predict(test)
good4 = fitted.values(model4)
graph4 = c(good4,predictions4)
tra = test$Confirmed
rs <- as.integer(predictions4)
summary(model4)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 4, raw = TRUE), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -3594.1 -1296.1  -125.7  1432.4  2681.8 
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)    
## (Intercept)              2.580e+03  1.193e+03   2.163   0.0352 *  
## poly(Day, 4, raw = TRUE)1 -6.542e+02  2.797e+02  -2.340   0.0232 *  
## poly(Day, 4, raw = TRUE)2  3.998e-01  1.937e+01   0.021   0.9836    
## poly(Day, 4, raw = TRUE)3  2.651e+00  4.997e-01   5.306 2.34e-06 ***
## poly(Day, 4, raw = TRUE)4 -3.229e-02  4.275e-03  -7.553 6.51e-10 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1615 on 52 degrees of freedom
## Multiple R-squared:  0.9986, Adjusted R-squared:  0.9985
## F-statistic:  9103 on 4 and 52 DF,  p-value: < 2.2e-16
```

Evaluation

```r
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model4, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.766463563700959"
```

```r
sprintf("RSE value is : %s", RSE)
```

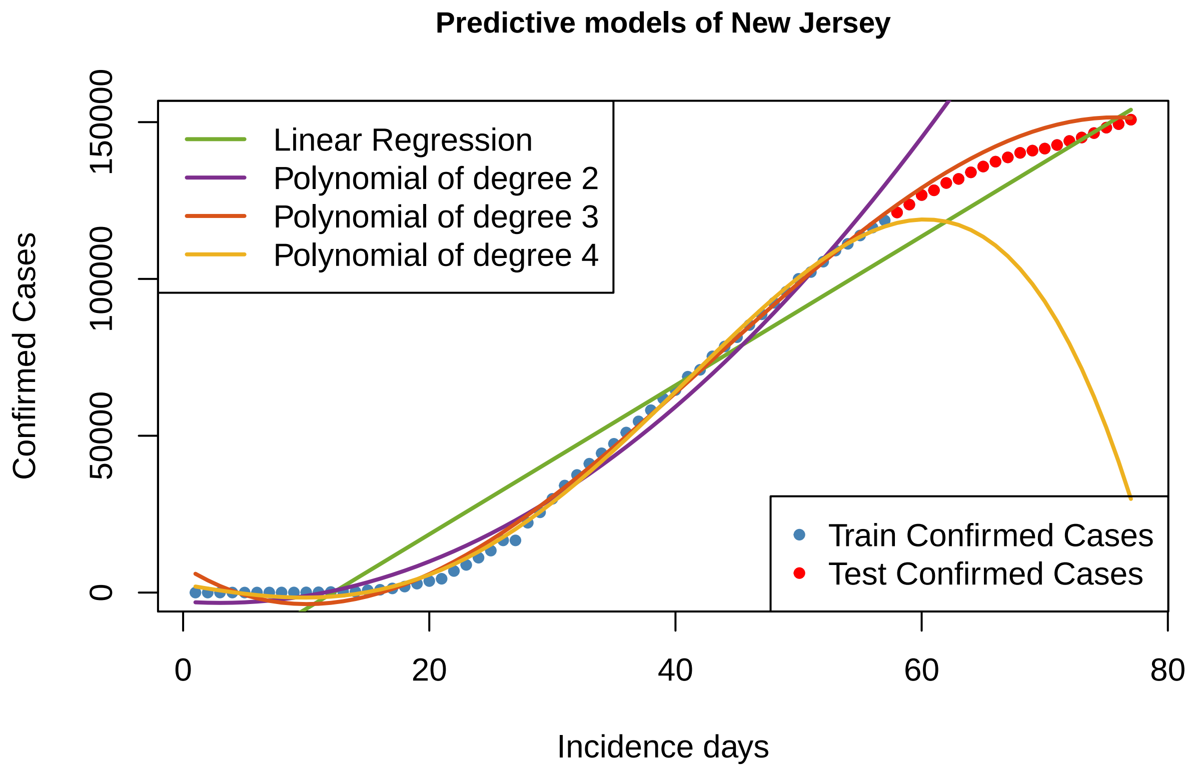
```
## [1] "RSE value is : 59558.9258435142"
```

```r
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.431943085203428"
```

Multimodel plot

```r
plot(x,y,col  = "steelblue",
     pch = 20,xlab = "Incidence days",
     ylab = "Confirmed Cases",
     main="Predictive models of New Jersey",
     cex.main = 0.9)
points(test$Day,test$Confirmed, col="red", pch = 20 )
lines(x,graph1, col='#77AC30', lwd =2)
lines(x,graph2, col='#7E2F8E', lwd =2)
lines(x,graph3, col='#D95319', lwd =2)
lines(x,graph4, col='#EDB120', lwd =2)
legend("topleft", legend=c("Linear Regression", "Polynomial of degree 2","Polynomial of degree 3","Poly
legend("bottomright",legend=c("Train Confirmed Cases","Test Confirmed Cases"), col=c('steelblue','red')
```



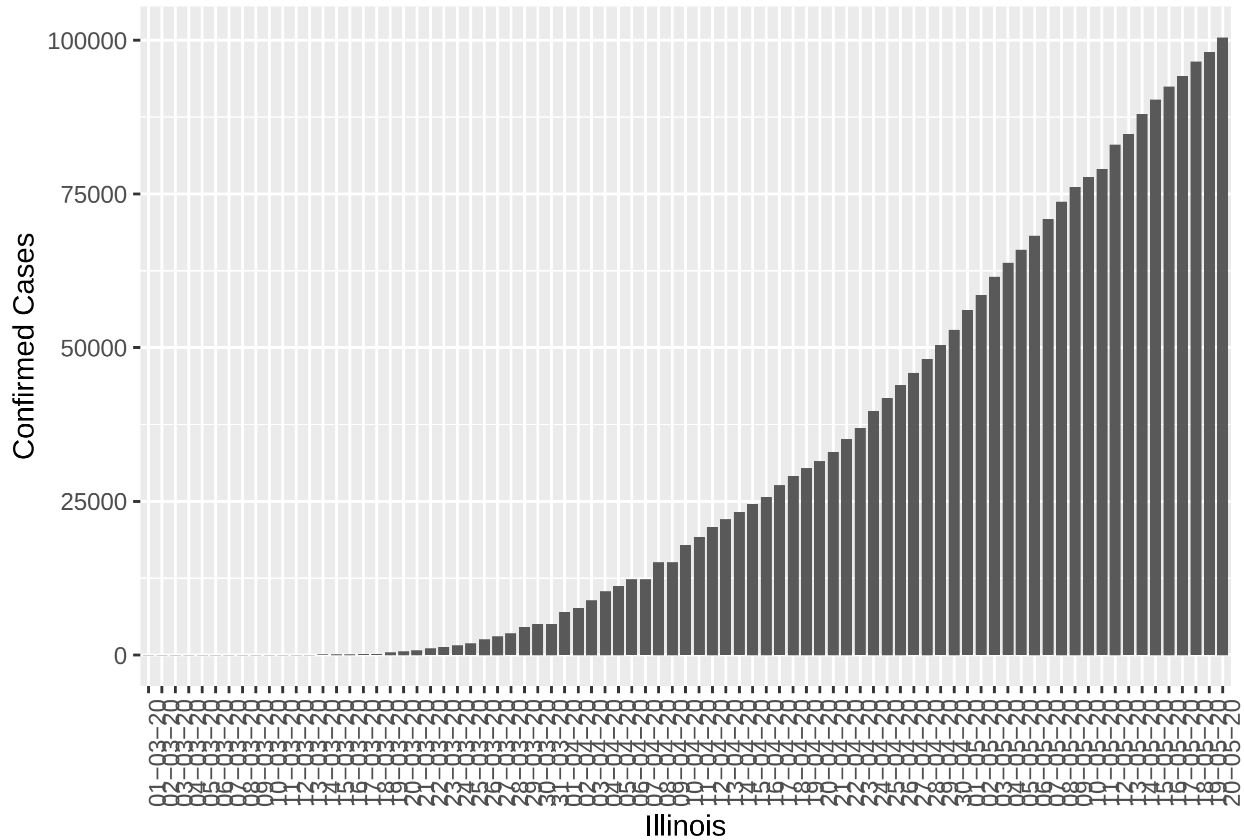
Illinois

Illinois confirmed cases plot

```
Illinoisds <- filter(ds, Province.State == "Illinois")
```

Illinois confirmed cases plot

```
plotIl <- ggplot(Illinoisds, aes(x = reorder(date, Confirmed), y = Confirmed)) +
  scale_x_discrete("Illinois") +
  scale_y_continuous("Confirmed Cases") +
  geom_bar(stat = "identity",width=.8)
plotIl<- plotIl + theme(axis.text.x=element_text(angle=90, hjust=1))
print(plotIl)
```



Create Illinois subdataset to change date of incidence to number of incidence

```
Illinoisds <- cbind(Illinoisds,(1:nrow(Illinoisds)))
names(Illinoisds)[7] <- "Day"
x <- Illinoisds$Day
y <- Illinoisds$Confirmed
```

Create the dataset of train set and test set

```
TrainP=nrow(Illinoisds)*.75
TestP =nrow(Illinoisds)*.25
train = Illinoisds[c(1:TrainP),c(1:7)]
test = Illinoisds[-c(1:TrainP),c(1:7)]
```

```
model1 <- lm(Confirmed ~ poly(Day, 1, raw = TRUE), data = train)
predictions1 <- model1 %>% predict(test)
good1 = fitted.values(model1)
graph1 = c(good1,predictions1)
tra = test$Confirmed
rs <- as.integer(predictions1)
summary(model1)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 1, raw = TRUE), data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##  -8329  -5714  -1036   4555  13504
```

```
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -11703.32    1618.73   -7.23  1.2e-09 ***
## poly(Day, 1, raw = TRUE)    809.29      46.15   17.54  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6191 on 58 degrees of freedom
## Multiple R-squared:  0.8413, Adjusted R-squared:  0.8386
## F-statistic: 307.5 on 1 and 58 DF,  p-value: < 2.2e-16
```

Evaluation

```r
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model1, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.998178655881839"
```

```r
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 35031.8634590429"
```

```r
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.450793951889131"
```

Degree 2 polynomial

```r
model2 <- lm(Confirmed ~ poly(Day, 2, raw = TRUE), data = train)
predictions2 <- model2 %>% predict(test)
good2 = fitted.values(model2)
graph2 = c(good2,predictions2)
tra = test$Confirmed
rs <- as.integer(predictions2)
summary(model2)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 2, raw = TRUE), data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1955.39  -589.32    83.54   562.07  1085.85
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               2501.1928   301.8888   8.285 2.31e-11 ***
## poly(Day, 2, raw = TRUE)1 -565.3423    22.8354 -24.757  < 2e-16 ***
## poly(Day, 2, raw = TRUE)2   22.5349     0.3628  62.106  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 753.6 on 57 degrees of freedom
## Multiple R-squared:  0.9977, Adjusted R-squared:  0.9976
## F-statistic: 1.23e+04 on 2 and 57 DF,  p-value: < 2.2e-16
```

Evaluation

```r
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model2, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.993595354992904"
```

```r
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 2347.61719791284"
```

```r
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.0302094016610708"
```

Degree 3 polynomial

```r
model3 <- lm(Confirmed ~ poly(Day, 3, raw = TRUE), data = train)
predictions3 <- model3 %>% predict(test)
good3 = fitted.values(model3)
graph3 = c(good3,predictions3)
tra = test$Confirmed
rs <- as.integer(predictions3)
summary(model3)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 3, raw = TRUE), data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1553.76  -538.28    23.67   569.01  1382.39
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                1534.36217  373.71857   4.106 0.000132 ***
## poly(Day, 3, raw = TRUE)1 -382.65298   52.62014  -7.272 1.23e-09 ***
## poly(Day, 3, raw = TRUE)2   15.10919    1.99556   7.571 3.93e-10 ***
## poly(Day, 3, raw = TRUE)3    0.08116    0.02151   3.772 0.000392 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 679 on 56 degrees of freedom
## Multiple R-squared:  0.9982, Adjusted R-squared:  0.9981
## F-statistic: 1.011e+04 on 3 and 56 DF,  p-value: < 2.2e-16
```

Evaluation

```r
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
```

```r
RSE <- sqrt(sum((predict(model3, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.991677747314202"
```

```r
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 5378.84144605132"
```

```r
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.0692155355904887"
```

Degree 4 polynomial

```r
model4 <- lm(Confirmed ~ poly(Day, 4, raw = TRUE), data = train)
predictions4 <- model4 %>% predict(test)
good4 = fitted.values(model4)
graph4 = c(good4,predictions4)
tra = test$Confirmed
rs <- as.integer(predictions4)
summary(model4)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 4, raw = TRUE), data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1306.77  -245.22    -7.46   268.19  1213.37
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               157.48905  395.16288   0.399    0.692
## poly(Day, 4, raw = TRUE)1  37.59559   88.18077   0.426    0.672
## poly(Day, 4, raw = TRUE)2 -15.29863    5.81028  -2.633    0.011 *
## poly(Day, 4, raw = TRUE)3   0.85225    0.14256   5.978 1.74e-07 ***
## poly(Day, 4, raw = TRUE)4  -0.00632    0.00116  -5.450 1.22e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 552.1 on 55 degrees of freedom
## Multiple R-squared:  0.9988, Adjusted R-squared:  0.9987
## F-statistic: 1.148e+04 on 4 and 55 DF,  p-value: < 2.2e-16
```

Evaluation

```r
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model4, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.995509428243233"
```

```r
sprintf("RSE value is : %s", RSE)
```

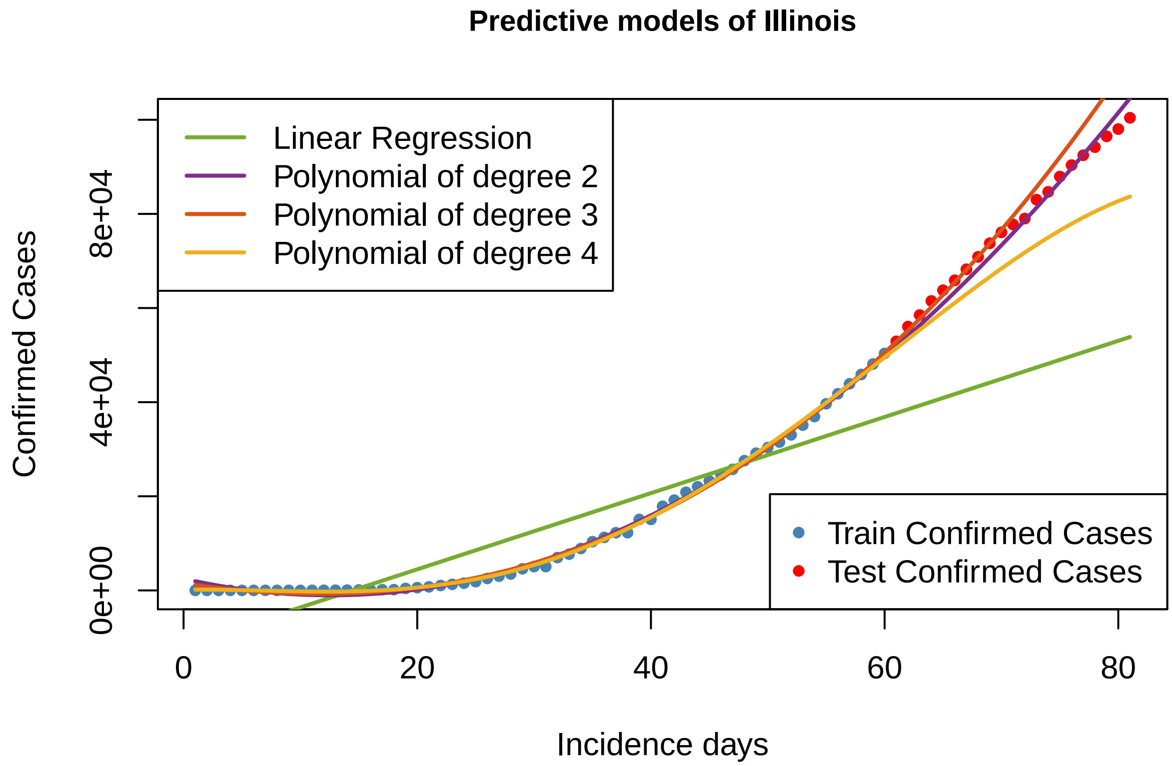
```
## [1] "RSE value is : 10091.6434509074"
```

```r
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.129860400877885"
```

Multimodel plot

```r
plot(x,y,col  = "steelblue",
     pch = 20,xlab = "Incidence days",
     ylab = "Confirmed Cases",
     main="Predictive models of Illinois",
     cex.main = 0.9)
points(test$Day,test$Confirmed, col="red", pch = 20 )
lines(x,graph1, col='#77AC30', lwd =2)
lines(x,graph2, col='#7E2F8E', lwd =2)
lines(x,graph3, col='#D95319', lwd =2)
lines(x,graph4, col='#EDB120', lwd =2)
legend("topleft", legend=c("Linear Regression", "Polynomial of degree 2","Polynomial of degree 3","Polyn
legend("bottomright",legend=c("Train Confirmed Cases","Test Confirmed Cases"), col=c('steelblue','red')
```



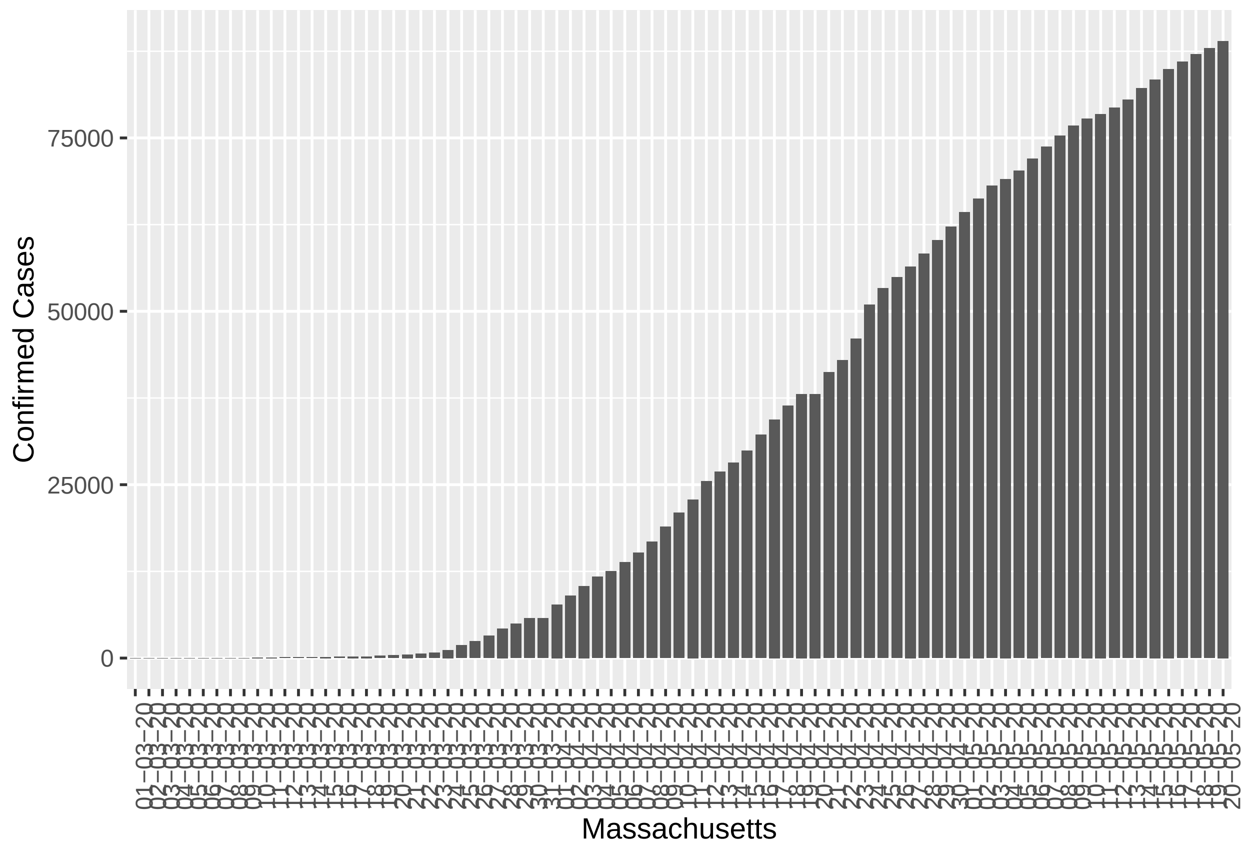
Massachusetts

Dataset of Massachusetts

```r
Massachusettsds <- filter(ds, Province.State == "Massachusetts")
```

Massachusetts confirmed cases plot

```
plotMa <- ggplot(Massachusettsds, aes(x = reorder(date, Confirmed), y = Confirmed)) +
  scale_x_discrete("Massachusetts") +
  scale_y_continuous("Confirmed Cases") +
  geom_bar(stat = "identity",width=.8)
plotMa<- plotMa + theme(axis.text.x=element_text(angle=90, hjust=1))
print(plotMa)
```



Create Massachusetts subdataset to change date of incidence to number of incidence

```
Massachusettsds <- cbind(Massachusettsds,(1:nrow(Massachusettsds)))
names(Massachusettsds)[7] <- "Day"
x <- Massachusettsds$Day
y <- Massachusettsds$Confirmed
```

Create the dataset of train set and test set

```
TrainP=nrow(Massachusettsds)*.75
TestP =nrow(Massachusettsds)*.25
train = Massachusettsds[c(1:TrainP),c(1:7)]
test = Massachusettsds[-c(1:TrainP),c(1:7)]
```

Linear Regression

```
model1 <- lm(Confirmed ~ poly(Day, 1, raw = TRUE), data = train)
predictions1 <- model1 %>% predict(test)
good1 = fitted.values(model1)
graph1 = c(good1,predictions1)
tra = test$Confirmed
rs <- as.integer(predictions1)
```

```r
summary(model1)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 1, raw = TRUE), data = train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -10538  -7662  -1232   5953  15017
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -14666.46    2070.65  -7.083 2.13e-09 ***
## poly(Day, 1, raw = TRUE)    998.58      59.04  16.914  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7920 on 58 degrees of freedom
## Multiple R-squared:  0.8314, Adjusted R-squared:  0.8285
## F-statistic: 286.1 on 1 and 58 DF,  p-value: < 2.2e-16
```

Evaluation

```r
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model1, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.992988681535783"
```

```r
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 21811.6216024649"
```

```r
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.283684831475684"
```

Degree 2 polynomial

```r
model2 <- lm(Confirmed ~ poly(Day, 2, raw = TRUE), data = train)
predictions2 <- model2 %>% predict(test)
good2 = fitted.values(model2)
graph2 = c(good2,predictions2)
tra = test$Confirmed
rs <- as.integer(predictions2)
summary(model2)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 2, raw = TRUE), data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2722.85  -851.45   -31.38   960.46  2081.87
```

```
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                3449.7140   457.1658   7.546 3.92e-10 ***
## poly(Day, 2, raw = TRUE)1 -754.5998    34.5809 -21.821  < 2e-16 ***
## poly(Day, 2, raw = TRUE)2   28.7406     0.5495  52.306  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1141 on 57 degrees of freedom
## Multiple R-squared:  0.9966, Adjusted R-squared:  0.9964
## F-statistic:  8256 on 2 and 57 DF,  p-value: < 2.2e-16
```

Evaluation

```r
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model2, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.984439987414782"
```

```r
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 23753.5562701823"
```

```r
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.308941890257867"
```

Degree 3 polynomial

```r
model3 <- lm(Confirmed ~ poly(Day, 3, raw = TRUE), data = train)
predictions3 <- model3 %>% predict(test)
good3 = fitted.values(model3)
graph3 = c(good3,predictions3)
tra = test$Confirmed
rs <- as.integer(predictions3)
summary(model3)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 3, raw = TRUE), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2342.7  -854.2  -144.9   890.8  1934.6
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                2267.63131  590.44287   3.841 0.000315 ***
## poly(Day, 3, raw = TRUE)1 -531.23710   83.13524  -6.390 3.50e-08 ***
## poly(Day, 3, raw = TRUE)2   19.66165    3.15281   6.236 6.26e-08 ***
## poly(Day, 3, raw = TRUE)3    0.09922    0.03399   2.919 0.005048 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1073 on 56 degrees of freedom
## Multiple R-squared:  0.997,  Adjusted R-squared:  0.9969
## F-statistic:  6233 on 3 and 56 DF,  p-value: < 2.2e-16
```

Evaluation

```
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model3, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.981644339198998"
```

```
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 29590.2772582226"
```

```
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.384855054351804"
```

Degree 4 polynomial

```
model4 <- lm(Confirmed ~ poly(Day, 4, raw = TRUE), data = train)
predictions4 <- model4 %>% predict(test)
good4 = fitted.values(model4)
graph4 = c(good4,predictions4)
tra = test$Confirmed
rs <- as.integer(predictions4)
summary(model4)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 4, raw = TRUE), data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2395.87  -224.24    50.39   305.09  1630.96
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -5.163e+02  5.055e+02  -1.021   0.3115
## poly(Day, 4, raw = TRUE)1  3.185e+02  1.128e+02   2.823   0.0066 **
## poly(Day, 4, raw = TRUE)2 -4.182e+01  7.432e+00  -5.627 6.40e-07 ***
## poly(Day, 4, raw = TRUE)3  1.658e+00  1.824e-01   9.094 1.50e-12 ***
## poly(Day, 4, raw = TRUE)4 -1.278e-02  1.483e-03  -8.615 8.78e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 706.2 on 55 degrees of freedom
## Multiple R-squared:  0.9987, Adjusted R-squared:  0.9986
## F-statistic: 1.081e+04 on 4 and 55 DF,  p-value: < 2.2e-16
```

Evaluation

```
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model4, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.900271291507601"
```

```
sprintf("RSE value is : %s", RSE)
```

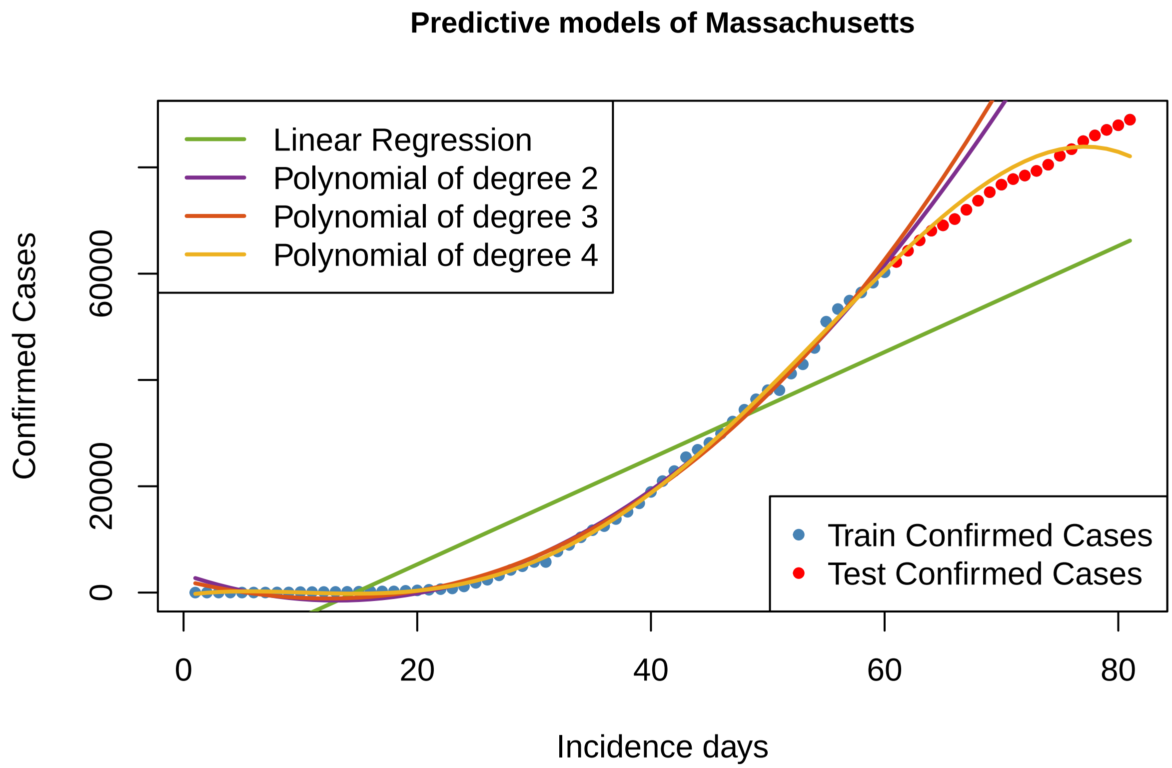
```
## [1] "RSE value is : 2780.75162430569"
```

```
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.0361668229118621"
```

Multimodel plot

```
plot(x,y,col  = "steelblue",
     pch = 20,xlab = "Incidence days",
     ylab = "Confirmed Cases",
     main="Predictive models of Massachusetts",
     cex.main = 0.9)
points(test$Day,test$Confirmed, col="red", pch = 20 )
lines(x,graph1, col='#77AC30', lwd =2)
lines(x,graph2, col='#7E2F8E', lwd =2)
lines(x,graph3, col='#D95319', lwd =2)
lines(x,graph4, col='#EDB120', lwd =2)
legend("topleft", legend=c("Linear Regression", "Polynomial of degree 2","Polynomial of degree 3","Poly
legend("bottomright",legend=c("Train Confirmed Cases","Test Confirmed Cases"), col=c('steelblue','red')
```



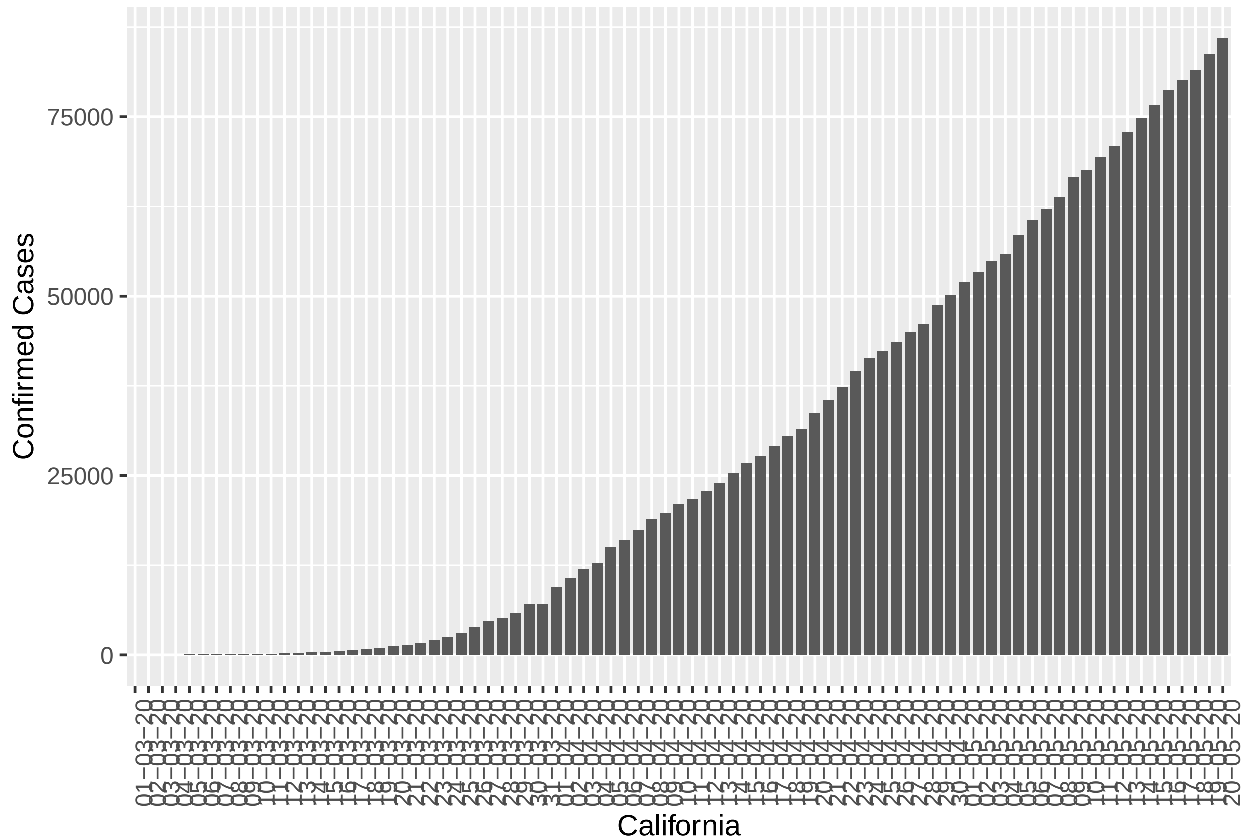
California

Dataset of California

```
Californiads <- filter(ds, Province.State == "California")
```

California confirmed cases plot

```
plotCa <- ggplot(Californiads, aes(x = reorder(date, Confirmed), y = Confirmed)) +
  scale_x_discrete("California") +
  scale_y_continuous("Confirmed Cases") +
  geom_bar(stat = "identity",width=.8)
plotCa<- plotCa + theme(axis.text.x=element_text(angle=90, hjust=1))
print(plotCa)
```



Create California subdataset to change date of incidence to number of incidence

```
Californiads <- cbind(Californiads,(1:nrow(Californiads)))
names(Californiads)[7] <- "Day"
x <- Californiads$Day
y <- Californiads$Confirmed
```

Create the dataset of train set and test set

```
TrainP=nrow(Californiads)*.75
TestP =nrow(Californiads)*.25
train =Californiads[c(1:TrainP),c(1:7)]
test = Californiads[-c(1:TrainP),c(1:7)]
```

Linear Regression

```
model1 <- lm(Confirmed ~ poly(Day, 1, raw = TRUE), data = train)
predictions1 <- model1 %>% predict(test)
good1 = fitted.values(model1)
graph1 = c(good1,predictions1)
tra = test$Confirmed
rs <- as.integer(predictions1)
summary(model1)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 1, raw = TRUE), data = train)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -7550.1 -4473.5  -960.7  4353.5 10414.7
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -11239.10    1392.70   -8.07 4.68e-11 ***
## poly(Day, 1, raw = TRUE)    836.36      39.71   21.06  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5327 on 58 degrees of freedom
## Multiple R-squared:  0.8844, Adjusted R-squared:  0.8824
## F-statistic: 443.6 on 1 and 58 DF,  p-value: < 2.2e-16
```

Evaluation

```r
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model1, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.998851469235359"
```

```r
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 21375.4261271832"
```

```r
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.316068046603465"
```

Degree 2 polynomial

```r
model2 <- lm(Confirmed ~ poly(Day, 2, raw = TRUE), data = train)
predictions2 <- model2 %>% predict(test)
good2 = fitted.values(model2)
graph2 = c(good2,predictions2)
tra = test$Confirmed
rs <- as.integer(predictions2)
summary(model2)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 2, raw = TRUE), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1798.0  -670.2   134.1   643.0  1897.7
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                886.7397   371.9688   2.384   0.0205 *
## poly(Day, 2, raw = TRUE)1 -337.1075    28.1364 -11.981   <2e-16 ***
## poly(Day, 2, raw = TRUE)2   19.2372     0.4471  43.029   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 928.6 on 57 degrees of freedom
## Multiple R-squared:  0.9965, Adjusted R-squared:  0.9964
## F-statistic:  8225 on 2 and 57 DF,  p-value: < 2.2e-16
```

Evaluation

```r
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model2, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.997718504010282"
```

```r
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 8327.43458085597"
```

```r
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.123133731488147"
```

Degree 3 polynomial

```r
model3 <- lm(Confirmed ~ poly(Day, 3, raw = TRUE), data = train)
predictions3 <- model3 %>% predict(test)
good3 = fitted.values(model3)
graph3 = c(good3,predictions3)
tra = test$Confirmed
rs <- as.integer(predictions3)
summary(model3)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 3, raw = TRUE), data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1813.28  -523.77    43.86   749.86  1486.16
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                2120.80182  456.17463   4.649 2.07e-05 ***
## poly(Day, 3, raw = TRUE)1 -570.29201   64.23007  -8.879 2.83e-12 ***
## poly(Day, 3, raw = TRUE)2   28.71538    2.43585  11.789  < 2e-16 ***
## poly(Day, 3, raw = TRUE)3   -0.10359    0.02626  -3.944 0.000225 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 828.8 on 56 degrees of freedom
## Multiple R-squared:  0.9973, Adjusted R-squared:  0.9972
## F-statistic:  6889 on 3 and 56 DF,  p-value: < 2.2e-16
```

Evaluation

```r
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model3, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)

sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.998845540484378"
```

```r
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 2248.53719029934"
```

```r
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.0332480275819797"
```

Degree 4 polynomial

```r
model4 <- lm(Confirmed ~ poly(Day, 4, raw = TRUE), data = train)
predictions4 <- model4 %>% predict(test)
good4 = fitted.values(model4)
graph4 = c(good4,predictions4)
tra = test$Confirmed
rs <- as.integer(predictions4)
summary(model4)
```

```
##
## Call:
## lm(formula = Confirmed ~ poly(Day, 4, raw = TRUE), data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1573.31  -484.81   -12.72   395.51  1618.60
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               5.851e+02  5.034e+02   1.162 0.250112
## poly(Day, 4, raw = TRUE)1 -1.016e+02  1.123e+02  -0.904 0.369829
## poly(Day, 4, raw = TRUE)2 -5.200e+00  7.402e+00  -0.702 0.485333
## poly(Day, 4, raw = TRUE)3  7.564e-01  1.816e-01   4.165 0.000111 ***
## poly(Day, 4, raw = TRUE)4 -7.049e-03  1.477e-03  -4.772 1.39e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 703.3 on 55 degrees of freedom
## Multiple R-squared:  0.9981, Adjusted R-squared:  0.9979
## F-statistic:  7181 on 4 and 55 DF,  p-value: < 2.2e-16
```

Evaluation

```r
R2 = R2(rs, tra)
RMSE = RMSE(rs, tra)
RSE <- sqrt(sum((predict(model4, test)-test$Confirmed)^2)/
              (nrow(test)-2))
errorRate = RSE/mean(test$Confirmed)
```

```r
sprintf("R^2 value is : %s", R2)
```

```
## [1] "R^2 value is : 0.638238795977524"
```

```r
sprintf("RSE value is : %s", RSE)
```

```
## [1] "RSE value is : 14613.8795958202"
```

```r
sprintf("The error rate value is: %s", errorRate)
```

```
## [1] "The error rate value is: 0.216088341334873"
```

Multimodel plot

```r
plot(x,y,col  = "steelblue",
     pch = 20,xlab = "Incidence days",
     ylab = "Confirmed Cases",
     main="Predictive models of California",
     cex.main = 0.9)
points(test$Day,test$Confirmed, col="red", pch = 20 )
lines(x,graph1, col='#77AC30', lwd =2)
lines(x,graph2, col='#7E2F8E', lwd =2)
lines(x,graph3, col='#D95319', lwd =2)
lines(x,graph4, col='#EDB120', lwd =2)
legend("topleft", legend=c("Linear Regression", "Polynomial of degree 2","Polynomial of degree 3","Poly
legend("bottomright",legend=c("Train Confirmed Cases","Test Confirmed Cases"), col=c('steelblue','red')
```