# Project Report

**Data Description :**

We have collected the data on corona virus from kaggle.com .The data variables are the date and date_ymd, corona-virus effect Status,name of states .The number of total observation 1485 of 62 variables .

**Methodology :**

We have use a Stochastic approach to meet our objective .Our Objective is to see whether the transition rate over certain range of reported cases over time and also calculating there steady state probability and predicting a range of values for number of confirm case over the next few days.

**Analysis:**

```
> train=read_csv(file=file.choose())
> head(train,10)## Here you can see the structure of the dataset and
a glimpse of the variables values.
```

| | Date | Date_YMD | Status | TT | AN | AP | AR | AS | BR | CH | CT | DN | DD | DL | GA |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 14-Mar-20 | 2020-03-14 | Confirmed | 81 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 0 |
| 2 | 14-Mar-20 | 2020-03-14 | Recovered | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | 14-Mar-20 | 2020-03-14 | Deceased | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 4 | 15-Mar-20 | 2020-03-15 | Confirmed | 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 15-Mar-20 | 2020-03-15 | Recovered | 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 6 | 15-Mar-20 | 2020-03-15 | Deceased | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 16-Mar-20 | 2020-03-16 | Confirmed | 15 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 16-Mar-20 | 2020-03-16 | Recovered | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 9 | 16-Mar-20 | 2020-03-16 | Deceased | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 17-Mar-20 | 2020-03-17 | Confirmed | 11 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

| | GJ | HR | HP | JK | JH | KA | KL | LA | LD | MP | MH | MN | ML | MZ | NL | OR | PY | PB | RJ | SK | TN | TG |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 14 | 0 | 2 | 0 | 6 | 19 | 0 | 0 | 0 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 0 | 1 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 2 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 1 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 1 | 0 | 1 | 3 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| | TR | UP | UT | WB | UN |
|---|---|---|---|---|---|
| 1 | 0 | 12 | 0 | 0 | 0 |
| 2 | 0 | 4 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 |

```
6   0  0  0  0  0
7   0  0  1  0  0
8   0  0  0  0  0
9   0  0  0  0  0
10  0  2  0  1  0
> str(train) ## Details about the dataset
'data.frame':   1485 obs. of  42 variables:
 $ Date    : chr  "14-Mar-20" "14-Mar-20" "14-Mar-20" "15-Mar-20" ...
 $ Date_YMD: chr  "2020-03-14" "2020-03-14" "2020-03-14" "2020-03-15"
...
 $ Status  : chr  "Confirmed" "Recovered" "Deceased" "Confirmed" ...
 $ TT      : int  81 9 2 27 4 0 15 1 0 11 ...
 $ AN      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ AP      : int  1 0 0 0 0 0 0 0 0 0 ...
 $ AR      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ AS      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ BR      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ CH      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ CT      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ DN      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ DD      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ DL      : int  7 1 1 0 1 0 0 0 0 1 ...
 $ GA      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ GJ      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ HR      : int  14 0 0 0 0 0 0 0 0 1 ...
 $ HP      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ JK      : int  2 0 0 0 0 0 1 0 0 0 ...
 $ JH      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ KA      : int  6 0 1 0 0 0 1 0 0 2 ...
 $ KL      : int  19 3 0 5 0 0 3 0 0 0 ...
 $ LA      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ LD      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ MP      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ MH      : int  14 0 0 18 0 0 6 0 0 3 ...
 $ MN      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ ML      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ MZ      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ NL      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ OR      : int  0 0 0 0 0 0 1 0 0 0 ...
 $ PY      : int  0 0 0 0 0 0 1 0 0 0 ...
 $ PB      : int  1 0 0 0 0 0 0 0 0 0 ...
 $ RJ      : int  3 1 0 1 2 0 0 0 0 0 ...
 $ SK      : int  0 0 0 0 0 0 0 0 0 0 ...
```

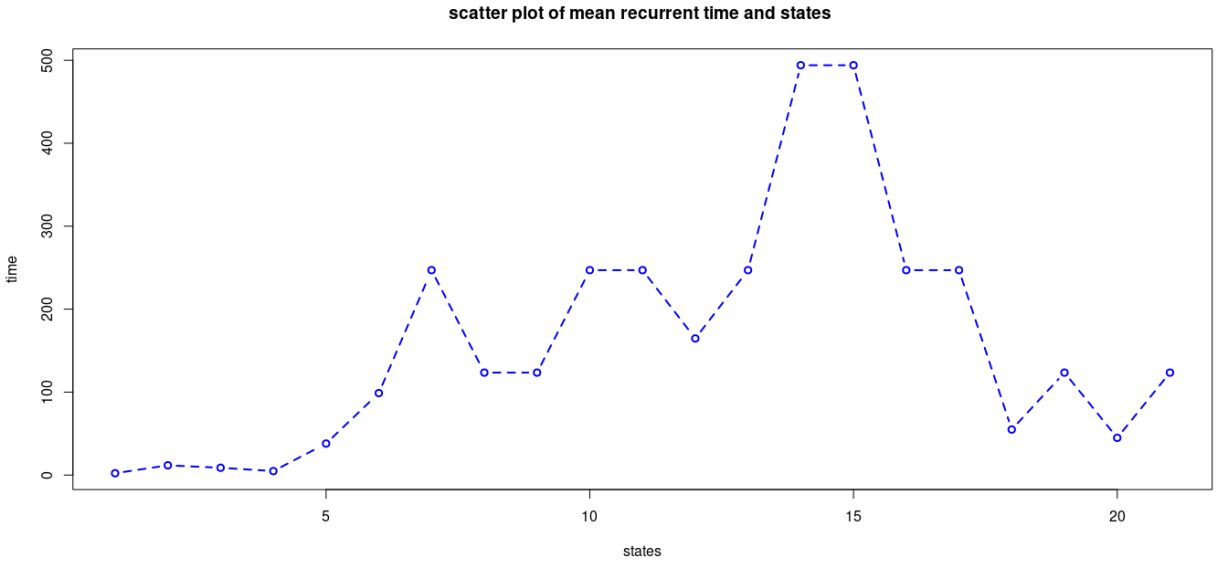
```
 $ TN      : int  1 0 0 0 0 0 0 1 0 0 ...
 $ TG      : int  1 0 0 2 1 0 1 0 0 1 ...
 $ TR      : int  0 0 0 0 0 0 0 0 0 0 ...
 $ UP      : int  12 4 0 1 0 0 0 0 0 2 ...
 $ UT      : int  0 0 0 0 0 0 1 0 0 0 ...
 $ WB      : int  0 0 0 0 0 0 0 0 0 1 ...
 $ UN      : int  0 0 0 0 0 0 0 0 0 0 ...
> ## Filtering the confirmed cases for West bengal state
> library(dplyr) ## here we are filtering the data of Confirm cases
in west bengal
> WB=train%>%
+  filter(Status=='Confirmed')%>%
+  select(Date_YMD,WB)
> head(WB) ## To see the filter data
    Date_YMD WB
1 2020-03-14  0
2 2020-03-15  0
3 2020-03-16  0
4 2020-03-17  1
5 2020-03-18  0
6 2020-03-19  0
> str(WB) ## Give a brief description about the filter data
'data.frame':   495 obs. of  2 variables:
 $ Date_YMD: chr  "2020-03-14" "2020-03-15" "2020-03-16" "2020-03-17"
...
 $ WB      : int  0 0 0 1 0 0 1 2 3 0 ...
> summary(WB) ## Descriptive stat of data
   Date_YMD               WB
 Length:495         Min.   :    0.0
 Class :character   1st Qu.:  284.5
 Mode  :character   Median : 1628.0
                    Mean   : 3071.7
                    3rd Qu.: 3312.5
                    Max.   :20846.0
Interpretation : From the output we can see that the mean,max and
minimum confirm case in the our state through the pandemic.
> WB$Date_YMD<-as.Date(WB$Date_YMD)
## Here we are changing the date variable for plotting it with
respect to the confirm cases.
> str(WB)
'data.frame':   495 obs. of  2 variables:
 $ Date_YMD: Date, format: "2020-03-14" ...
 $ WB      : int  0 0 0 1 0 0 1 2 3 0 ...
```

```
> plot(WB$Date_YMD,WB$WB)
> lines(WB$Date_YMD,WB$WB,lwd=2)


> ## Checking is there any na values in the dataset
> sapply(train,function(x)sum(is.na(x)))
    Date Date_YMD   Status       TT       AN       AP       AR
       0        0        0        0        0        0        0
      AS       BR       CH       CT       DN       DD       DL
       0        0        0        0        0        0        0
      GA       GJ       HR       HP       JK       JH       KA
       0        0        0        0        0        0        0
      KL       LA       LD       MP       MH       MN       ML
       0        0        0        0        0        0        0
      MZ       NL       OR       PY       PB       RJ       SK
       0        0        0        0        0        0        0
      TN       TG       TR       UP       UT       WB       UN
       0        0        0        0        0        0        0
>
> ## Transforming the data into states
> min(WB$WB)
[1] 0
> max(WB$WB)
[1] 20846
> corona<-WB$WB
> #transformation=function(input,t,y,interval){
>
> f<-seq(0,21000,1000)
> length(f)
[1] 22
> h=list()
> for(i in 1:(length(f)-1)){
+  h[[i]]=which(corona>=f[i]& corona<f[i+1])
+  corona[h[[i]]]=i
+ }
> corona
  [1]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
 [23]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
 [45]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
```

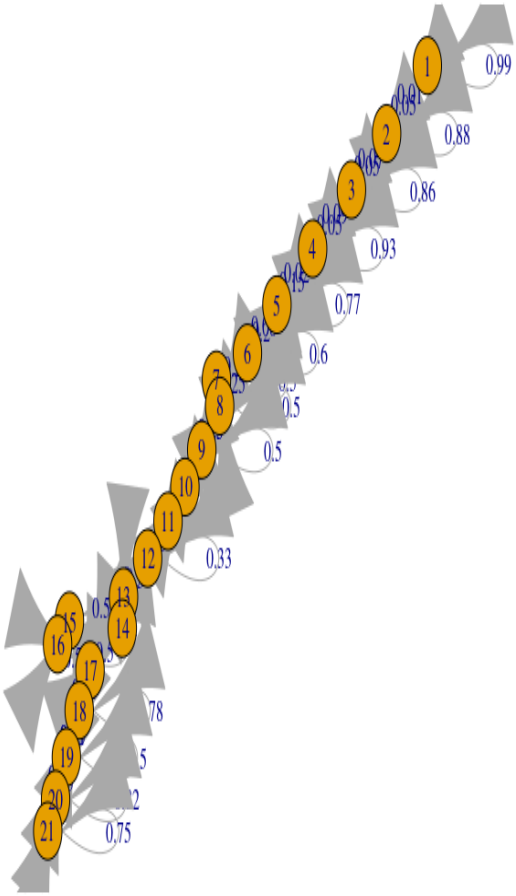
```
 [67]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
1
 [89]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
1
[111]  1  1  1  1  1  1  1  2  2  2  2  2  2  2  2  2  3  3  3  3  3
3
[133]  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
4
[155]  4  4  4  4  4  4  4  4  4  3  3  3  3  3  4  4  3  3  3  3  3
4
[177]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
4
[199]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
4
[221]  5  5  5  5  5  5  5  4  4  4  4  4  4  4  4  4  4  4  4  4  4
4
[243]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  3  4
4
[265]  4  4  4  4  3  3  3  3  3  3  3  2  3  3  3  3  3  2  2  2  2
2
[287]  2  2  2  2  2  2  2  2  1  1  1  1  1  1  1  1  1  1  1  1  1
1
[309]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
1
[331]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
1
[353]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
1
[375]  1  1  1  1  1  1  1  1  1  2  2  2  2  2  3  3  3  4  5  5  5
5
[397]  6  7  7  8  9  9 10 11 12 13 15 16 16 17 18 18 18 18 18 18 18
19
[419] 19 20 20 20 20 21 21 21 21 20 20 20 20 20 20 20 19 19 18 18 17
14
[441] 13 12 12 11 10  9  9  8  8  8  6  6  6  6  5  5  4  4  4  4  4
3
[463]  3  3  2  2  2  2  2  2  2  2  2  2  2  2  2  2  1  1  1  1  1
1
[485]  1  1  1  1  1  1  1  1  1  1  1
> ## Preparing the transition matrix
> rv=function(x,t,s){
+  # x: realization of the Markov chain
+  # t: row whose prob. we estimate
```

```
+  # s: state space
+  y=NULL
+  c=0
+  for (i in 1:(length(x)-1))
+  {
+    if(x[i]==t){
+      c=c+1
+      y[c]=x[i+1]
+    }
+  }
+  p=s*0
+  for(j in 1:length(s))
+    p[j]=sum(y==j)/length(y)
+  return(p)
+ }
> t=1
> s=1:21
> rv(corona,t,s)
 [1] 0.990990991 0.009009009 0.000000000 0.000000000 0.000000000
> p_est=matrix(0,length(s),length(s))
> for(i in 1:length(s))
+  p_est[i,]=rv(corona,i,s)
> p_est
            [,1]        [,2]       [,3]       [,4]       [,5]
 [1,] 0.99099099 0.009009009 0.00000000 0.00000000 0.00000000
 [2,] 0.04761905 0.880952381 0.07142857 0.00000000 0.00000000
 [3,] 0.00000000 0.053571429 0.85714286 0.08928571 0.00000000
 [4,] 0.00000000 0.000000000 0.04854369 0.93203883 0.01941748
 [5,] 0.00000000 0.000000000 0.00000000 0.15384615 0.76923077
 [6,] 0.00000000 0.000000000 0.00000000 0.00000000 0.20000000

            [,6] [,7] [,8] [,9] [,10]     [,11]     [,12]     [,13]
 [5,] 0.07692308  0.0 0.00 0.00  0.00 0.0000000 0.0000000 0.0000000
 [6,] 0.60000000  0.2 0.00 0.00  0.00 0.0000000 0.0000000 0.0000000
 [7,] 0.00000000  0.5 0.50 0.00  0.00 0.0000000 0.0000000 0.0000000
 [8,] 0.25000000  0.0 0.50 0.25  0.00 0.0000000 0.0000000 0.0000000
 [9,] 0.00000000  0.0 0.25 0.50  0.25 0.0000000 0.0000000 0.0000000
[10,] 0.00000000  0.0 0.00 0.50  0.00 0.5000000 0.0000000 0.0000000
[11,] 0.00000000  0.0 0.00 0.00  0.50 0.0000000 0.5000000 0.0000000
[12,] 0.00000000  0.0 0.00 0.00  0.00 0.3333333 0.3333333 0.3333333
[13,] 0.00000000  0.0 0.00 0.00  0.00 0.0000000 0.5000000 0.0000000
[14,] 0.00000000  0.0 0.00 0.00  0.00 0.0000000 0.0000000 1.0000000
      [,14] [,15] [,16]     [,17]     [,18]      [,19]     [,20]
```

```
[13,]   0.0   0.5   0.0 0.0000000 0.0000000 0.00000000 0.0000000
[15,]   0.0   0.0   1.0 0.0000000 0.0000000 0.00000000 0.0000000
[16,]   0.0   0.0   0.5 0.5000000 0.0000000 0.00000000 0.0000000
[17,]   0.5   0.0   0.0 0.0000000 0.5000000 0.00000000 0.0000000
[18,]   0.0   0.0   0.0 0.1111111 0.7777778 0.11111111 0.0000000
[19,]   0.0   0.0   0.0 0.0000000 0.2500000 0.50000000 0.2500000
[20,]   0.0   0.0   0.0 0.0000000 0.0000000 0.09090909 0.8181818
[21,]   0.0   0.0   0.0 0.0000000 0.0000000 0.00000000 0.2500000
           [,21]
[20,] 0.09090909
[21,] 0.75000000
> ## This is valid for n >= 2.
> mp=function(P,n){
+  temp=P
+  for (i in 2:n){
+    temp=temp%*%P
+  }
+  return(temp)
+ }
> library(markovchain)
> ## Calculating prediction for 7 days
> ## initial state is "1"
> initialstate=c(1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
> after2days=initialstate*(p_est*p_est)
> after7days=initialstate*(p_est^7)
> round(after2days[1,],6)
 [1] 0.982063 0.000081 0.000000 0.000000 0.000000 0.000000 0.000000
 [8] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
[15] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
> ## Checking the mean recurrent time
>
mc<-new("markovchain",states=as.character(1:21),transitionMatrix=p_es
t)
> time=meanRecurrenceTime(mc)
> states=as.character(1:21)
> plot(states,time,lty=2,lwd=2,col="Blue",main="scatter plot of mean
recurrent time and states ")
```




```
> dev.off()
null device
          1
> plot(mc)
```




```
> class(mc)
[1] "markovchain"
attr(,"package")
[1] "markovchain"
> ## Steady states
> steadyStates(mc)
             1          2         3        4           5           6
[1,] 0.4493927 0.08502024 0.1133603 0.208502 0.02631579 0.01012146
                7           8           9          10          11
[1,] 0.004048583 0.008097166 0.008097166 0.004048583 0.004048583
               12          13          14          15          16
[1,] 0.006072874 0.004048583 0.002024291 0.002024291 0.004048583
               17          18          19          20          21
```

```
[1,] 0.004048583 0.01821862 0.008097166 0.02226721 0.008097166
```