A4_R_Code

Advait Shah

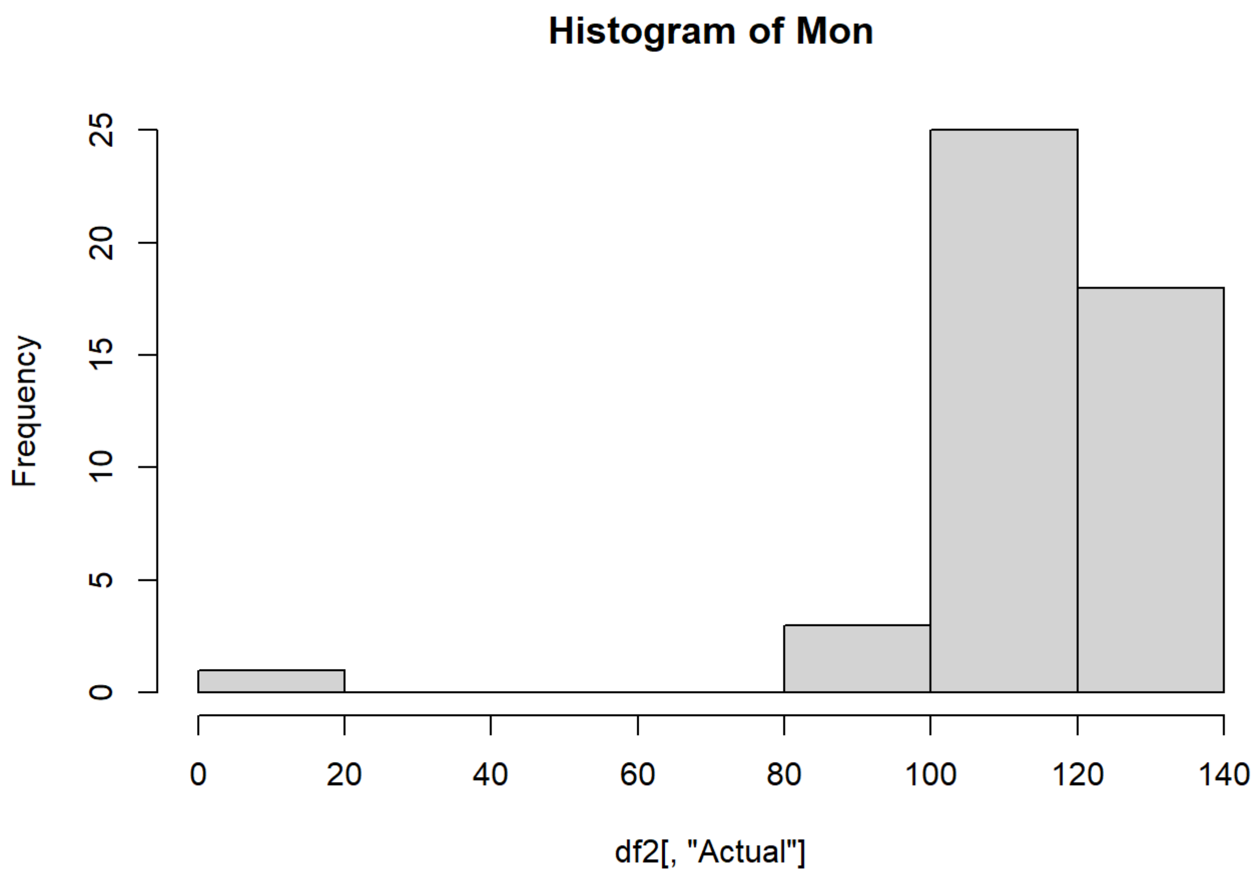
12/02/2023

```r
#a = read.csv(file.choose())

a = read.csv("Case5-data.csv")
```

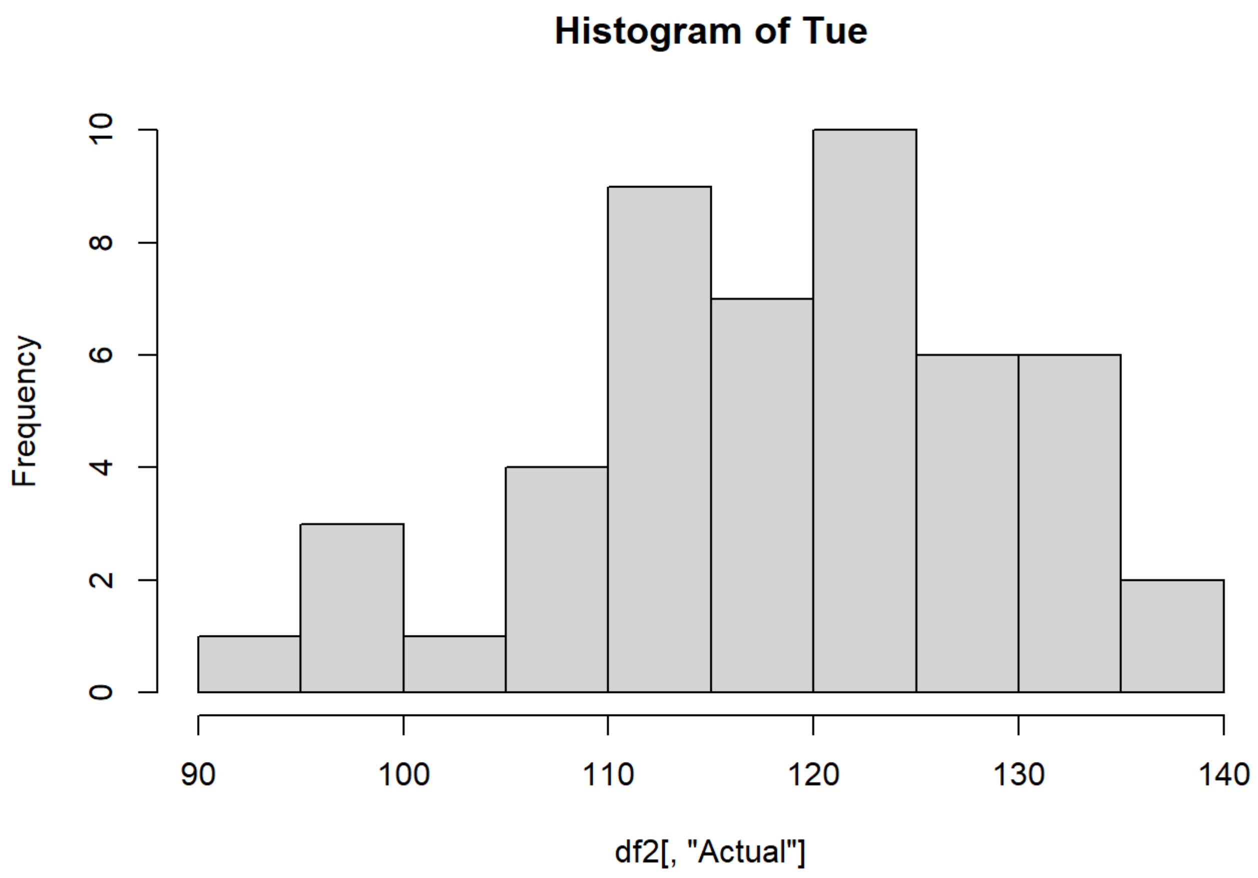
```r
head(a)
```

```
##     SurgDate DOW T...28 T...21 T...14 T...13 T...12 T...11 T...10 T...9 T...8
## 1 10-10-2011 Mon     38     45     60     63     65     70     73    73    73
## 2 11-10-2011 Tue     35     47     65     68     78     82     82    82    86
## 3 12-10-2011 Wed     26     43     54     62     72     72     72    74    87
## 4 13-10-2011 Thu     28     48     65     70     72     72     72    82    87
## 5 14-10-2011 Fri     31     40     50     50     50     54     62    68    71
## 6 17-10-2011 Mon     41     56     65     69     72     73     77    78    78
##   T...7 T...6 T...5 T...4 T...3 T...2 T...1 Actual
## 1    80    84    89    94    98   100   104    106
## 2    89    92    95    99    99    99   114    121
## 3    94    96   101   102   102   106   114    126
## 4    91    94    94    94    97    98   103    114
## 5    73    73    73    78    83    87    94    106
## 6    80    86    85    86    92    96   102    111
```

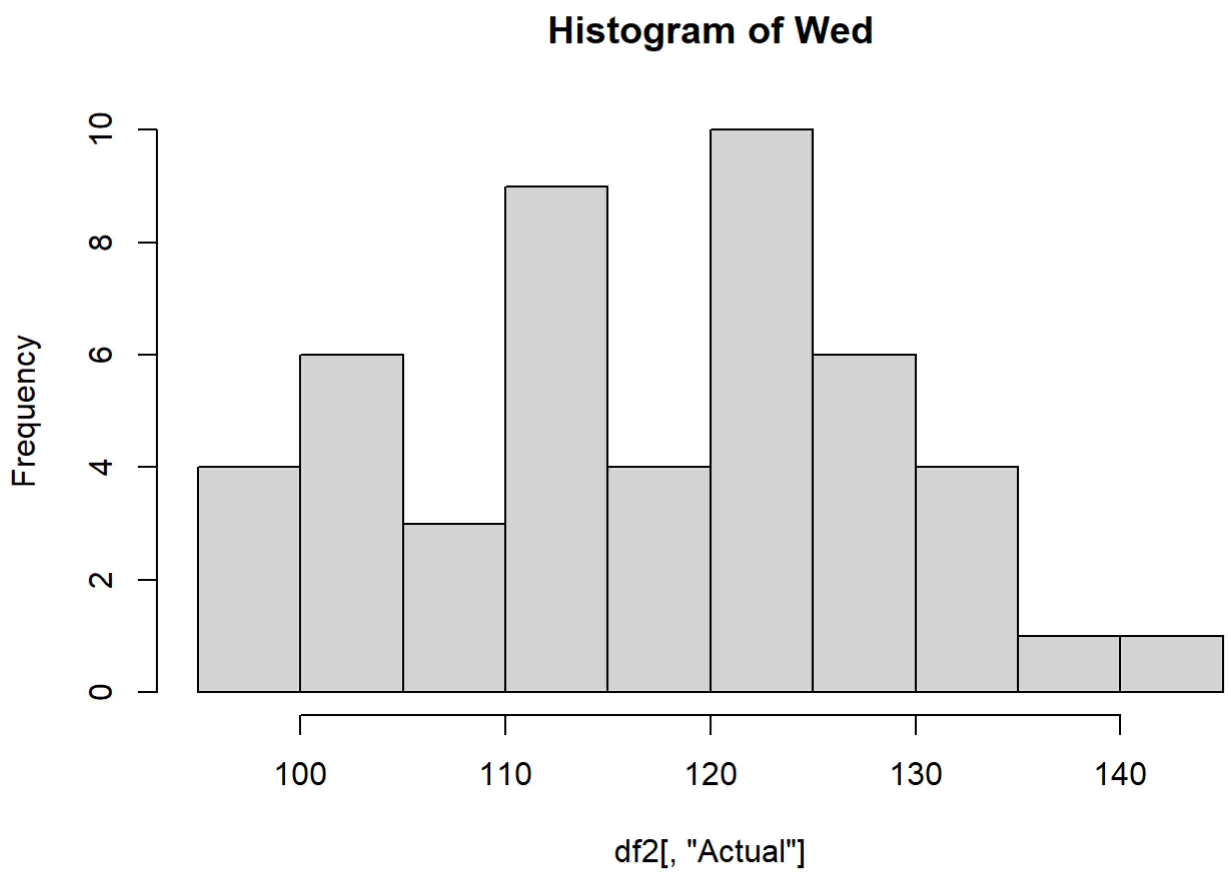
```r
days=c("Mon", "Tue", "Wed", "Thu", "Fri")
for (i in 1:5){
    df2=a[a[,2]==days[i],]
    print(hist(df2[,"Actual"],main = paste("Histogram of" , days[i])))
}
```



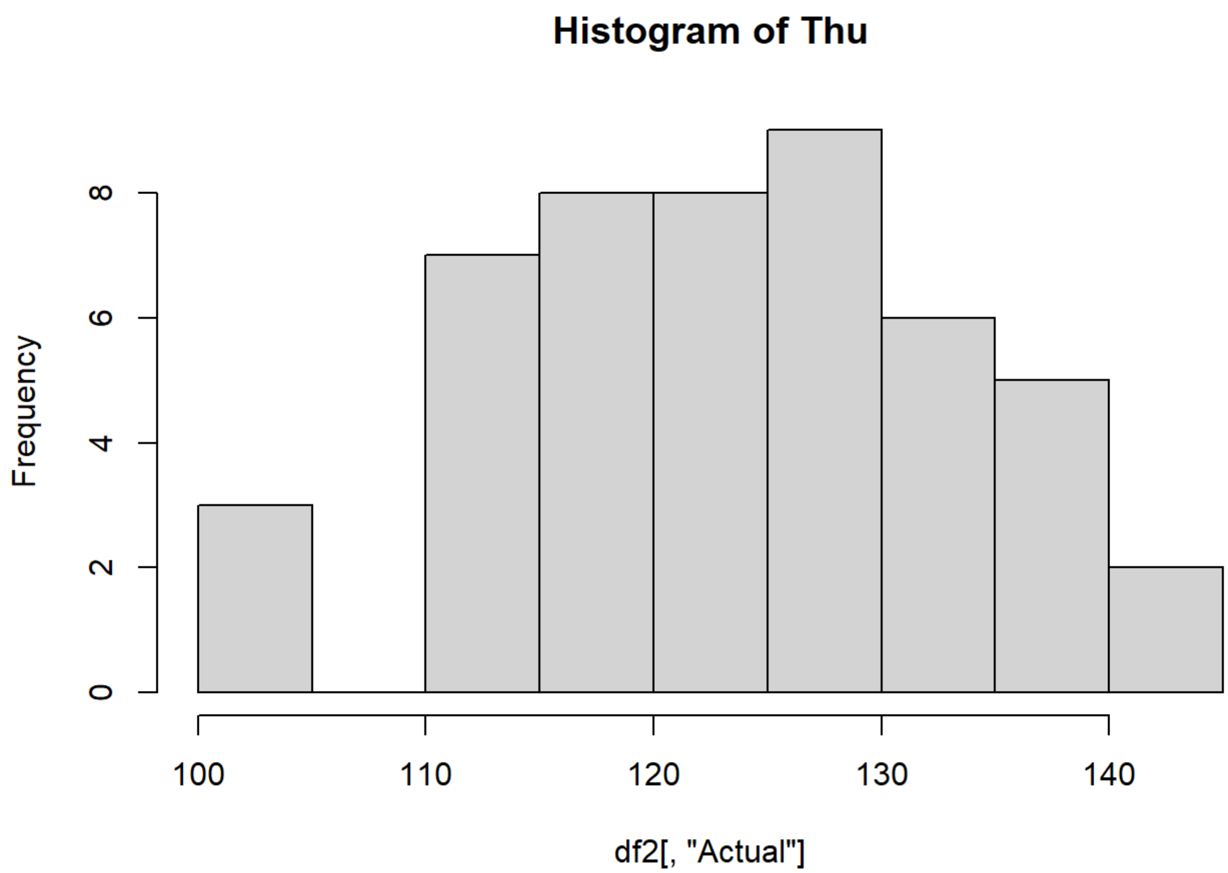

```
## $breaks
## [1]   0  20  40  60  80 100 120 140
##
## $counts
## [1]  1  0  0  0  3 25 18
##
## $density
## [1] 0.001063830 0.000000000 0.000000000 0.000000000 0.003191489 0.026595745
## [7] 0.019148936
##
## $mids
## [1]  10  30  50  70  90 110 130
##
## $xname
## [1] "df2[, \"Actual\"]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```



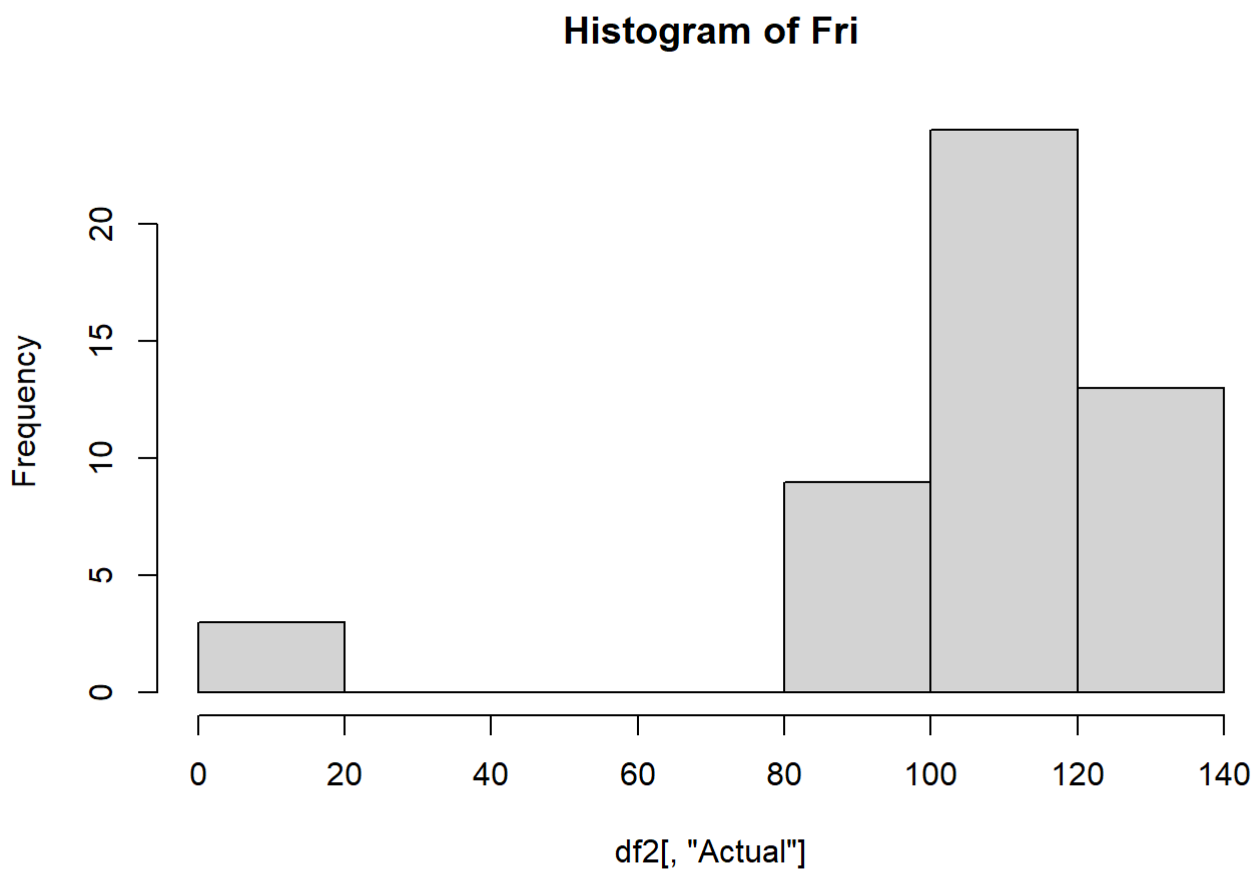

```
## $breaks
##  [1]  90  95 100 105 110 115 120 125 130 135 140
##
## $counts
##  [1]  1  3  1  4  9  7 10  6  6  2
##
## $density
##  [1] 0.004081633 0.012244898 0.004081633 0.016326531 0.036734694 0.028571429
##  [7] 0.040816327 0.024489796 0.024489796 0.008163265
##
## $mids
##  [1]  92.5  97.5 102.5 107.5 112.5 117.5 122.5 127.5 132.5 137.5
##
## $xname
## [1] "df2[, \"Actual\"]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```




```
## $breaks
##  [1]  95 100 105 110 115 120 125 130 135 140 145
##
## $counts
##  [1]  4  6  3  9  4 10  6  4  1  1
##
## $density
##  [1] 0.016666667 0.025000000 0.012500000 0.037500000 0.016666667 0.041666667
##  [7] 0.025000000 0.016666667 0.004166667 0.004166667
##
## $mids
##  [1]  97.5 102.5 107.5 112.5 117.5 122.5 127.5 132.5 137.5 142.5
##
## $xname
## [1] "df2[, \"Actual\"]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```




```
## $breaks
##  [1] 100 105 110 115 120 125 130 135 140 145
##
## $counts
## [1] 3 0 7 8 8 9 6 5 2
##
## $density
## [1] 0.012500000 0.000000000 0.029166667 0.033333333 0.033333333 0.037500000
## [7] 0.025000000 0.020833333 0.008333333
##
## $mids
## [1] 102.5 107.5 112.5 117.5 122.5 127.5 132.5 137.5 142.5
##
## $xname
## [1] "df2[, \"Actual\"]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```




```
## $breaks
## [1]   0  20  40  60  80 100 120 140
##
## $counts
## [1]  3  0  0  0  9 24 13
##
## $density
## [1] 0.003061224 0.000000000 0.000000000 0.000000000 0.009183673 0.024489796
## [7] 0.013265306
##
## $mids
## [1]  10  30  50  70  90 110 130
##
## $xname
## [1] "df2[, \"Actual\"]"
##
## $equidist
## [1] TRUE
##
## attr(,"class")
## [1] "histogram"
```

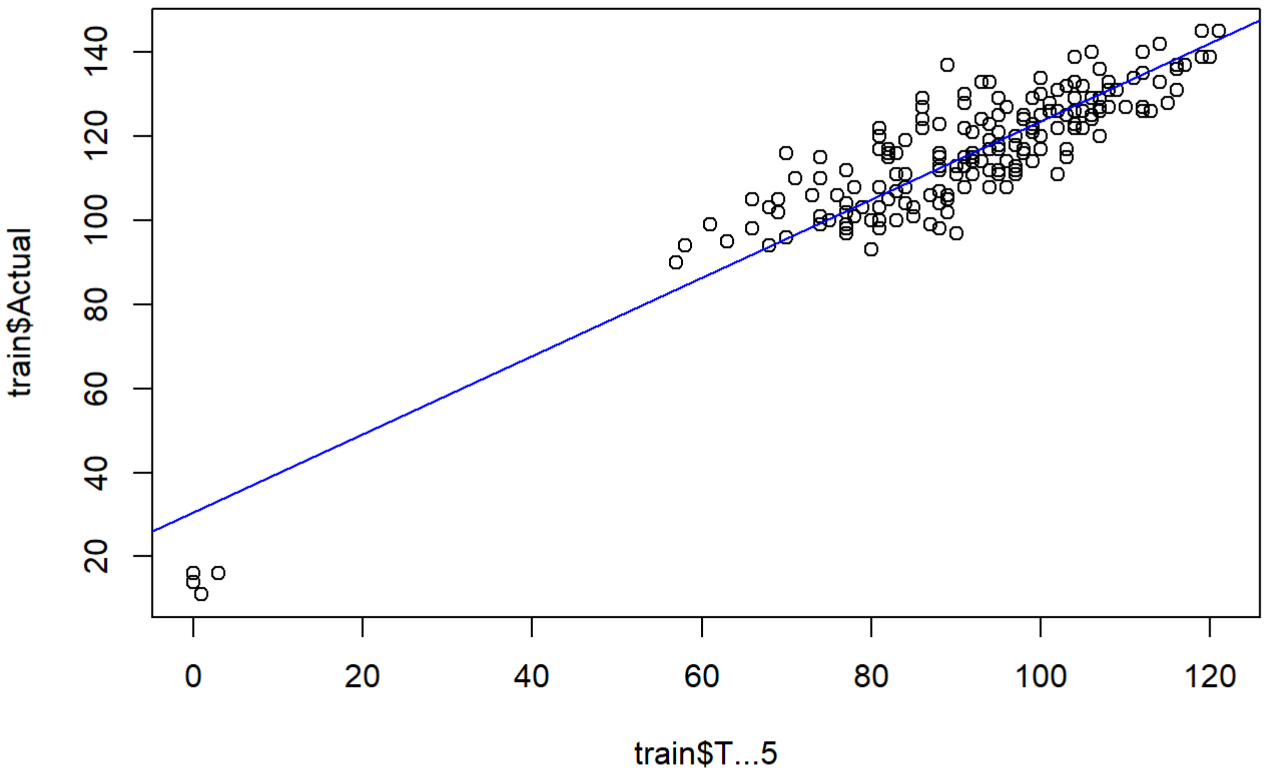
```
#make this example reproducible
set.seed(1)

#use 80% of dataset as training set and 20% as test set
sample <- sample(c(TRUE, FALSE), nrow(a), replace=TRUE, prob=c(0.8,0.2))
train  <- a[sample, ]
test   <- a[!sample, ]

t5_model = lm(Actual~T...5, train)
print(paste("T-5 based model R-Squared:",summary(t5_model)$r.squared))
```

```
## [1] "T-5 based model R-Squared: 0.827860438875018"
```

```
print(paste("T-5 based model MSE:",mean((test$Actual - predict.lm(t5_model, test)) ^ 2)))
```

```
## [1] "T-5 based model MSE: 41.6998887704155"
```

```
plot(train$T...5,train$Actual)
abline(t5_model, col = "blue")
```



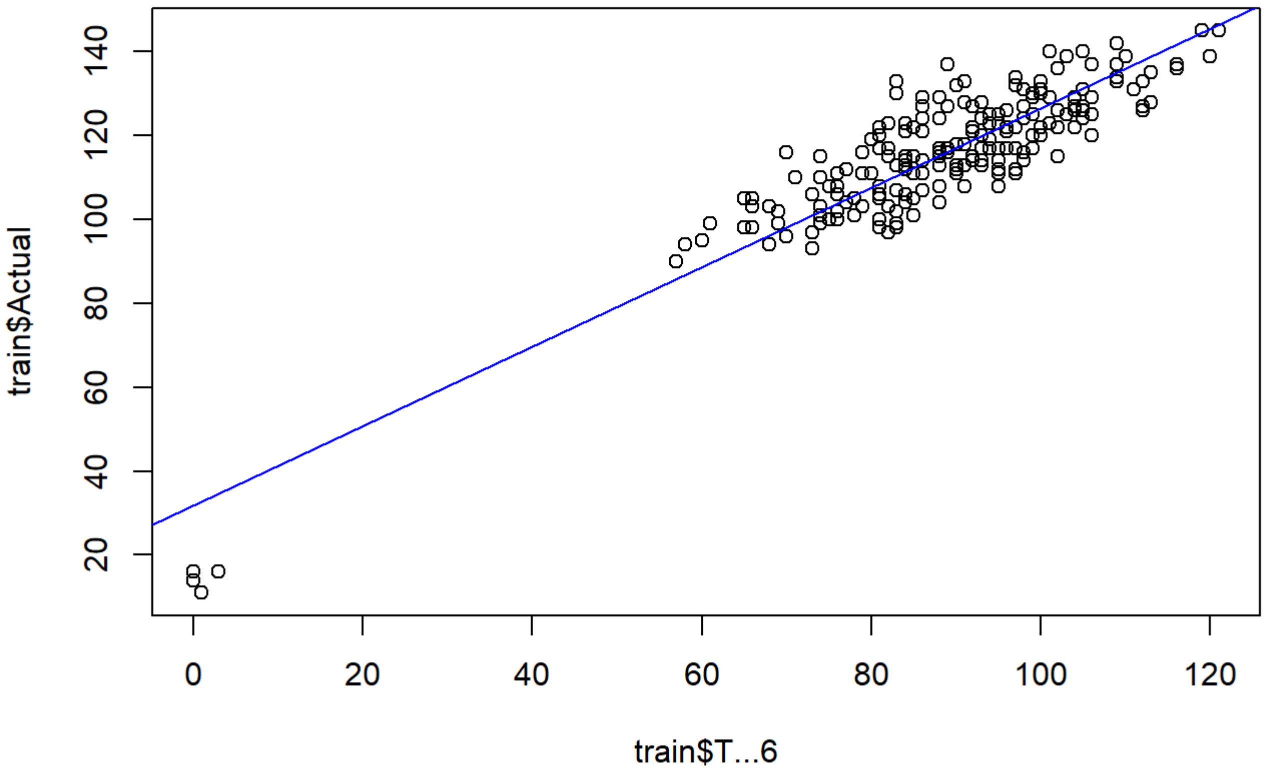

```
t6_model = lm(Actual~T...6, train)
print(paste("T-6 based model R-Squared:",summary(t6_model)$r.squared))
```

```
## [1] "T-6 based model R-Squared: 0.819831417075027"
```

```
print(paste("T-6 based model MSE:",mean((test$Actual - predict.lm(t6_model, test)) ^ 2)))
```

```
## [1] "T-6 based model MSE: 40.7105698327936"
```

```
plot(train$T...6,train$Actual)
abline(t6_model, col = "blue")
```



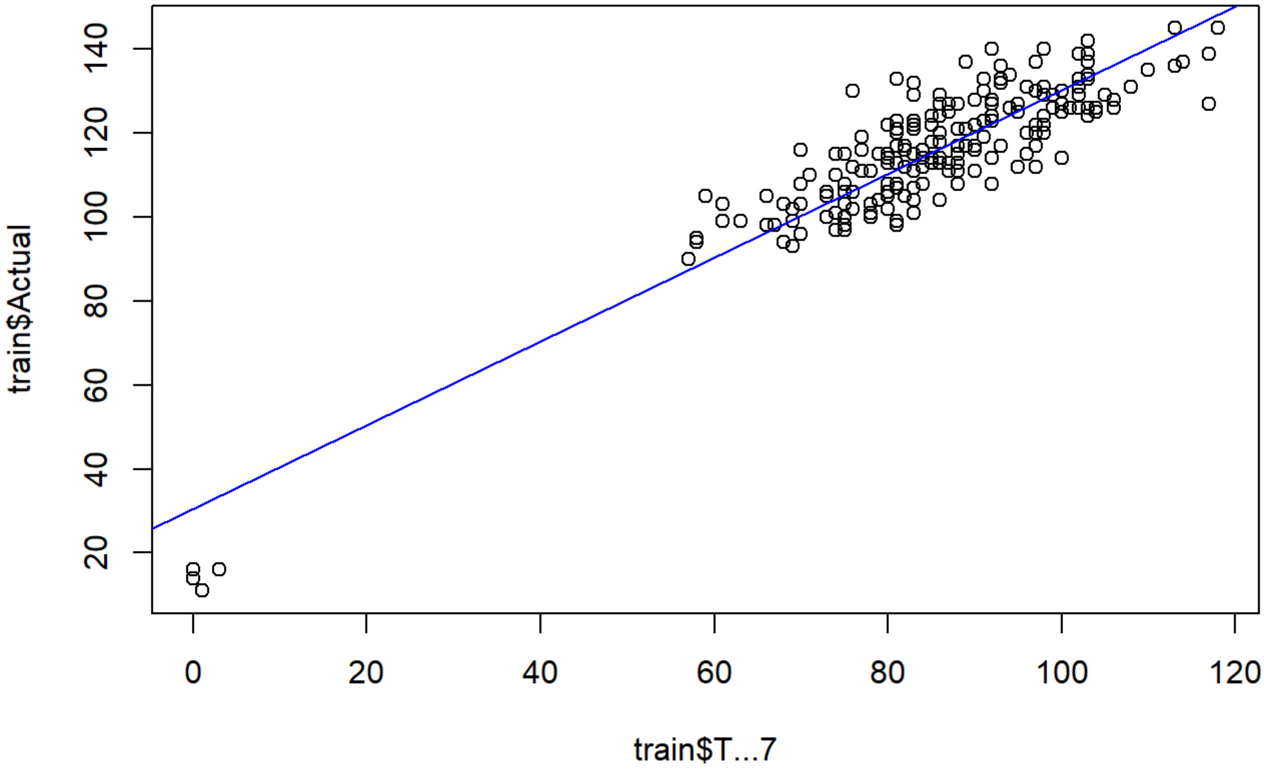

```
t7_model = lm(Actual~T...7, train)
print(paste("T-7 based model R-Squared:",summary(t7_model)$r.squared))
```

```
## [1] "T-7 based model R-Squared: 0.81812575168335"
```

```
print(paste("T-7 based model MSE:",mean((test$Actual - predict.lm(t7_model, test)) ^ 2)))
```

```
## [1] "T-7 based model MSE: 47.9691965433956"
```

```
plot(train$T...7,train$Actual)
abline(t7_model, col = "blue")
```



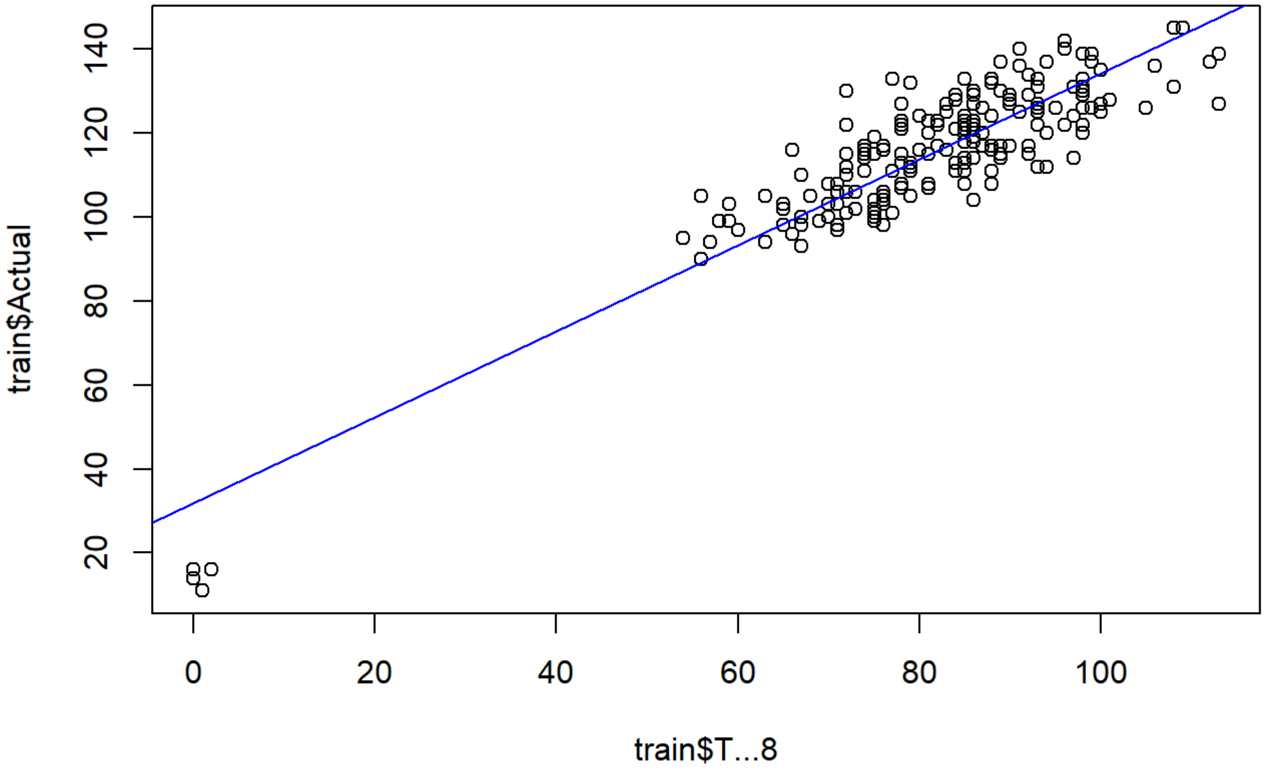

```
t8_model = lm(Actual~T...8, train)
print(paste("T-8 based model R-Squared:",summary(t8_model)$r.squared))
```

```
## [1] "T-8 based model R-Squared: 0.808648102575112"
```

```
print(paste("T-8 based model MSE:",mean((test$Actual - predict.lm(t8_model, test)) ^ 2)))
```

```
## [1] "T-8 based model MSE: 58.2459247847008"
```

```
plot(train$T...8,train$Actual)
abline(t8_model, col = "blue")
```



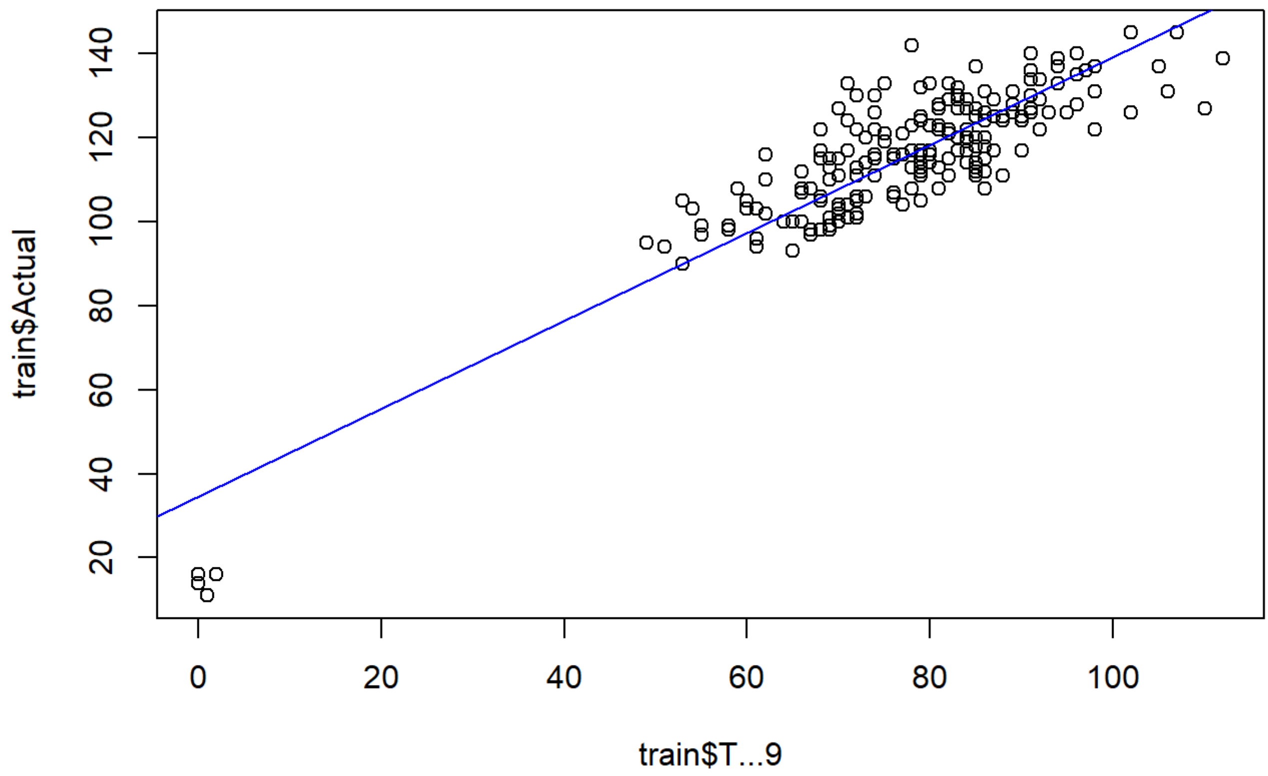

```
t9_model = lm(Actual~T...9, train)
print(paste("T-9 based model R-Squared:",summary(t9_model)$r.squared))
```

```
## [1] "T-9 based model R-Squared: 0.785580610181517"
```

```
print(paste("T-9 based model MSE:",mean((test$Actual - predict.lm(t9_model, test)) ^ 2)))
```

```
## [1] "T-9 based model MSE: 66.1200884449541"
```

```
plot(train$T...9,train$Actual)
abline(t9_model, col = "blue")
```



[1] “T-5 based model R-Squared: 0.827860438875018” [1] “T-5 based model MSE: 41.6998887704155” [1] “T-6 based model R-Squared: 0.819831417075027” [1] “T-6 based model MSE: 40.7105698327936” [1] “T-7 based model R-Squared: 0.81812575168335” [1] “T-7 based model MSE: 47.9691965433956” [1] “T-8 based model R-Squared: 0.808648102575112” [1] “T-8 based model MSE: 58.2459247847008” [1] “T-9 based model R-Squared: 0.785580610181517” [1] “T-9 based model MSE: 66.1200884449541”

As we can see, T-5 and T-6 gives the least mean square error in predicting the actual surgeries when tested on the test dataset. whereas, T-7, T-8, T-9 predictors based model gave comparatively higher prediction error. So, we could argue that it is a trade off between accuracy level and prediction timing. We also plotted the best fit linear regression models based on each of these predictors, and calculated R-Squared values. It also suggests that linear regression model could be best fit on the T-5 and T-6 based predictors and thus, they gave better R-square values compared to T-7, T-8, T-9 predictors. This also explains why we have better prediction accuracy with T-5 and T-6 based models, as their linear regression models are more robust due to less bias in training dataset.

I would suggest T-6 based model to be used for predictions, because it has high prediction accuracy as well as team will have this information 6 days in advance of surgery, which would be sufficient in planning the schedule accurately about 6 days in advance of actual surgery days.

```
# checking correlation between raw predictors
cor(a[,3:19])
```

```
##            T...28    T...21    T...14    T...13    T...12    T...11    T...10
## T...28 1.0000000 0.8947001 0.7669813 0.7612578 0.7642718 0.7696805 0.7442815
## T...21 0.8947001 1.0000000 0.8714275 0.8625057 0.8491198 0.8396694 0.8218751
## T...14 0.7669813 0.8714275 1.0000000 0.9755926 0.9403742 0.9188442 0.9134200
## T...13 0.7612578 0.8625057 0.9755926 1.0000000 0.9773372 0.9550263 0.9415538
## T...12 0.7642718 0.8491198 0.9403742 0.9773372 1.0000000 0.9866184 0.9620743
## T...11 0.7696805 0.8396694 0.9188442 0.9550263 0.9866184 1.0000000 0.9792885
## T...10 0.7442815 0.8218751 0.9134200 0.9415538 0.9620743 0.9792885 1.0000000
## T...9  0.7186066 0.8073506 0.9247739 0.9404125 0.9415326 0.9477643 0.9733221
## T...8  0.6978915 0.7946385 0.9199291 0.9311218 0.9221583 0.9181422 0.9351916
## T...7  0.6698647 0.7692789 0.9004517 0.9144495 0.9040636 0.8964697 0.9122042
## T...6  0.6694209 0.7713110 0.8901081 0.9119550 0.9128071 0.9064878 0.9185980
## T...5  0.6797108 0.7667651 0.8635365 0.8955542 0.9194134 0.9202565 0.9222467
## T...4  0.6854683 0.7662302 0.8460239 0.8782672 0.9109578 0.9239382 0.9279820
## T...3  0.6861281 0.7637451 0.8456964 0.8705655 0.8938988 0.9088628 0.9261966
## T...2  0.6550219 0.7429564 0.8481115 0.8627048 0.8769547 0.8856744 0.9079661
## T...1  0.6294322 0.7183636 0.8214784 0.8350405 0.8473868 0.8518777 0.8711997
## Actual 0.6082898 0.7024592 0.8008768 0.8127298 0.8187144 0.8198549 0.8421934
##             T...9     T...8     T...7     T...6     T...5     T...4     T...3
## T...28 0.7186066 0.6978915 0.6698647 0.6694209 0.6797108 0.6854683 0.6861281
## T...21 0.8073506 0.7946385 0.7692789 0.7713110 0.7667651 0.7662302 0.7637451
## T...14 0.9247739 0.9199291 0.9004517 0.8901081 0.8635365 0.8460239 0.8456964
## T...13 0.9404125 0.9311218 0.9144495 0.9119550 0.8955542 0.8782672 0.8705655
## T...12 0.9415326 0.9221583 0.9040636 0.9128071 0.9194134 0.9109578 0.8938988
## T...11 0.9477643 0.9181422 0.8964697 0.9064878 0.9202565 0.9239382 0.9088628
## T...10 0.9733221 0.9351916 0.9122042 0.9185980 0.9222467 0.9279820 0.9261966
## T...9  1.0000000 0.9715325 0.9550606 0.9456784 0.9333638 0.9258257 0.9245283
## T...8  0.9715325 1.0000000 0.9848287 0.9692359 0.9483349 0.9300646 0.9203496
## T...7  0.9550606 0.9848287 1.0000000 0.9845418 0.9600000 0.9383918 0.9255027
## T...6  0.9456784 0.9692359 0.9845418 1.0000000 0.9839807 0.9632278 0.9465638
## T...5  0.9333638 0.9483349 0.9600000 0.9839807 1.0000000 0.9849111 0.9643172
## T...4  0.9258257 0.9300646 0.9383918 0.9632278 0.9849111 1.0000000 0.9841580
## T...3  0.9245283 0.9203496 0.9255027 0.9465638 0.9643172 0.9841580 1.0000000
## T...2  0.9228736 0.9277084 0.9342842 0.9506493 0.9596923 0.9687847 0.9831174
## T...1  0.8951393 0.9092333 0.9181239 0.9279542 0.9373311 0.9431323 0.9509280
## Actual 0.8728896 0.8876746 0.8957787 0.8989004 0.9028267 0.9060397 0.9132423
##             T...2     T...1    Actual
## T...28 0.6550219 0.6294322 0.6082898
## T...21 0.7429564 0.7183636 0.7024592
## T...14 0.8481115 0.8214784 0.8008768
## T...13 0.8627048 0.8350405 0.8127298
## T...12 0.8769547 0.8473868 0.8187144
## T...11 0.8856744 0.8518777 0.8198549
## T...10 0.9079661 0.8711997 0.8421934
## T...9  0.9228736 0.8951393 0.8728896
## T...8  0.9277084 0.9092333 0.8876746
## T...7  0.9342842 0.9181239 0.8957787
## T...6  0.9506493 0.9279542 0.8989004
## T...5  0.9596923 0.9373311 0.9028267
## T...4  0.9687847 0.9431323 0.9060397
## T...3  0.9831174 0.9509280 0.9132423
## T...2  1.0000000 0.9700632 0.9364301
```

```
## T...1  0.9700632 1.0000000 0.9647269
## Actual 0.9364301 0.9647269 1.0000000
```

This suggests predictors are highly correlated

```
#transforming columns to reduce correlation and create new predictors
df = data.frame(a$DOW,a$Actual)
for (i in 3:18){
  df[,i] <- a[,i+1] - a[,i]
}

head(df)
```

```
##   a.DOW a.Actual V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18
## 1   Mon      106  7 15  3  2  5  3  0   0   7   4   5   5   4   2   4   2
## 2   Tue      121 12 18  3 10  4  0  0   4   3   3   3   4   0   0  15   7
## 3   Wed      126 17 11  8 10  0  0  2  13   7   2   5   1   0   4   8  12
## 4   Thu      114 20 17  5  2  0  0 10   5   4   3   0   0   3   1   5  11
## 5   Fri      106  9 10  0  0  4  8  6   3   2   0   0   5   5   4   7  12
## 6   Mon      111 15  9  4  3  1  4  1   0   2   6  -1   1   6   4   6   9
```

```
cor(df[,2:18])
```

```
##            a.Actual            V3           V4           V5          V6
## a.Actual 1.00000000  0.4387188710  0.430460180  0.233514088  0.20064928
## V3       0.43871887  1.0000000000  0.085278214  0.075292587 -0.05653795
## V4       0.43046018  0.0852782138  1.000000000 -0.032837320 -0.12036956
## V5       0.23351409  0.0752925875 -0.032837320  1.000000000  0.29963241
## V6       0.20064928 -0.0565379465 -0.120369558  0.299632409  1.00000000
## V7       0.03972180 -0.1700832420 -0.151003310 -0.036153646  0.24055643
## V8       0.14382801  0.0540182054  0.113227291 -0.170295299 -0.25605335
## V9       0.12258045  0.0593385279  0.201505217 -0.237344504 -0.39855741
## V10      0.18992776  0.1101809539  0.123514207 -0.048779413 -0.17825376
## V11      0.20302726  0.0573891014  0.111827293  0.093094290 -0.03123994
## V12      0.26200727  0.1590979681  0.004177361  0.258653719  0.33282301
## V13      0.15485553 -0.0914336913 -0.186146694  0.268414589  0.61490297
## V14      0.04769927 -0.0533199467 -0.185790129 -0.005863455  0.22793634
## V15      0.07584365 -0.0207768508  0.036230015 -0.175276205 -0.24357969
## V16      0.11391471  0.0545345681  0.213964551 -0.253732554 -0.24487502
## V17      0.16774121 -0.0063629971 -0.025014011 -0.029635744 -0.04711121
## V18      0.10023360 -0.0008892634 -0.070056359 -0.054439086 -0.13748347
##                   V7          V8          V9          V10         V11
## a.Actual  0.03972180  0.14382801  0.12258045  0.189927759  0.20302726
## V3       -0.17008324  0.05401821  0.05933853  0.110180954  0.05738910
## V4       -0.15100331  0.11322729  0.20150522  0.123514207  0.11182729
## V5       -0.03615365 -0.17029530 -0.23734450 -0.048779413  0.09309429
## V6        0.24055643 -0.25605335 -0.39855741 -0.178253762 -0.03123994
## V7        1.00000000  0.11593586 -0.29620821 -0.257168308 -0.12782580
## V8        0.11593586  1.00000000  0.09596981 -0.158029789 -0.02084597
## V9       -0.29620821  0.09596981  1.00000000  0.193339581  0.18896773
## V10      -0.25716831 -0.15802979  0.19333958  1.000000000  0.06389516
## V11      -0.12782580 -0.02084597  0.18896773  0.063895156  1.00000000
## V12       0.04293635 -0.06668441 -0.33541318 -0.082325235  0.06007495
## V13       0.24692634 -0.26187287 -0.37100426 -0.177848747 -0.09941007
## V14       0.42852077  0.06082614 -0.33026216 -0.261193841 -0.12156296
## V15       0.06914415  0.36783426  0.01120868 -0.195192470 -0.10403560
## V16      -0.21333424  0.12602889  0.39087753  0.203840755  0.04889313
## V17      -0.10644765 -0.05780631  0.16516317  0.152398441  0.05226849
## V18      -0.08324249  0.04647092  0.10765362 -0.007550471 -0.03814337
##                   V12          V13          V14         V15         V16
## a.Actual  0.262007265  0.154855531  0.047699271  0.07584365  0.11391471
## V3        0.159097968 -0.091433691 -0.053319947 -0.02077685  0.05453457
## V4        0.004177361 -0.186146694 -0.185790129  0.03623001  0.21396455
## V5        0.258653719  0.268414589 -0.005863455 -0.17527620 -0.25373255
## V6        0.332823013  0.614902974  0.227936337 -0.24357969 -0.24487502
## V7        0.042936353  0.246926337  0.428520766  0.06914415 -0.21333424
## V8       -0.066684408 -0.261872870  0.060826136  0.36783426  0.12602889
## V9       -0.335413183 -0.371004256 -0.330262157  0.01120868  0.39087753
## V10      -0.082325235 -0.177848747 -0.261193841 -0.19519247  0.20384075
## V11       0.060074949 -0.099410070 -0.121562959 -0.10403560  0.04889313
## V12       1.000000000  0.282735968  0.006464538 -0.12370849 -0.13506213
## V13       0.282735968  1.000000000  0.172891048 -0.12757307 -0.26499132
## V14       0.006464538  0.172891048  1.000000000  0.15532127 -0.33978771
## V15      -0.123708493 -0.127573072  0.155321273  1.00000000 -0.05115462
## V16      -0.135062133 -0.264991319 -0.339787708 -0.05115462  1.00000000
## V17      -0.149643413  0.004412088 -0.076388266 -0.14887133  0.05637770
## V18      -0.173948762 -0.136714109 -0.062228387 -0.02059245  0.08136524
##                   V17          V18
```

```
## a.Actual  0.167741214  0.1002335955
## V3       -0.006362997 -0.0008892634
## V4       -0.025014011 -0.0700563594
## V5       -0.029635744 -0.0544390856
## V6       -0.047111210 -0.1374834693
## V7       -0.106447648 -0.0832424928
## V8       -0.057806310  0.0464709186
## V9        0.165163169  0.1076536235
## V10       0.152398441 -0.0075504711
## V11       0.052268494 -0.0381433746
## V12      -0.149643413 -0.1739487615
## V13       0.004412088 -0.1367141088
## V14      -0.076388266 -0.0622283875
## V15      -0.148871331 -0.0205924538
## V16       0.056377697  0.0813652380
## V17       1.000000000 -0.0380040786
## V18      -0.038004079  1.0000000000
```

the predictors are not correlated now, and hence we will include all of these predictors in our actual surgeries linear regression prediction.

```
#removing DOW from df for model 1

df1 = df[c(-1)]
head(df1)
```

```
##   a.Actual V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14 V15 V16 V17 V18
## 1      106  7 15  3  2  5  3  0   0   7   4   5   5   4   2   4   2
## 2      121 12 18  3 10  4  0  0   4   3   3   3   4   0   0  15   7
## 3      126 17 11  8 10  0  0  2  13   7   2   5   1   0   4   8  12
## 4      114 20 17  5  2  0  0 10   5   4   3   0   0   3   1   5  11
## 5      106  9 10  0  0  4  8  6   3   2   0   0   5   5   4   7  12
## 6      111 15  9  4  3  1  4  1   0   2   6  -1   1   6   4   6   9
```

```
model_1_not_str = lm(a.Actual~., df1)

summary(model_1_not_str)
```

```
##
## Call:
## lm(formula = a.Actual ~ ., data = df1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -27.0749  -6.7439   0.1262   5.7944  23.7427
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 28.26458    3.61133   7.827 1.97e-13 ***
## V3           1.26528    0.12082  10.472  < 2e-16 ***
## V4           1.09476    0.09467  11.563  < 2e-16 ***
## V5           1.17473    0.21123   5.561 7.60e-08 ***
## V6           1.29381    0.25981   4.980 1.27e-06 ***
## V7           1.22558    0.28702   4.270 2.89e-05 ***
## V8           0.76784    0.22280   3.446 0.000679 ***
## V9           1.09053    0.21554   5.059 8.75e-07 ***
## V10          1.19785    0.17721   6.760 1.18e-10 ***
## V11          0.84311    0.22037   3.826 0.000169 ***
## V12          1.28450    0.22854   5.620 5.63e-08 ***
## V13          1.12558    0.25647   4.389 1.76e-05 ***
## V14          0.94348    0.23563   4.004 8.47e-05 ***
## V15          1.31492    0.21856   6.016 7.21e-09 ***
## V16          0.59949    0.21436   2.797 0.005613 **
## V17          0.91463    0.14303   6.395 9.25e-10 ***
## V18          0.91519    0.12995   7.043 2.29e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.137 on 224 degrees of freedom
## Multiple R-squared:  0.7493, Adjusted R-squared:  0.7314
## F-statistic: 41.84 on 16 and 224 DF,  p-value: < 2.2e-16
```

So, we are getting 74.93 R-squared value in this linear regression fit. But, this model would not be much helpful in predicting surgeries in advance, because its predictors require data of daily addition in scheduled surgeries even close to surgery day.

```
# model 2: Stratified by the day of the week

DOW = c("Mon","Tue","Wed","Thu","Fri")

for (i in 1:5){
  df2 = df[df[,1]==DOW[i],]
  df2 = df2[c(-1)]
  model1 = lm(a.Actual~., df2)
  print(paste(DOW[i],"Multiple LR model R-squared value:",summary(model1)$r.squared))

}
```

```
## [1] "Mon Multiple LR model R-squared value: 0.866348077861499"
## [1] "Tue Multiple LR model R-squared value: 0.646609148632659"
## [1] "Wed Multiple LR model R-squared value: 0.686860674014442"
## [1] "Thu Multiple LR model R-squared value: 0.572552488179656"
## [1] "Fri Multiple LR model R-squared value: 0.935184215065896"
```

Other Possible Models as discussed in the lecture:

Model 1 : Does not stratify by day of the week and use just T-7 as predictor

```
t7_model = lm(Actual~T...7, a)
print(paste("T-7 based LR model R-squared value:",summary(t7_model)$r.squared))
```

```
## [1] "T-7 based LR model R-squared value: 0.802419501289214"
```

Model 2 : Includes day of the week as dummy variables

```
t7d_model = lm(Actual~T...7+DOW, a)
print(paste("T-7+DOW based LR model R-squared value:",summary(t7d_model)$r.squared))
```

```
## [1] "T-7+DOW based LR model R-squared value: 0.817761391970581"
```

Model 3 : stratify by day of the week and use just T-7 as predictor

```
DOW = c("Mon","Tue","Wed","Thu","Fri")

for (i in 1:5){
  df3 = a[a[,2]==DOW[i],]
  t7s_model = lm(Actual~T...7, df3)
  print(paste(DOW[i],"stratified T-7 based LR model R-squared value:",summary(t7s_model)$r.sq
uared))

}
```

```
## [1] "Mon stratified T-7 based LR model R-squared value: 0.824985517101034"
## [1] "Tue stratified T-7 based LR model R-squared value: 0.537466514873414"
## [1] "Wed stratified T-7 based LR model R-squared value: 0.668428942481873"
## [1] "Thu stratified T-7 based LR model R-squared value: 0.615438510844795"
## [1] "Fri stratified T-7 based LR model R-squared value: 0.915886722095723"
```

Now, consider the surgery (actual) date as a time series with September 4th to September 14th as the testing set and the rest as the training set. Fit a Moving Average (MA) model to the time series and visualize it.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

```
library(magrittr)
a %<>%
  mutate(SurgDate= as.Date(SurgDate, format= "%d-%m-%Y"))
```

```
train <- subset(a, SurgDate < "2012-09-04")
test <- subset(a, SurgDate >= "2012-09-04")

tail(train)
```

```
##       SurgDate DOW T...28 T...21 T...14 T...13 T...12 T...11 T...10 T...9 T...8
## 227 2012-08-24 Fri     29     34     67     67     67     67     75    91    95
## 228 2012-08-27 Mon     40     44     66     69     79     82     85    85    86
## 229 2012-08-28 Tue     34     56     69     84     91     94     94    94    99
## 230 2012-08-29 Wed     36     57     76     81     87     87     87    92    99
## 231 2012-08-30 Thu     29     59     86     88     88     88     97   102   105
## 232 2012-08-31 Fri     19     38     58     58     58     62     68    71    80
##     T...7 T...6 T...5 T...4 T...3 T...2 T...1 Actual
## 227   104   104   104   108   115   119   126    126
## 228    92    98   107   109   111   116   123    127
## 229   103   110   119   124   125   128   139    139
## 230   101   102   104   103   103   107   114    125
## 231   106   112   113   113   113   115   124    126
## 232    86    86    86    94    93    99   116    124
```

```
library(zoo)
```

```
##
## Attaching package: 'zoo'
```

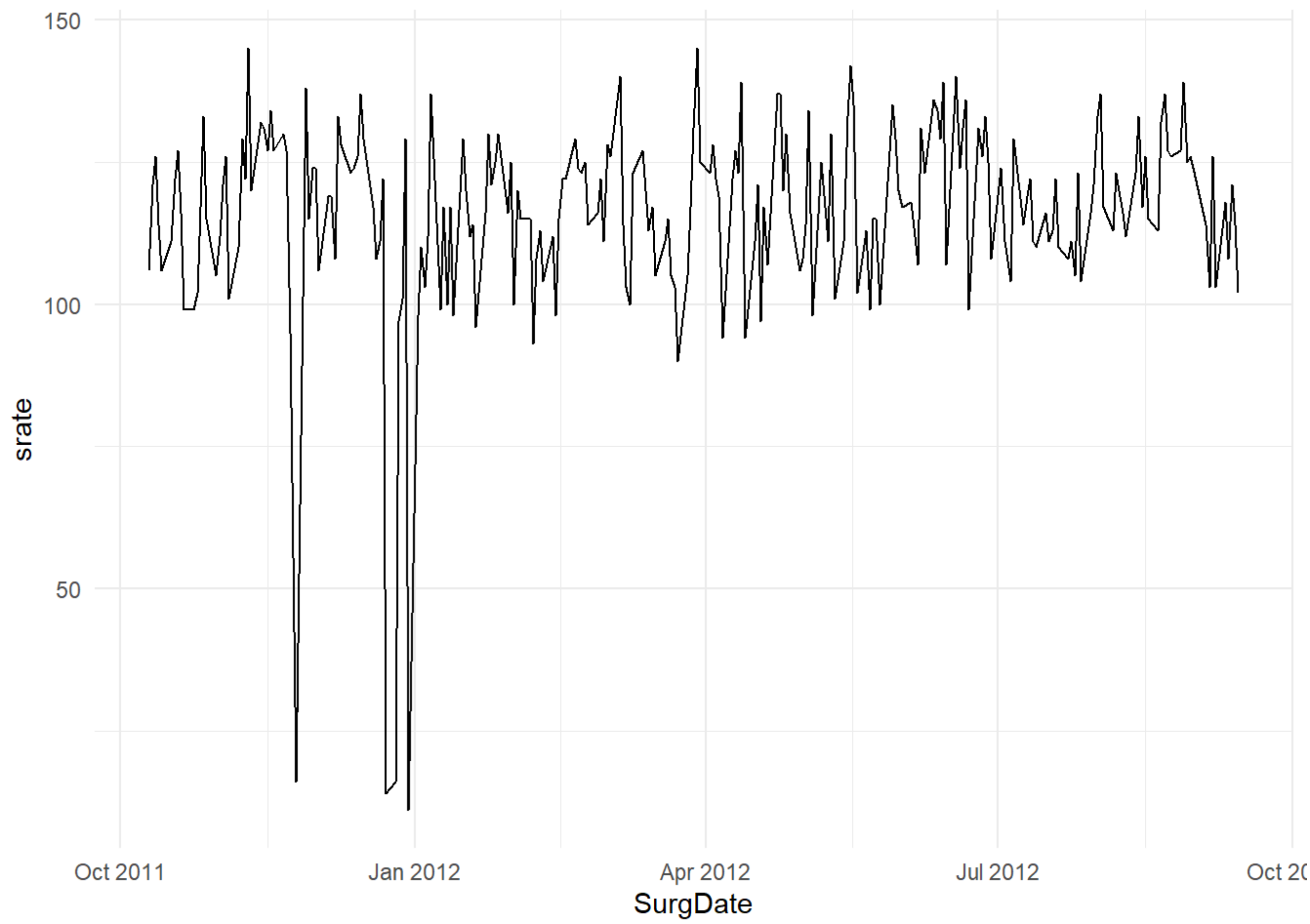
```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
s <- a %>%
  select(SurgDate, srate = Actual) %>%
  mutate(srate_tma = rollmean(srate, k = 3, fill = NA, align = "right"))
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.3
```

```
ggplot(s, aes(SurgDate,srate)) +
  geom_line() +
  theme_minimal()
```




```
# let's use T-6 based model, which had given us least MSE of prediction on test data
summary(t6_model)
```

```
##
## Call:
## lm(formula = Actual ~ T...6, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -21.711  -5.370  -0.689   5.068  22.534
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 31.76291    2.84760   11.15   <2e-16 ***
## T...6        0.94823    0.03167   29.94   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.003 on 197 degrees of freedom
## Multiple R-squared:  0.8198, Adjusted R-squared:  0.8189
## F-statistic: 896.4 on 1 and 197 DF,  p-value: < 2.2e-16
```

```r
print(paste("T-6 based model MSE:",mean((test$Actual - predict.lm(t6_model, test)) ^ 2)))
```

```
## [1] "T-6 based model MSE: 28.2909188554732"
```