Titanic Survivor Prediction

David (Yongbock) Kwon

========== Titanic Survivor Classification Prediction ==========

Importing and Manipulating Data - Feature Engineering

```
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(ggplot2)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
library(rpart)
library(rpart.plot)
library(caret)
```

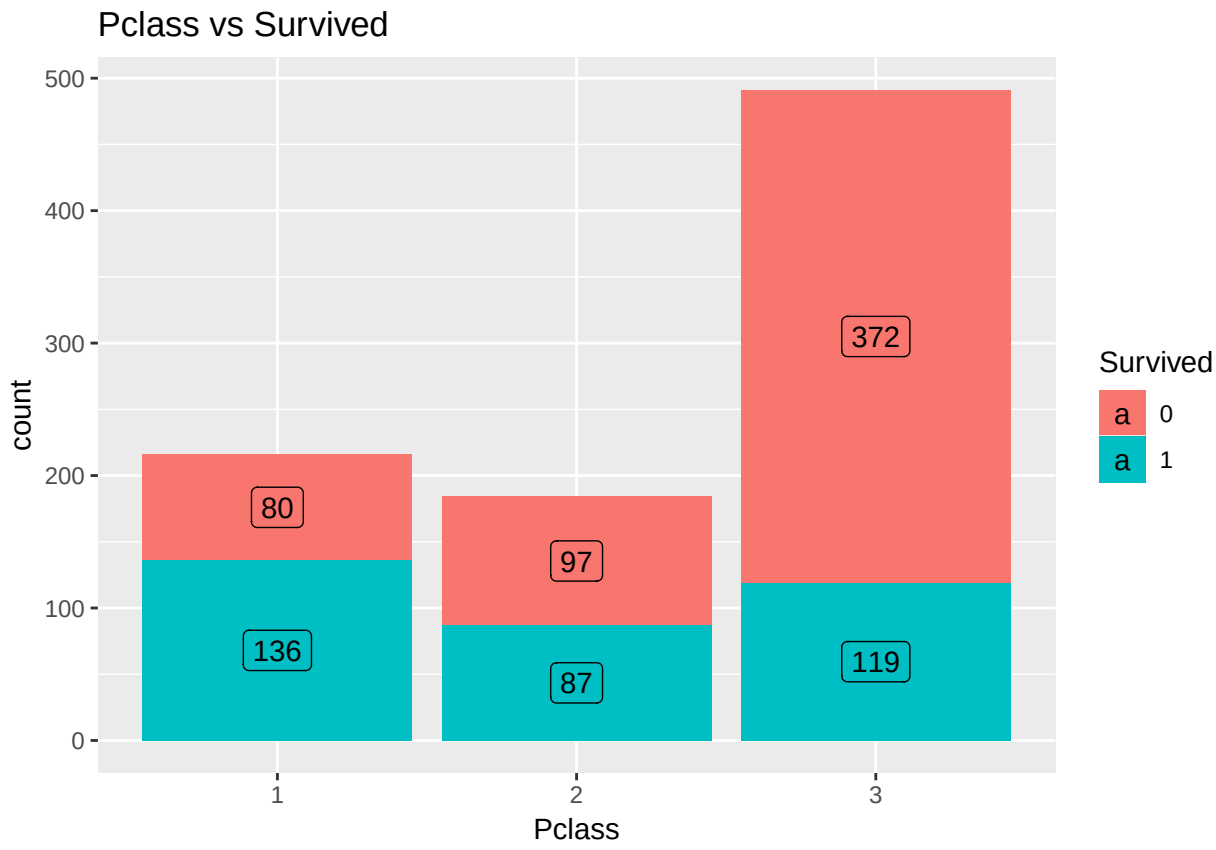
```
## Loading required package: lattice
```

```
#train and test
train <- read.csv("Datasets/train.csv", stringsAsFactors = TRUE, na.strings = "")
test <- read.csv("Datasets/test.csv", stringsAsFactors = TRUE, na.strings = "")


#creating survived variables in test set and combinining train and test
test$Survived <- NA
dat <- rbind(train,test)
```

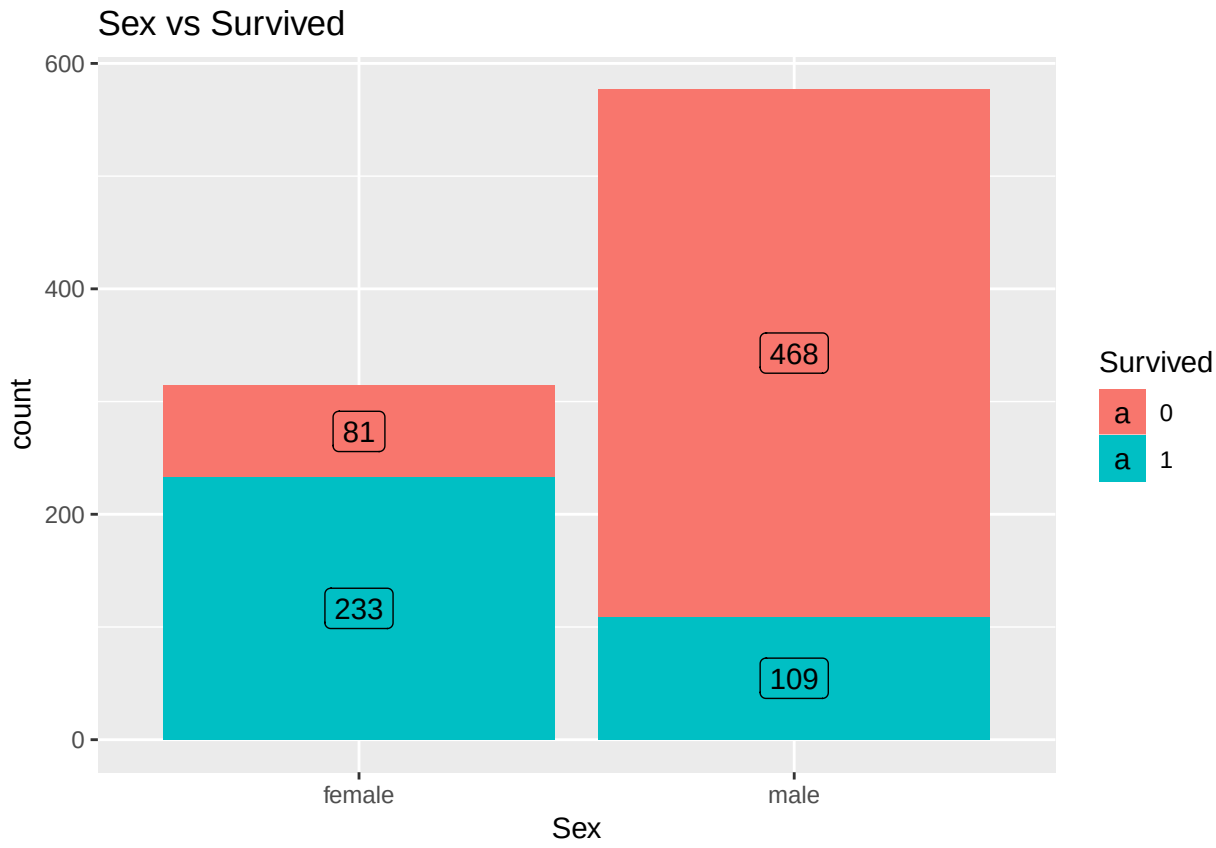
Survived and Pclass

```
#convert survived and pclass to factor variable
dat$Survived <- as.factor(dat$Survived)
dat$Pclass <- as.factor(dat$Pclass)
#Survived : 1 / no Survived : 0

#Bar graph for Pclass vs Survived
dat %>% filter(!is.na(Survived)) %>%
```

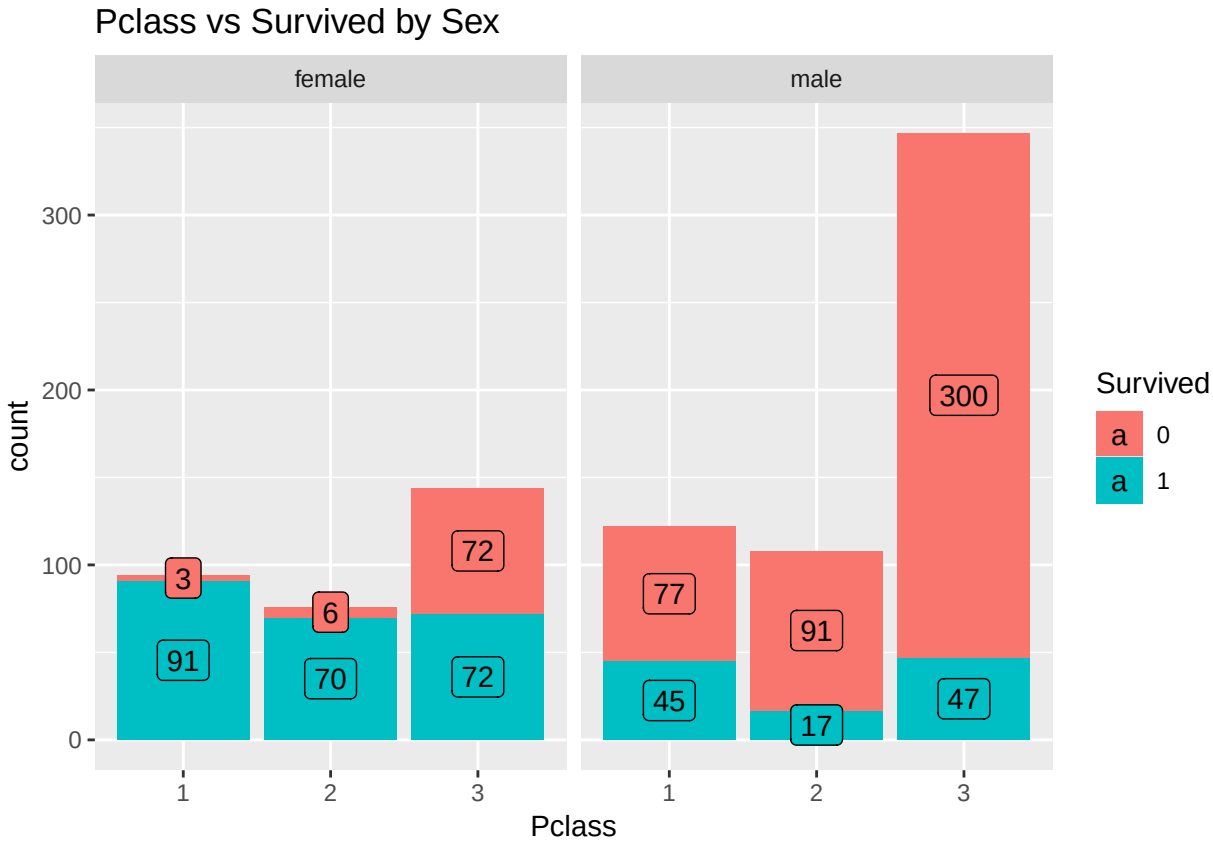
```
  ggplot(aes(x=Pclass, fill=Survived))+
  geom_bar()+
  geom_label(stat="count",
             position=position_stack(0.5),
             aes(label=..count..))+
  ggtitle("Pclass vs Survived")
```




```
#Bar graph for Sex vs Survived
dat %>% filter(!is.na(Survived)) %>%
  ggplot(aes(x=Sex, fill=Survived))+
  geom_bar()+
  geom_label(stat="count",
             position=position_stack(0.5),
             aes(label=..count..))+
  ggtitle("Sex vs Survived")
```




```
dat %>% filter(!is.na(Survived)) %>%
  ggplot(aes(x=Pclass, fill=Survived))+
  geom_bar()+
  geom_label(stat="count",
             position=position_stack(0.5),
             aes(label=..count..))+
  ggtitle("Pclass vs Survived by Sex")+
  facet_grid(~Sex)
```



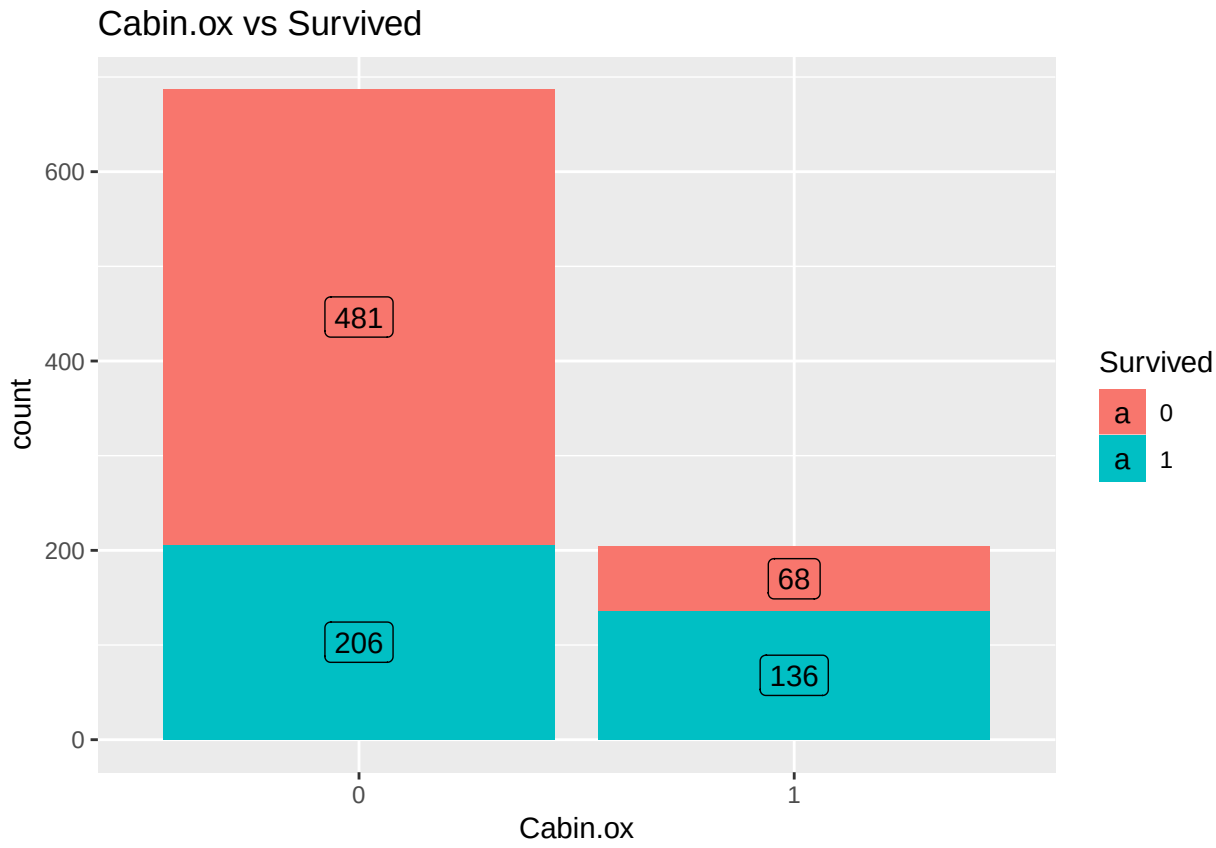

```
#In Pclass 1 and 2, obviously male mostly not survived and female survived
#In Pclass 3, male mostly not survived, but female hard to predict whether surv or not
```

From Cabin, Cabin.ox

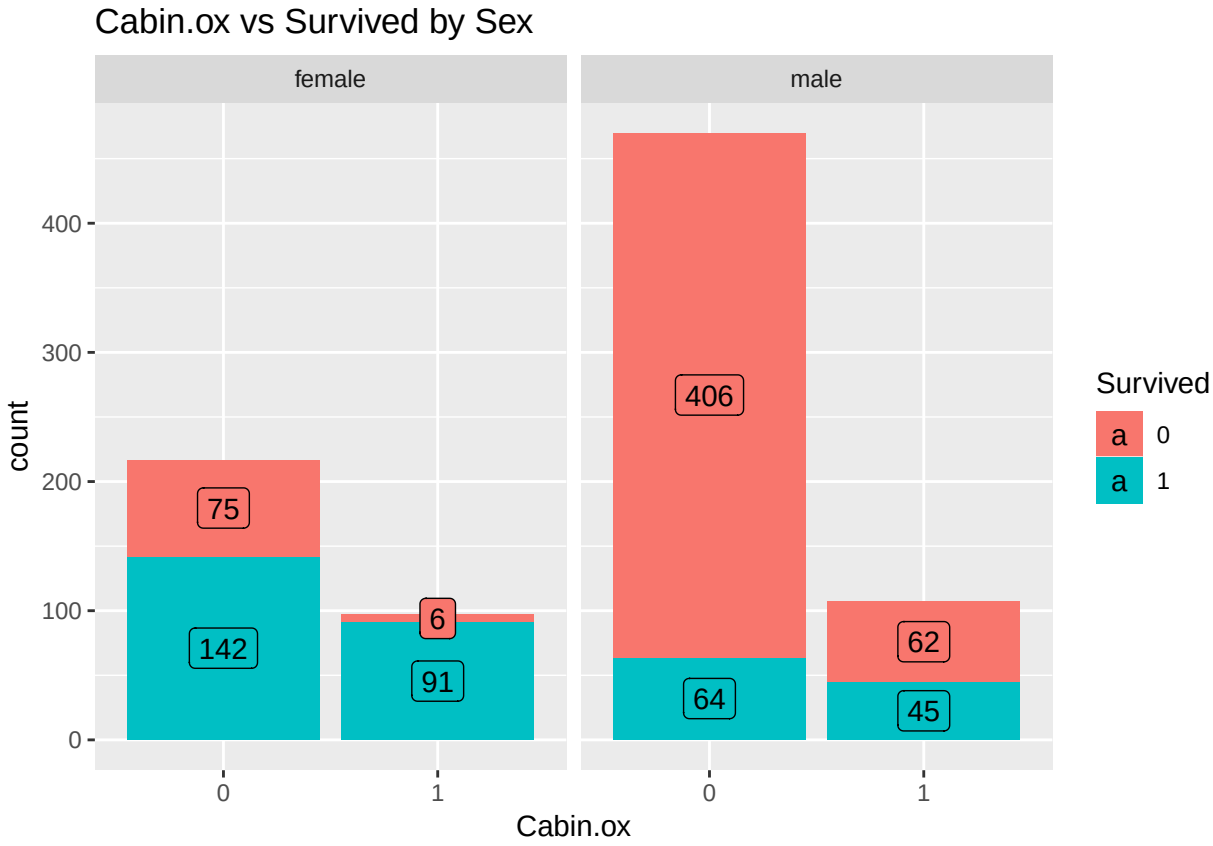
```
#Cabin NA values -> 0, otherwise 1
dat$Cabin.ox <- as.factor(ifelse(is.na(dat$Cabin), 0, 1))
table(dat$Cabin.ox)
```

```
##
##    0    1
## 1014  295
```

```
#no cabin : 0 / cabin : 1

dat %>% filter(!is.na(Survived)) %>%
  ggplot(aes(x=Cabin.ox, fill=Survived))+
  geom_bar()+
  geom_label(stat="count",
             position=position_stack(0.5),
             aes(label=..count..))+
  ggtitle("Cabin.ox vs Survived")
```

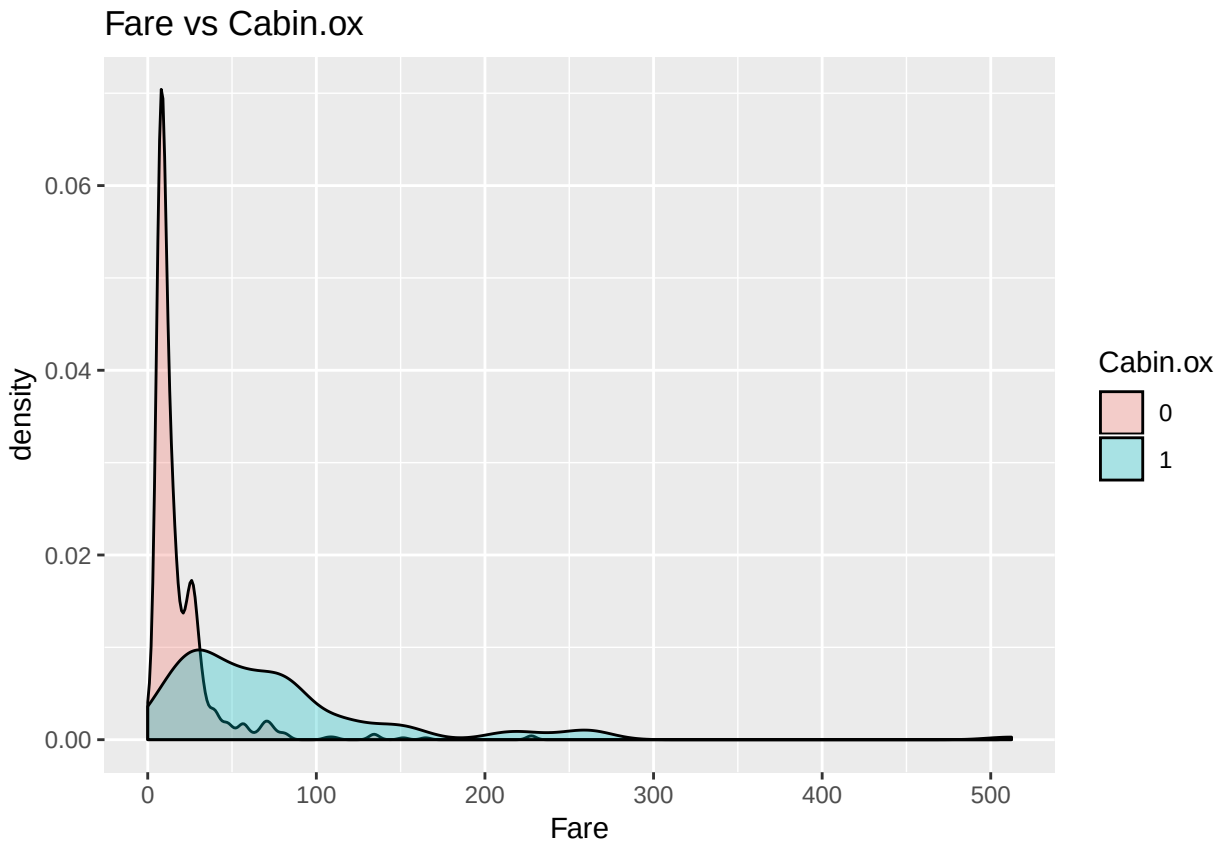



```
dat %>% filter(!is.na(Survived)) %>%
  ggplot(aes(x=Cabin.ox, fill=Survived))+
  geom_bar()+
  geom_label(stat="count",
             position=position_stack(0.5),
             aes(label=..count..))+
  ggtitle("Cabin.ox vs Survived by Sex")+
  facet_grid(~Sex)
```

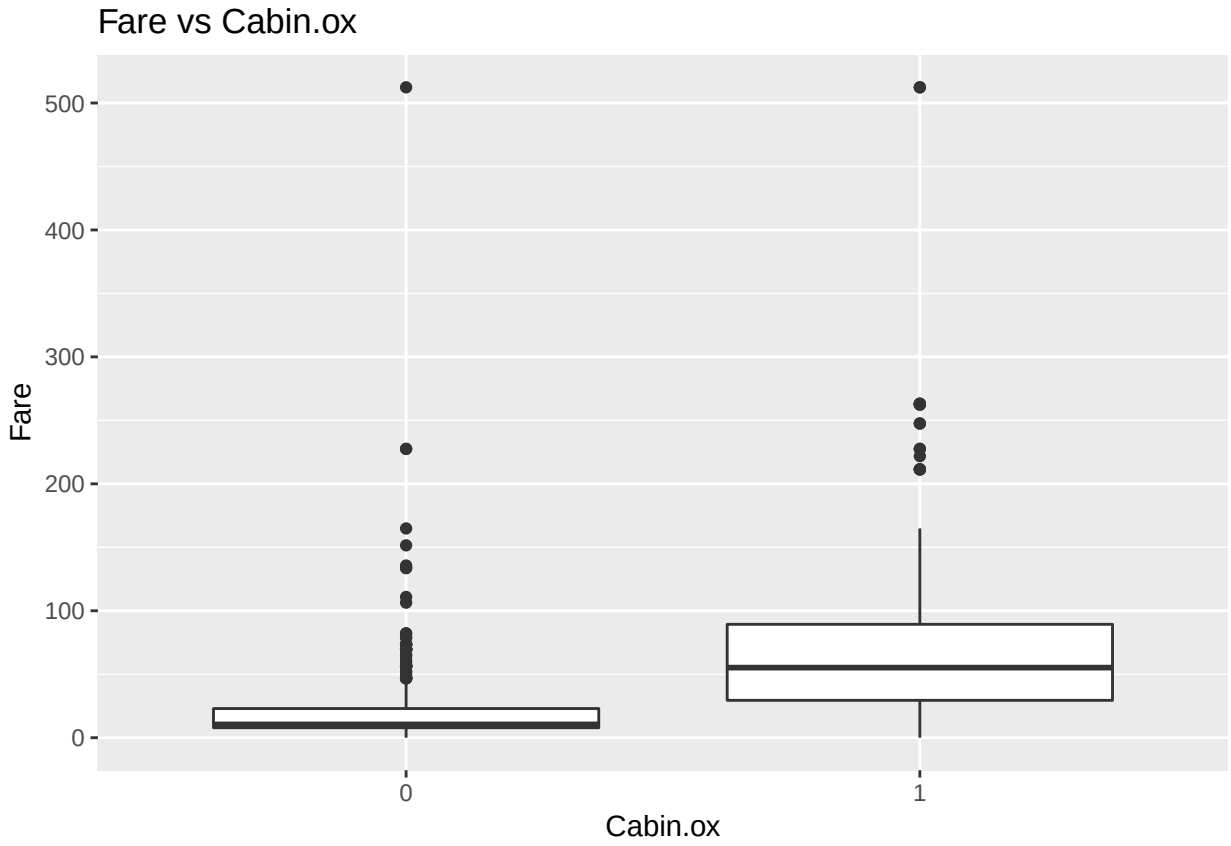



```
#If Cabin is not NA, then more likely survived
#no cabin likely not survived

dat %>% filter(!is.na(Survived)) %>%
  ggplot(aes(x=Cabin.ox, fill=Survived))+
  geom_bar()+
  geom_label(stat="count",
             position=position_stack(0.5),
             aes(label=..count..))+
  ggtitle("Cabin.ox vs Survived by Pclass")+
  facet_grid(~Pclass)
```


```
#Also, notice Pclass 1 people mostly have cabin
#Pclass 2 and 3 not

dat %>% filter(!is.na(Survived)) %>%
  ggplot(aes(x=Fare, fill=Cabin.ox))+
  geom_density(alpha=0.3)+
  ggtitle("Fare vs Cabin.ox")
```




```
dat %>% filter(!is.na(Survived)) %>%
  ggplot(aes(x=Cabin.ox, y=Fare))+
  geom_boxplot()+
  ggtitle("Fare vs Cabin.ox")
```




```
#Fare difference by Cabin.ox
```

Function to make prop.table

```
#creating function to make prop.table

prop.func <- function(predictor){
  prop.tab <- data.frame(
    prop.table(
      matrix(
        c(table(dat[1:891,predictor], dat$Survived[1:891])[,1],
          table(dat[1:891,predictor], dat$Survived[1:891])[,2]),
        ncol=2),
      1))
  colnames(prop.tab) <- c("no surv", "surv")
  rownames(prop.tab) <- c(levels(dat[,predictor]))

  return(prop.tab)
}
```

From Cabin, deck.surv

```
#deck from Cabin
dat$deck <- as.factor(ifelse(is.na(substr(dat$Cabin,1,1)), "no", substr(dat$Cabin,1,1)))
```

```r
which(dat$deck == "T") #the element where is in traing set.. lets replace this to something else
```

```
## [1] 340
```

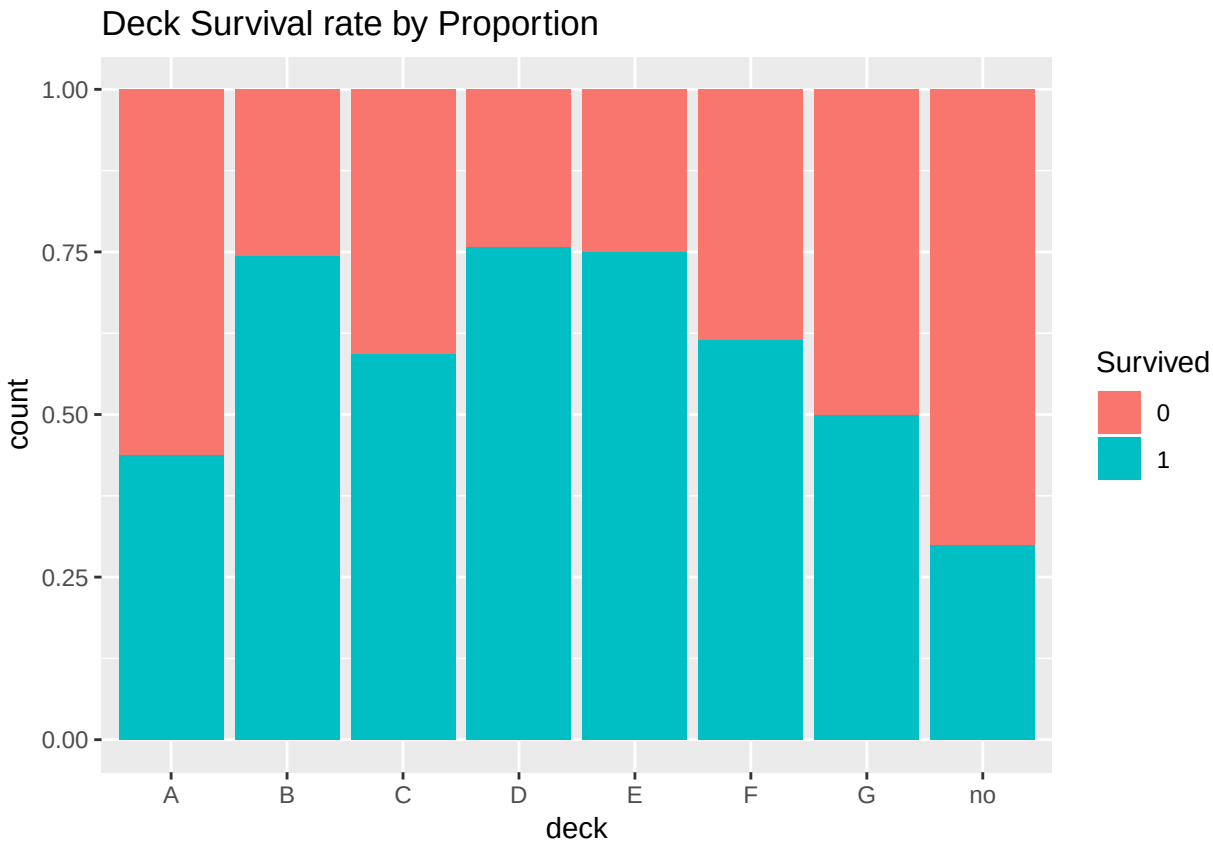
```r
dat %>%
  subset(select = -c(PassengerId)) %>%
  filter(!is.na(Survived)) %>%
  group_by(deck) %>%
  summarise(count = n(),
            mean = mean(Fare))
```

```
## # A tibble: 9 x 3
##   deck  count  mean
##   <fct> <int> <dbl>
## 1 A        15  39.6
## 2 B        47 114.
## 3 C        59 100.
## 4 D        33  57.2
## 5 E        32  46.0
## 6 F        13  18.7
## 7 G         4  13.6
## 8 no      687  19.2
## 9 T         1  35.5
```

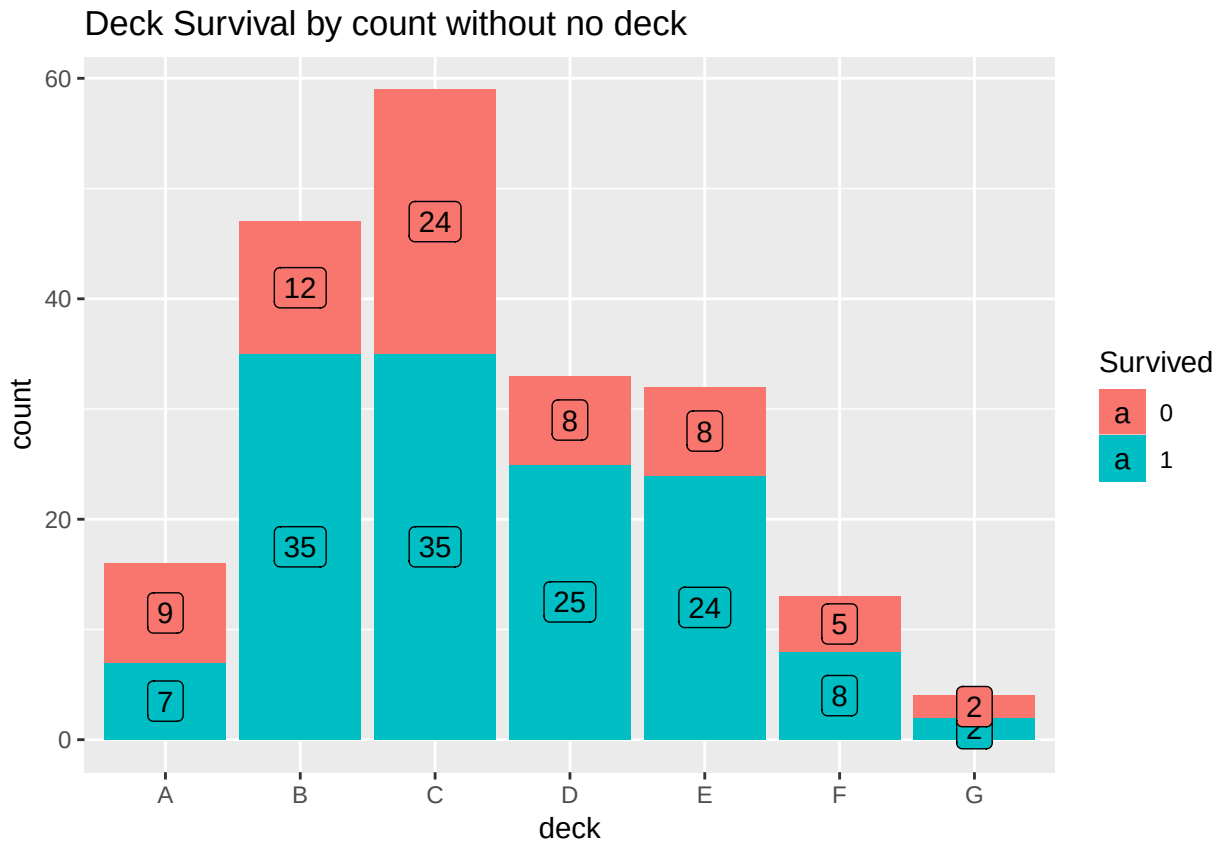
```r
#mean of Fare for deck "T" is close to the mean of Fare for deck "A"
#replace "T" to "A"

dat$deck[dat$deck=="T"] <- "A"
dat$deck <- as.factor(as.character(dat$deck))

summary(dat$deck)
```

```
##    A    B    C    D    E    F    G   no
##   23   65   94   46   41   21    5 1014
```

```r
#proportional bar graph
dat %>% filter(!is.na(Survived)) %>%
  ggplot(aes(x=deck, fill=Survived))+
  geom_bar(position = "fill")+
  ggtitle("Deck Survival rate by Proportion")
```




```
#count bar graph without no deck
dat %>% filter(!is.na(Survived) & deck != "no") %>%
  ggplot(aes(x=deck, fill=Survived)) +
  geom_bar() +
  geom_label(stat = "count",
             position = position_stack(0.5),
             aes(label= ..count..))+
  ggtitle("Deck Survival by count without no deck")
```




```
table(dat$deck[1:891], dat$Survived[1:891])
```

```
##
##        0   1
##   A    9   7
##   B   12  35
##   C   24  35
##   D    8  25
##   E    8  24
##   F    5   8
##   G    2   2
##   no 481 206
```

```
deck.prop <- prop.func("deck")

#proportional deck table
deck.prop
```

```
##      no surv      surv
## A  0.5625000 0.4375000
## B  0.2553191 0.7446809
## C  0.4067797 0.5932203
## D  0.2424242 0.7575758
## E  0.2500000 0.7500000
## F  0.3846154 0.6153846
## G  0.5000000 0.5000000
## no 0.7001456 0.2998544
```

```
#we might want to group up B/D/E together (which have high prob for survived)
#so, B/C/D/E/F -> high prob surv rate deck
#    A/G/no -> low prob surv rate
dat$deck <- as.character(dat$deck)

dat$deck.surv <- NA
for(i in 1:nrow(dat)){
  if(dat$deck[i] %in% c("B", "C", "D", "E","F")){
    dat$deck.surv[i] <- "high"
  }
  if(dat$deck[i] %in% c("no", "A", "G")){
    dat$deck.surv[i] <- "low"
  }
}

table(dat$deck.surv)
```

```
##
## high  low
##  267 1042
```

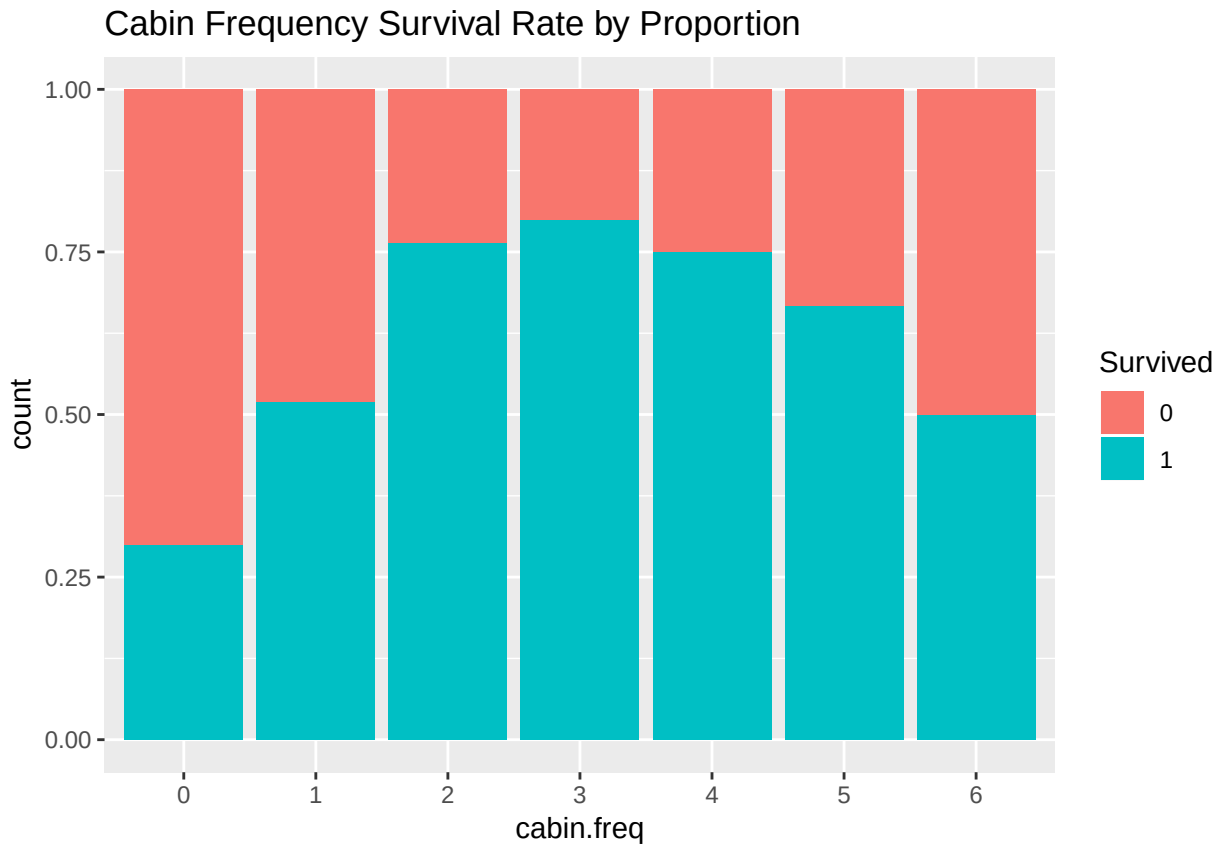
```
dat$deck.surv <- as.factor(dat$deck.surv)

dat <- dat %>% subset(select=-c(deck))
```

From Cabin, cabin.freq.surv

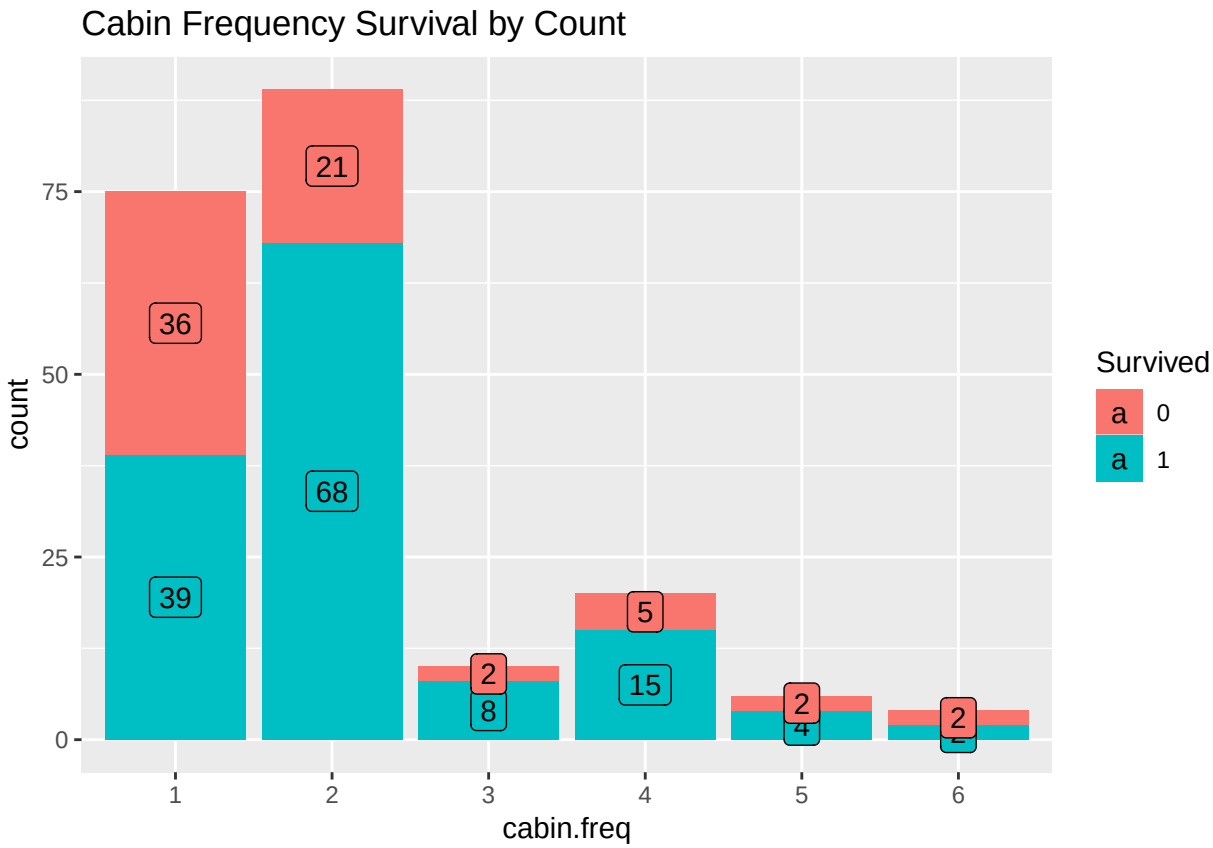
```
#cabin frequency.. might have relationship between cabin freq
cabin.freq <- data.frame(table(dat$Cabin))


dat$cabin.freq <- NA
for(i in 1:nrow(dat)){
  if(dat$Cabin[i] %in% cabin.freq$Var1){
    dat$cabin.freq[i] <- cabin.freq$Freq[cabin.freq$Var1==dat$Cabin[i]]
  }
  else{
    dat$cabin.freq[i] <- 0
  }
}
dat$cabin.freq <- as.factor(dat$cabin.freq)
summary(dat$cabin.freq)
```

```
##    0    1    2    3    4    5    6
## 1014  107  126   18   28   10    6
```

```
#proportional bar graph
dat %>% filter(!is.na(Survived)) %>%
  ggplot(aes(x=cabin.freq, fill=Survived)) +
  geom_bar(position = "fill")+
  ggtitle("Cabin Frequency Survival Rate by Proportion")
```




```
#bar graph without no cabin
dat %>% filter(!is.na(Survived) & cabin.freq != 0) %>%
  ggplot(aes(x=cabin.freq, fill=Survived)) +
  geom_bar() +
  geom_label(stat = "count",
             position = position_stack(0.5),
             aes(label= ..count..))+
  ggtitle("Cabin Frequency Survival by Count")
```




```
table(dat$cabin.freq[1:891], dat$Survived[1:891])
```

```
##
##       0   1
##   0 481 206
##   1  36  39
##   2  21  68
##   3   2   8
##   4   5  15
##   5   2   4
##   6   2   2
```

```
cabin.freq.prop <- prop.func("cabin.freq")


cabin.freq.prop
```

```
##     no surv      surv
## 0 0.7001456 0.2998544
## 1 0.4800000 0.5200000
## 2 0.2359551 0.7640449
## 3 0.2000000 0.8000000
## 4 0.2500000 0.7500000
## 5 0.3333333 0.6666667
## 6 0.5000000 0.5000000
```

```
#no cabin barely survived
#cabin freq 1 / 2 / 3 / 4 / 5 more likely surv
```

```
#no cabin , cabin freq 6 -> low
#cabin freq 1,2,3,4,5    -> high

dat$cabin.freq.surv <- NA

for(i in 1:nrow(dat)){
  if(dat$cabin.freq[i] %in% c(1,2,3,4,5)){
    dat$cabin.freq.surv[i] <- "high"
  }
  if(dat$cabin.freq[i] %in% c(0,6)){
    dat$cabin.freq.surv[i] <- "low"
  }
}

dat$cabin.freq.surv <- as.factor(dat$cabin.freq.surv)
table(dat$cabin.freq.surv)
```

```
##
## high  low
##  289 1020
```

```
dat <- subset(dat, select = -c(Cabin, cabin.freq))
```

Dealing with NA values in Embarked and Fare

```
#Gender -> male = 0, female = 1
dat$Sex <- as.factor(ifelse(dat$Sex == "male", 0, 1))

dat[is.na(dat$Embarked),]
```

```
##     PassengerId Survived Pclass                                      Name
## 62           62        1      1                      Icard, Miss. Amelie
## 830         830        1      1 Stone, Mrs. George Nelson (Martha Evelyn)
##     Sex Age SibSp Parch Ticket Fare Embarked Cabin.ox deck.surv
## 62    1  38     0     0 113572   80     <NA>        1      high
## 830   1  62     0     0 113572   80     <NA>        1      high
##     cabin.freq.surv
## 62             high
## 830            high
```

```
#Pclass = 1 / Sex = Female / have cabin /
#deck surv rate high / cabin freq surv rate high
dat %>%
  filter(Pclass == 1 &
           Sex == 1 &
           Cabin.ox==1 &
           deck.surv == "high" &
           cabin.freq.surv == "high" &
           SibSp == 0 &
           Parch == 0) %>% group_by(Embarked) %>%
  summarise(count = n(),
            mean = mean(Fare),
            min = min(Fare),
```

```
            max = max(Fare))
```

```
## Warning: Factor `Embarked` contains implicit NA, consider using
## `forcats::fct_explicit_na`
```

```
## # A tibble: 3 x 5
##   Embarked count  mean   min   max
##   <fct>    <int> <dbl> <dbl> <dbl>
## 1 C           18  113.  27.7  262.
## 2 S           14  102.  25.9  222.
## 3 <NA>         2   80   80     80
```

```
#Na value for Embarked
dat$Embarked[is.na(dat$Embarked)] <- "C"

dat[is.na(dat$Fare),]
```

```
##      PassengerId Survived Pclass               Name Sex  Age SibSp Parch
## 1044        1044     <NA>      3 Storey, Mr. Thomas   0 60.5     0     0
##      Ticket Fare Embarked Cabin.ox deck.surv cabin.freq.surv
## 1044   3701   NA        S        0       low             low
```

```
summary(aov(Fare~Cabin.ox, dat))
```

```
##               Df  Sum Sq Mean Sq F value Pr(>F)
## Cabin.ox       1  900931  900931   452.5 <2e-16 ***
## Residuals   1306 2600469    1991
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 1 observation deleted due to missingness
```

```
summary(aov(Fare~Pclass, dat))
```

```
##               Df  Sum Sq Mean Sq F value Pr(>F)
## Pclass         2 1272986  636493   372.7 <2e-16 ***
## Residuals   1305 2228414    1708
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 1 observation deleted due to missingness
```

```
#NA value for Fare
dat[dat$Pclass == 3,] %>%
  group_by(Embarked, Cabin.ox, Pclass) %>%
  summarise(mean = mean(Fare, na.rm=TRUE))
```

```
## # A tibble: 6 x 4
## # Groups:   Embarked, Cabin.ox [6]
##   Embarked Cabin.ox Pclass  mean
##   <fct>    <fct>    <fct>  <dbl>
## 1 C        0        3      11.0
## 2 C        1        3      12.3
## 3 Q        0        3      10.4
## 4 Q        1        3       7.75
## 5 S        0        3      14.5
## 6 S        1        3      11.2
```

```
#Pclass 3 / Embarked S / no cabin
#mean of Pclass 3 and Embarked S, and no cabin is 14.5
```

```
dat$Fare[is.na(dat$Fare)] <- 14.5
```

From Ticket, ticket.alone

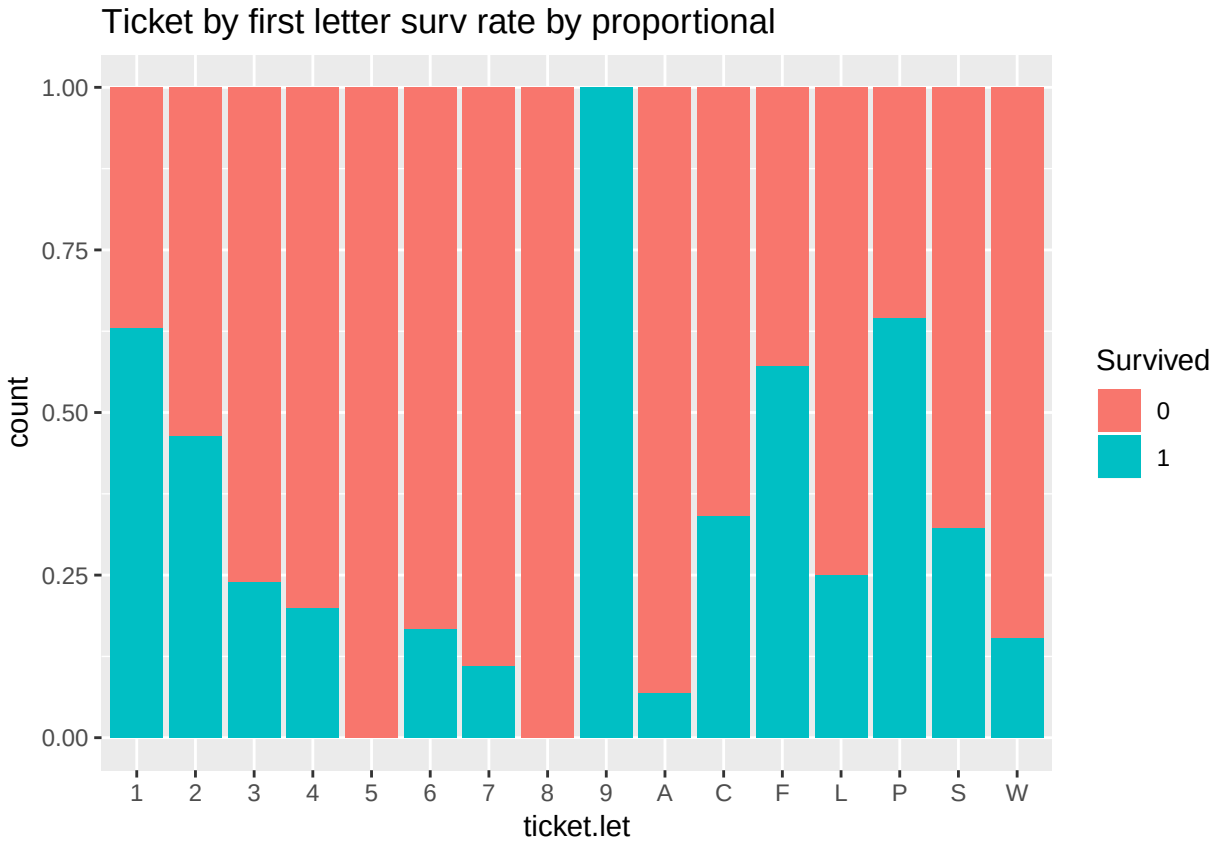
```
#Ticket
ticket.alone <- data.frame(table(dat$Ticket))

dat$ticket.alone <- NA
for(i in 1:nrow(dat)){
  if(dat$Ticket[i] %in% ticket.alone$Var1[ticket.alone$Freq==1]){
    dat$ticket.alone[i] <- 0
  }
  if(dat$Ticket[i] %in% ticket.alone$Var1[ticket.alone$Freq>1]){
    dat$ticket.alone[i] <- 1
  }
}

table(dat$ticket.alone)
```

```
##
##   0   1
## 713 596
```

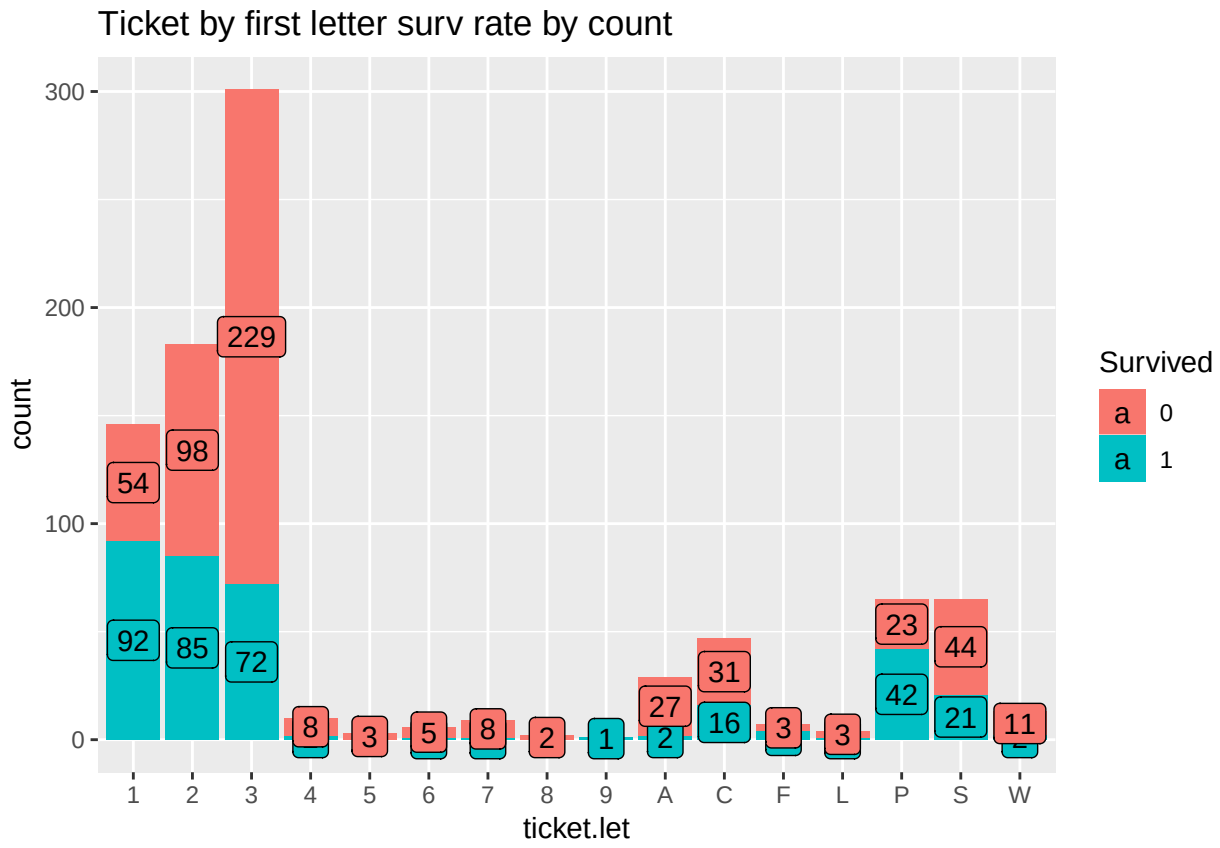
```
dat$ticket.alone <- as.factor(dat$ticket.alone)
```

From Ticket, ticket.let.surv

```
#ticket by first letter
dat$ticket.let <- substr(dat$Ticket, 1,1)

ticket.let <- data.frame(table(dat$ticket.let))


#proportional bar graph
dat %>% filter(!is.na(Survived) ) %>%
  ggplot(aes(x=ticket.let, fill=Survived)) +
  geom_bar(position = "fill")+
  ggtitle("Ticket by first letter surv rate by proportional")
```




```
dat %>% filter(!is.na(Survived) ) %>%
  ggplot(aes(x=ticket.let, fill=Survived)) +
  geom_bar() +
  geom_label(stat = "count",
             position = position_stack(0.5),
             aes(label= ..count..))+
  ggtitle("Ticket by first letter surv rate by count")
```




```
table(dat$ticket.let[1:891], dat$Survived[1:891])
```

```
##
##       0   1
##   1  54  92
##   2  98  85
##   3 229  72
##   4   8   2
##   5   3   0
##   6   5   1
##   7   8   1
##   8   2   0
##   9   0   1
##   A  27   2
##   C  31  16
##   F   3   4
##   L   3   1
##   P  23  42
##   S  44  21
##   W  11   2
```

```
dat$ticket.let <- as.factor(dat$ticket.let)

ticket.let.prop <- prop.func("ticket.let")
ticket.let.prop
```

```
##     no surv       surv
## 1 0.3698630 0.63013699
```

```
## 2 0.5355191 0.46448087
## 3 0.7607973 0.23920266
## 4 0.8000000 0.20000000
## 5 1.0000000 0.00000000
## 6 0.8333333 0.16666667
## 7 0.8888889 0.11111111
## 8 1.0000000 0.00000000
## 9 0.0000000 1.00000000
## A 0.9310345 0.06896552
## C 0.6595745 0.34042553
## F 0.4285714 0.57142857
## L 0.7500000 0.25000000
## P 0.3538462 0.64615385
## S 0.6769231 0.32307692
## W 0.8461538 0.15384615
```

```
dat$ticket.let <- as.factor(dat$ticket.let)

die <- rownames(ticket.let.prop[ticket.let.prop$`no surv`>=0.5,])
surv <- rownames(ticket.let.prop[ticket.let.prop$`no surv`<0.5,])

dat$ticket.let <- as.character(dat$ticket.let)
dat$ticket.let.surv <- NA
for(i in 1:nrow(dat)){
  if(dat$ticket.let[i] %in% die){
    dat$ticket.let.surv[i] <- "low"
  }
  if(dat$ticket.let[i] %in% surv){
    dat$ticket.let.surv[i] <- "high"
  }
}

dat$ticket.let.surv <- as.factor(dat$ticket.let.surv)
summary(dat$ticket.let.surv)
```

```
## high  low
##  323  986
```

```
dat <- dat %>% subset(select =-c(Ticket, ticket.let))
```

Creating family variable

```
#family size (if family = 1, then it's alone)
dat$family <- dat$SibSp + dat$Parch + 1
#1 == alone

dat <- subset(dat, select = -c(SibSp, Parch))
```

From Name, name and surname.freq.surv Dealing with NA values in Age ——————————

```
#converting names
dat <- dat %>%
  mutate(name = sub("\\..*$", "", sub("^.*, ", "", Name)),
         surname = sub(",.*$", "", Name))
```

```
summary(as.factor(dat$name))
```

```
##        Capt         Col         Don        Dona          Dr
##           1           4           1           1           8
##    Jonkheer        Lady       Major      Master        Miss
##           1           1           2          61         260
##        Mlle         Mme          Mr         Mrs          Ms
##           2           1         757         197           2
##         Rev         Sir the Countess
##           8           1           1
```

```
summary(as.factor(dat$surname))
```

```
##     Andersson          Sage       Asplund       Goodwin        Davies
##            11            11             8             8             7
##         Brown        Carter          Ford       Fortune       Johnson
##             6             6             6             6             6
##        Panula          Rice         Skoog         Smith         Kelly
##             6             6             6             6             5
##       Lefebre       Palsson       Ryerson        Thomas      Williams
##             5             5             5             5             5
##       Allison       Baclini        Becker        Boulos         Cacic
##             4             4             4             4             4
##          Dean         Elias     Goldsmith    Gustafsson        Hansen
##             4             4             4             4             4
##        Harper        Harris          Hart        Herman       Hocking
##             4             4             4             4             4
##     Johansson      Johnston       Laroche         Olsen Vander Planke
##             4             4             4             4             4
##          Ware          West        Abbott        Bourke      Caldwell
##             4             4             3             3             3
##      Carlsson       Chapman       Collyer       Compton           Cor
##             3             3             3             3             3
##        Coutts        Crosby          Daly        Danbom         Dodge
##             3             3             3             3             3
##       Douglas          Drew         Flynn    Frauenthal         Giles
##             3             3             3             3             3
##        Graham          Hays       Hickman        Howard          Hoyt
##             3             3             3             3             3
##        Jensen       Jussila      Karlsson         Keane Kink-Heilmann
##             3             3             3             3             3
##        Klasen        Mallet         McCoy         Meyer       Minahan
##             3             3             3             3             3
##         Moran      Moubarek        Murphy         Nakid      Navratil
##             3             3             3             3             3
##        Newell       Nilsson       O'Brien        Olsson     Oreskovic
##             3             3             3             3             3
##       Peacock         Peter      Phillips         Quick      Richards
##             3             3             3             3             3
##       Rosblom        Samaan     Sandstrom       Spedden      Svensson
##             3             3             3             3             3
##       Taussig        Thayer         Touma  van Billiard       (Other)
##             3             3             3             3           921
```

```r
#name first

dat %>%
  group_by(name, Sex) %>%
  summarise(mean = mean(Age, na.rm=TRUE),
            min = min(Age, na.rm=TRUE),
            max = max(Age, na.rm=TRUE),
            count = n())
```

```
## # A tibble: 19 x 6
## # Groups:   name [18]
##    name         Sex    mean   min   max count
##    <chr>        <fct> <dbl> <dbl> <dbl> <int>
##  1 Capt         0     70    70     70       1
##  2 Col          0     54    47     60       4
##  3 Don          0     40    40     40       1
##  4 Dona         1     39    39     39       1
##  5 Dr           0     42.7  23     54       7
##  6 Dr           1     49    49     49       1
##  7 Jonkheer     0     38    38     38       1
##  8 Lady         1     48    48     48       1
##  9 Major        0     48.5  45     52       2
## 10 Master       0      5.48  0.33  14.5    61
## 11 Miss         1     21.8   0.17  63     260
## 12 Mlle         1     24    24     24       2
## 13 Mme          1     24    24     24       1
## 14 Mr           0     32.3  11     80     757
## 15 Mrs          1     37.0  14     76     197
## 16 Ms           1     28    28     28       2
## 17 Rev          0     41.2  27     57       8
## 18 Sir          0     49    49     49       1
## 19 the Countess 1     33    33     33       1
```

```r
#Master / Miss / Mr / Mrs
#Matser seems obvious young male
#Mr teenage to old male
#Miss and Mrs female in range young to old

#Age first.. to predict name by age

dat %>% filter(is.na(Age)) %>% group_by(name,Sex) %>% tally()
```

```
## # A tibble: 6 x 3
## # Groups:   name [6]
##   name   Sex       n
##   <chr>  <fct> <int>
## 1 Dr     0         1
## 2 Master 0         8
## 3 Miss   1        50
## 4 Mr     0       176
## 5 Mrs    1        27
## 6 Ms     1         1
```

```r
#dealing with Dr
dat %>% filter(name == "Dr")
```

```
##   PassengerId Survived Pclass                          Name Sex Age
## 1         246        0      1   Minahan, Dr. William Edward   0  44
## 2         318        0      2          Moraweck, Dr. Ernest   0  54
## 3         399        0      2              Pain, Dr. Alfred   0  23
## 4         633        1      1     Stahelin-Maeglin, Dr. Max   0  32
## 5         661        1      1 Frauenthal, Dr. Henry William   0  50
## 6         767        0      1     Brewe, Dr. Arthur Jackson   0  NA
## 7         797        1      1   Leader, Dr. Alice (Farnham)   1  49
## 8        1185     <NA>      1         Dodge, Dr. Washington   0  53
##       Fare Embarked Cabin.ox deck.surv cabin.freq.surv ticket.alone
## 1  90.0000        Q        1      high            high            1
## 2  14.0000        S        0       low             low            0
## 3  10.5000        S        0       low             low            0
## 4  30.5000        C        1      high            high            0
## 5 133.6500        S        0       low             low            1
## 6  39.6000        C        0       low             low            0
## 7  25.9292        S        1      high            high            0
## 8  81.8583        S        1       low            high            1
##   ticket.let.surv family name          surname
## 1            high      3   Dr          Minahan
## 2             low      1   Dr         Moraweck
## 3             low      1   Dr             Pain
## 4            high      1   Dr Stahelin-Maeglin
## 5            high      3   Dr       Frauenthal
## 6            high      1   Dr            Brewe
## 7            high      1   Dr           Leader
## 8             low      3   Dr            Dodge
```

```
dat$Age[which(dat$name == "Dr" & is.na(dat$Age))] <- mean(dat$Age[which(dat$name == "Dr")], na.rm=TRUE)

#dealing with Ms
dat %>% filter(name == "Ms")
```

```
##   PassengerId Survived Pclass                       Name Sex Age  Fare
## 1         444        1      2 Reynaldo, Ms. Encarnacion   1  28 13.00
## 2         980     <NA>      3   O'Donoghue, Ms. Bridget   1  NA  7.75
##   Embarked Cabin.ox deck.surv cabin.freq.surv ticket.alone ticket.let.surv
## 1        S        0       low             low            0             low
## 2        Q        0       low             low            0             low
##   family name    surname
## 1      1   Ms   Reynaldo
## 2      1   Ms O'Donoghue
```

```
dat$Age[which(dat$name == "Ms" & is.na(dat$Age))] <- mean(dat$Age[which(dat$name == "Ms")], na.rm=TRUE)

dat$name <- as.character(dat$name)
dat$surname <- as.character(dat$surname)


summary(aov(Age~Pclass, dat))
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## Pclass         2  37501   18750     109 <2e-16 ***
## Residuals   1045 179788     172
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 261 observations deleted due to missingness
```

```
summary(aov(Age~name, dat))
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## name          17  65448    3850   26.11 <2e-16 ***
## Residuals   1030 151840     147
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 261 observations deleted due to missingness
```

```
#I use Pclass and name to predict NA values in Age
#replacing NA's of Age with the mean by name and Pclass, respectively

dat %>% filter(is.na(Age)) %>% group_by(name,Pclass) %>% tally()
```

```
## # A tibble: 10 x 3
## # Groups:   name [4]
##    name   Pclass     n
##    <chr>  <fct>  <int>
##  1 Master 3          8
##  2 Miss   1          1
##  3 Miss   2          2
##  4 Miss   3         47
##  5 Mr     1         27
##  6 Mr     2         13
##  7 Mr     3        136
##  8 Mrs    1         10
##  9 Mrs    2          1
## 10 Mrs    3         16
```

```
dat[dat$name %in% c("Mr", "Miss", "Mrs", "Master"),] %>%
  group_by(name, Pclass) %>%
  summarise(count = n(),
            mean = mean(Age, na.rm=TRUE),
            min = min(Age, na.rm=TRUE),
            max = max(Age, na.rm=TRUE))
```

```
## # A tibble: 12 x 6
## # Groups:   name [4]
##    name   Pclass count  mean   min   max
##    <chr>  <fct>  <int> <dbl> <dbl> <dbl>
##  1 Master 1          5  6.98  0.92  13
##  2 Master 2         11  2.76  0.67   8
##  3 Master 3         45  6.09  0.33  14.5
##  4 Miss   1         60 30.3   2     63
##  5 Miss   2         50 20.7   0.92  50
##  6 Miss   3        150 17.4   0.17  45
##  7 Mr     1        159 41.5  17     80
##  8 Mr     2        150 32.3  14     70
##  9 Mr     3        448 28.3  11     74
## 10 Mrs    1         77 43.2  17     76
## 11 Mrs    2         55 33.5  14     60
## 12 Mrs    3         65 32.3  15     63
```

```
for(i in 1:nrow(dat)){
  if(is.na(dat$Age[i])){
    #Master
    if(dat$name[i] == "Master" & dat$Pclass[i] == 3){
      dat$Age[i] <- mean(dat$Age[which(dat$name == "Master" & dat$Pclass == 3)], na.rm=TRUE)
    }

    #Miss
    if(dat$name[i] == "Miss" & dat$Pclass[i] == 1){
      dat$Age[i] <- mean(dat$Age[which(dat$name == "Miss" & dat$Pclass == 1)], na.rm=TRUE)
    }
    if(dat$name[i] == "Miss" & dat$Pclass[i] == 2){
      dat$Age[i] <- mean(dat$Age[which(dat$name == "Miss" & dat$Pclass == 2)], na.rm=TRUE)
    }
    if(dat$name[i] == "Miss" & dat$Pclass[i] == 3){
      dat$Age[i] <- mean(dat$Age[which(dat$name == "Miss" & dat$Pclass == 3)], na.rm=TRUE)
    }

    #Mr
    if(dat$name[i] == "Mr" & dat$Pclass[i] == 1){
      dat$Age[i] <- mean(dat$Age[which(dat$name == "Mr" & dat$Pclass == 1)], na.rm=TRUE)
    }
    if(dat$name[i] == "Mr" & dat$Pclass[i] == 2){
      dat$Age[i] <- mean(dat$Age[which(dat$name == "Mr" & dat$Pclass == 2)], na.rm=TRUE)
    }
    if(dat$name[i] == "Mr" & dat$Pclass[i] == 3){
      dat$Age[i] <- mean(dat$Age[which(dat$name == "Mr" & dat$Pclass == 3)], na.rm=TRUE)
    }

    #Mrs
    if(dat$name[i] == "Mrs" & dat$Pclass[i] == 1){
      dat$Age[i] <- mean(dat$Age[which(dat$name == "Mrs" & dat$Pclass == 1)], na.rm=TRUE)
    }
    if(dat$name[i] == "Mrs" & dat$Pclass[i] == 2){
      dat$Age[i] <- mean(dat$Age[which(dat$name == "Mrs" & dat$Pclass == 2)], na.rm=TRUE)
    }
    if(dat$name[i] == "Mrs" & dat$Pclass[i] == 3){
      dat$Age[i] <- mean(dat$Age[which(dat$name == "Mrs" & dat$Pclass == 3)], na.rm=TRUE)
    }

    #Ms
    if(dat$name[i] == "Ms" & dat$Pclass[i] == 3){
      dat$Age[i] <- mean(dat$Age[which(dat$name == "Ms" & dat$Pclass == 3)], na.rm=TRUE)
    }
  }
}


#dealing with other names
dat$name[!dat$name %in% c("Mr", "Miss", "Mrs", "Master") ]
```

```
##  [1] "Don"          "Rev"          "Rev"          "Dr"
##  [5] "Rev"          "Dr"           "Mme"          "Dr"
##  [9] "Ms"           "Major"        "Major"        "Lady"
```

```
## [13] "Sir"          "Rev"          "Dr"           "Mlle"
## [17] "Col"          "Dr"           "Col"          "Mlle"
## [21] "Capt"         "the Countess" "Dr"           "Dr"
## [25] "Jonkheer"     "Rev"          "Rev"          "Ms"
## [29] "Col"          "Rev"          "Rev"          "Col"
## [33] "Dr"           "Dona"
```

```
dat %>% filter(!name %in% c("Mr", "Miss", "Mrs", "Master")) %>%
  group_by(name, Sex) %>%
  summarise(count = n(),
            mean = mean(Age),
            min = min(Age, na.rm=TRUE),
            max = max(Age, na.rm=TRUE))
```

```
## # A tibble: 15 x 6
## # Groups:   name [14]
##    name         Sex   count  mean   min   max
##    <chr>        <fct> <int> <dbl> <dbl> <dbl>
##  1 Capt         0         1  70      70    70
##  2 Col          0         4  54      47    60
##  3 Don          0         1  40      40    40
##  4 Dona         1         1  39      39    39
##  5 Dr           0         7  42.8    23    54
##  6 Dr           1         1  49      49    49
##  7 Jonkheer     0         1  38      38    38
##  8 Lady         1         1  48      48    48
##  9 Major        0         2  48.5    45    52
## 10 Mlle         1         2  24      24    24
## 11 Mme          1         1  24      24    24
## 12 Ms           1         2  28      28    28
## 13 Rev          0         8  41.2    27    57
## 14 Sir          0         1  49      49    49
## 15 the Countess 1         1  33      33    33
```

```
dat[dat$name %in% c("Mr", "Miss", "Mrs", "Master"),] %>%
  group_by(name) %>%
  summarise(count = n(),
            mean = mean(Age, na.rm=TRUE),
            min = min(Age, na.rm=TRUE),
            max = max(Age, na.rm=TRUE))
```

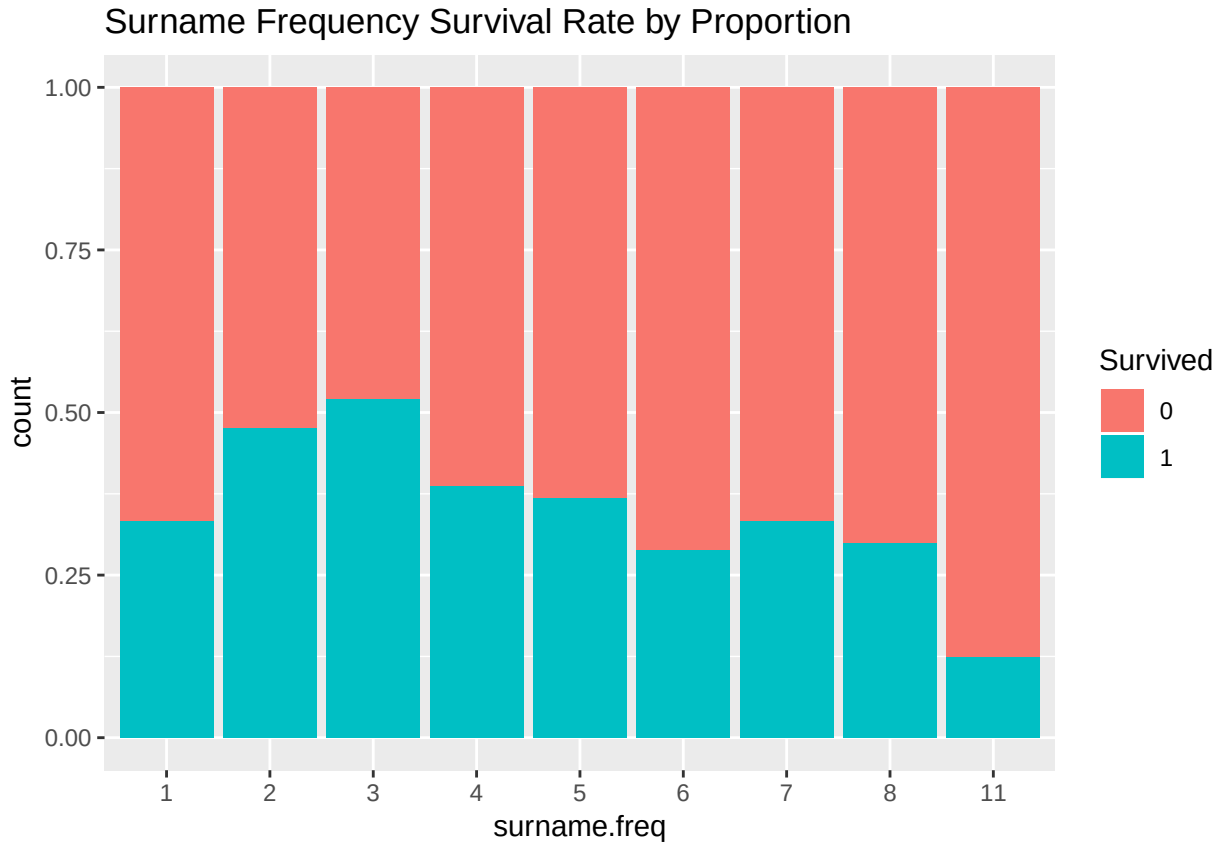
```
## # A tibble: 4 x 5
##   name   count  mean   min   max
##   <chr>  <int> <dbl> <dbl> <dbl>
## 1 Master    61  5.56  0.33  14.5
## 2 Miss     260 21.0   0.17  63
## 3 Mr       757 31.9  11     80
## 4 Mrs      197 36.9  14     76
```

```
#Master max age 14.5
#Master -> young male : sex==male & Age < 14.5
#Mr -> adult male :     sex==male & Age > 14.5
#Miss -> adult female : sex==female & Age < 14
#Mrs -> adult female :  sex==female & Age > 14

for(i in 1:nrow(dat)){
```

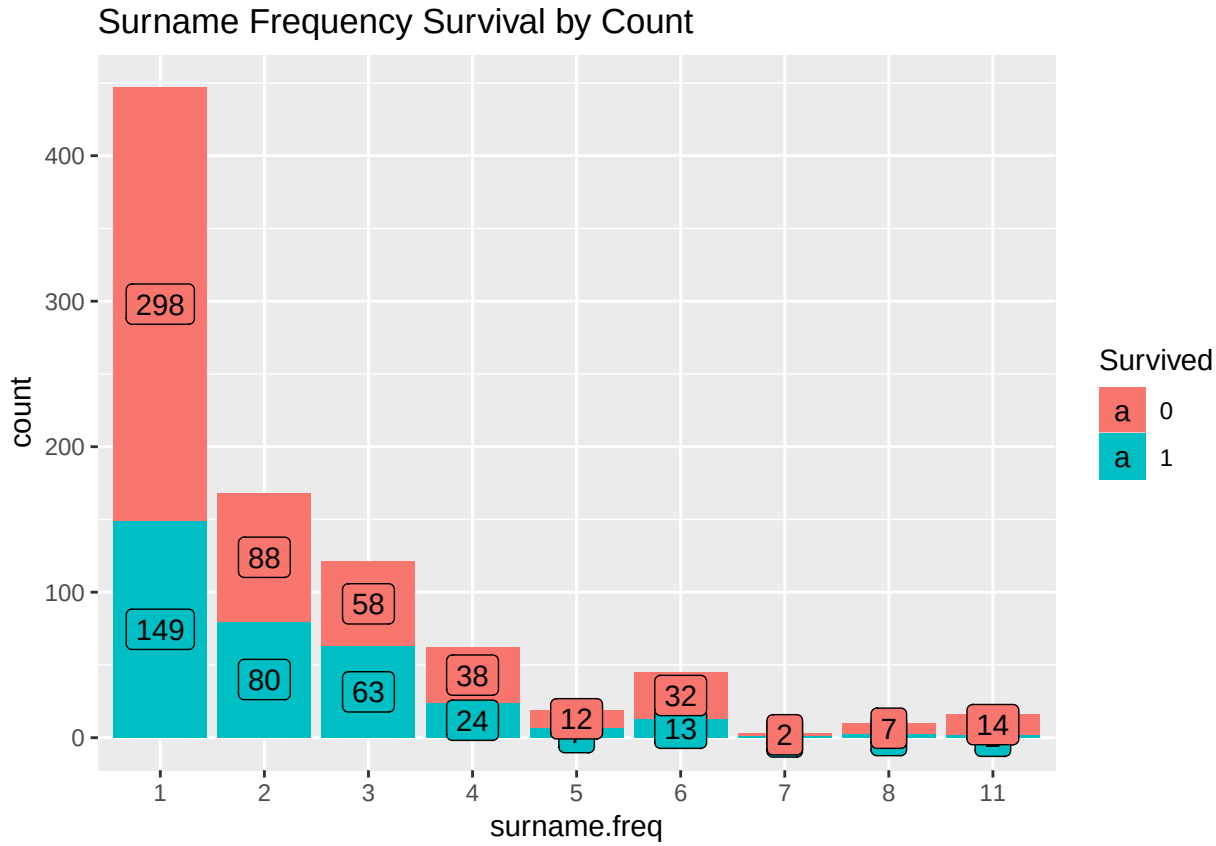
```
  if(!is.na(dat$Age[i])){
    if(!dat$name[i] %in% c("Mr", "Miss", "Mrs", "Master")){
      if(dat$Sex[i] == 0 & dat$Age[i] <= 14.5){
      dat$name[i] = "Master"
      }
      if(dat$Sex[i] == 0 & dat$Age[i] > 14.5){
      dat$name[i] <- "Mr"
      }
      if(dat$Sex[i] == 1 & dat$Age[i] < 14){
      dat$name[i] <- "Miss"
      }
      if(dat$Sex[i] == 1 & dat$Age[i] > 14){
      dat$name[i] <- "Mrs"
      }
    }
  }
}


dat$name <- as.factor(as.character(dat$name))

table(dat$name)
```

```
##
## Master   Miss     Mr    Mrs
##     61    260    782    206
```

```
#surname frequency
surname.freq <- data.frame(table(dat$surname))

dat$surname.freq <-NA

for(i in 1:nrow(dat)){
  for(j in 1:11){
    if(dat$surname[i] %in% surname.freq$Var1[surname.freq$Freq == j]){
      dat$surname.freq[i] <- j
    }
  }
}

dat$surname.freq <- as.factor(dat$surname.freq)


#bar graph
dat %>% filter(!is.na(Survived)) %>%
  ggplot(aes(x=surname.freq, fill=Survived)) +
  geom_bar(position = "fill")+
  ggtitle("Surname Frequency Survival Rate by Proportion")
```




```
dat %>% filter(!is.na(Survived)) %>%
  ggplot(aes(x=surname.freq, fill=Survived)) +
  geom_bar() +
  geom_label(stat = "count", position = position_stack(0.5), aes(label= ..count..))+
  ggtitle("Surname Frequency Survival by Count")
```




```
table(dat$surname.freq[1:891], dat$Survived[1:891])
```

```
##
##        0   1
##   1  298 149
##   2   88  80
##   3   58  63
##   4   38  24
##   5   12   7
##   6   32  13
##   7    2   1
##   8    7   3
##   11  14   2
```

```
surname.freq.prop <- prop.func("surname.freq")

surname.freq.prop
```

```
##       no surv      surv
## 1  0.6666667 0.3333333
## 2  0.5238095 0.4761905
## 3  0.4793388 0.5206612
## 4  0.6129032 0.3870968
## 5  0.6315789 0.3684211
## 6  0.7111111 0.2888889
## 7  0.6666667 0.3333333
## 8  0.7000000 0.3000000
## 11 0.8750000 0.1250000
```

```r
#notice that surname.freq 2,3 is likely hard to predict
#however, more the surname.freq increased from 4 to 11, they are more likely not survived

#therefore, low surv rate -> 1,4,5,6,7,8,11
#unknown -> 2,3

dat$surname.freq <- as.character(dat$surname.freq)

dat$surname.freq.surv <- NA
for(i in 1:nrow(dat)){
  if(dat$surname.freq[i] %in% c(1,4,5,6,7,8,11)){
    dat$surname.freq.surv[i] <- "low"
  }
  if(dat$surname.freq[i] %in% c(2,3)){
    dat$surname.freq.surv[i] <- "unknown"
  }
}
dat$surname.freq.surv <- as.factor(dat$surname.freq.surv)

table(dat$surname.freq.surv)
```

```
##
##     low unknown
##     854     455
```

```r
dat <- subset(dat, select=-c(surname.freq, Name, surname))

summary(dat)
```

```
##   PassengerId   Survived   Pclass  Sex          Age
##  Min.   :   1   0   :549   1:323   0:843   Min.   : 0.17
##  1st Qu.: 328   1   :342   2:277   1:466   1st Qu.:21.00
##  Median : 655   NA's:418   3:709           Median :28.32
##  Mean   : 655                               Mean   :29.52
##  3rd Qu.: 982                               3rd Qu.:36.50
##  Max.   :1309                               Max.   :80.00
##       Fare          Embarked Cabin.ox deck.surv   cabin.freq.surv
##  Min.   :  0.000   C:272    0:1014   high: 267   high: 289
##  1st Qu.:  7.896   Q:123    1: 295   low :1042   low :1020
##  Median : 14.454   S:914
##  Mean   : 33.281
##  3rd Qu.: 31.275
##  Max.   :512.329
##  ticket.alone ticket.let.surv     family           name
##  0:713        high:323        Min.   : 1.000   Master: 61
##  1:596        low :986        1st Qu.: 1.000   Miss  :260
##                               Median : 1.000   Mr    :782
##                               Mean   : 1.884   Mrs   :206
##                               3rd Qu.: 2.000
##                               Max.   :11.000
##  surname.freq.surv
##  low    :854
##  unknown:455
##
```

```
##
##
##
```

Investigating correlation or relationship between each variables in our dataset

```
#Let's see the correlation or relationship between each variables in our dataset

#factor vs factor - chisq test : null H0 = two factor variables are independent
#factor vs numeric - anova test : null H0 = at least one factor has different mean than others
#numeric vs numeric - correlation : linear relationship between vars,
#more than 0.5 means they have some relationship to each other

relationship.test <- function(variables, dummy.data, data){

  for(i in variables){
    for(j in variables){

      #factor vs factor : chisq.test
      if(is.factor(data[,i])){
        if(is.factor(data[,j])){
          dummy.data[dummy.data$cols == i,j] <- round(chisq.test(data[,i], data[,j])$p.value,3)
          }
      }

      #factor vs numeric : anova
      if(is.factor(data[,i])){
        if(is.numeric(data[,j])){
          dummy.data[dummy.data$cols == i,j] <-
            round(summary(aov(data[,j]~data[,i]))[[1]][["Pr(>F)"]][[1]],3)
          }
      }
      if(is.numeric(data[,i])){
        if(is.factor(data[,j])){
          dummy.data[dummy.data$cols == i,j] <-
            round(summary(aov(data[,i]~data[,j]))[[1]][["Pr(>F)"]][[1]],3)
          }
        }

      #numeric vs numeric : correlation
      if(is.numeric(data[,i])){
        if(is.numeric(data[,j])){
          dummy.data[dummy.data$cols == i,j] <- round(cor(data[,i], data[,j]),3)
          }
        }
    }
  }

  return(dummy.data)
}

#creating variables
variables <- colnames(dat)[2:ncol(dat)]
```

```
#dummy data
test.data <- data.frame(cols = variables)

data.pval <- relationship.test(variables, test.data, dat)
```

```
## Warning in chisq.test(data[, i], data[, j]): Chi-squared approximation may
## be incorrect
```

```
data.pval
```

```
##                 cols Survived Pclass   Sex    Age  Fare Embarked Cabin.ox
## 1           Survived    0.000  0.000 0.000  0.031 0.000    0.000    0.000
## 2             Pclass    0.000  0.000 0.000  0.000 0.000    0.000    0.000
## 3                Sex    0.000  0.000 0.000  0.002 0.000    0.000    0.000
## 4                Age    0.031  0.000 0.002  1.000 0.190    0.000    0.000
## 5               Fare    0.000  0.000 0.000  0.190 1.000    0.000    0.000
## 6           Embarked    0.000  0.000 0.000  0.000 0.000    0.000    0.000
## 7           Cabin.ox    0.000  0.000 0.000  0.000 0.000    0.000    0.000
## 8          deck.surv    0.000  0.000 0.000  0.000 0.000    0.000    0.000
## 9    cabin.freq.surv    0.000  0.000 0.000  0.000 0.000    0.000    0.000
## 10      ticket.alone    0.000  0.000 0.000  0.007 0.000    0.000    0.000
## 11   ticket.let.surv    0.000  0.000 0.000  0.000 0.000    0.000    0.000
## 12            family    0.620  0.102 0.000 -0.224 0.227    0.001    0.609
## 13              name    0.000  0.000 0.000  0.000 0.000    0.000    0.000
## 14 surname.freq.surv    0.000  0.000 0.000  0.753 0.000    0.001    0.000
##    deck.surv cabin.freq.surv ticket.alone ticket.let.surv family name
## 1      0.000           0.000        0.000           0.000  0.620    0
## 2      0.000           0.000        0.000           0.000  0.102    0
## 3      0.000           0.000        0.000           0.000  0.000    0
## 4      0.000           0.000        0.007           0.000 -0.224    0
## 5      0.000           0.000        0.000           0.000  0.227    0
## 6      0.000           0.000        0.000           0.000  0.001    0
## 7      0.000           0.000        0.000           0.000  0.609    0
## 8      0.000           0.000        0.000           0.000  0.386    0
## 9      0.000           0.000        0.000           0.000  0.601    0
## 10     0.000           0.000        0.000           0.000  0.000    0
## 11     0.000           0.000        0.000           0.000  0.085    0
## 12     0.386           0.601        0.000           0.085  1.000    0
## 13     0.000           0.000        0.000           0.000  0.000    0
## 14     0.000           0.000        0.000           0.014  0.037    0
##    surname.freq.surv
## 1              0.000
## 2              0.000
## 3              0.000
## 4              0.753
## 5              0.000
## 6              0.001
## 7              0.000
## 8              0.000
## 9              0.000
## 10             0.000
## 11             0.014
## 12             0.037
## 13             0.000
```

```
## 14             0.000
```

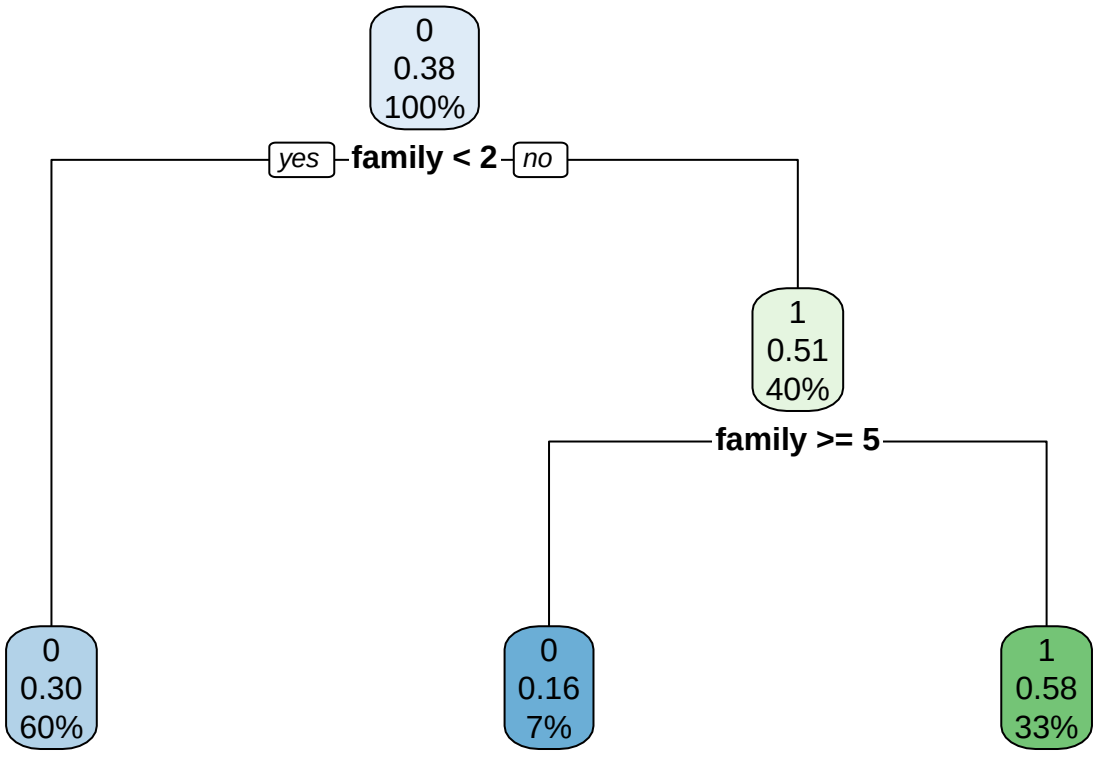
```
#factor vs factor : if <0.05 (p value), highly dependent, if not, independent
#factor vs numeric : if <0.05, at least one factor has different mean than others.
#if not, all factor has similar mean (non linear)
#numeric vs numeric : if <0.5, low correlation, if not, high correlation
```

Creating familyGroup from investigation of relationship between each variables

```
#Lets make family to be better predictor

tr <- rpart(Survived~family, dat)
tr
```

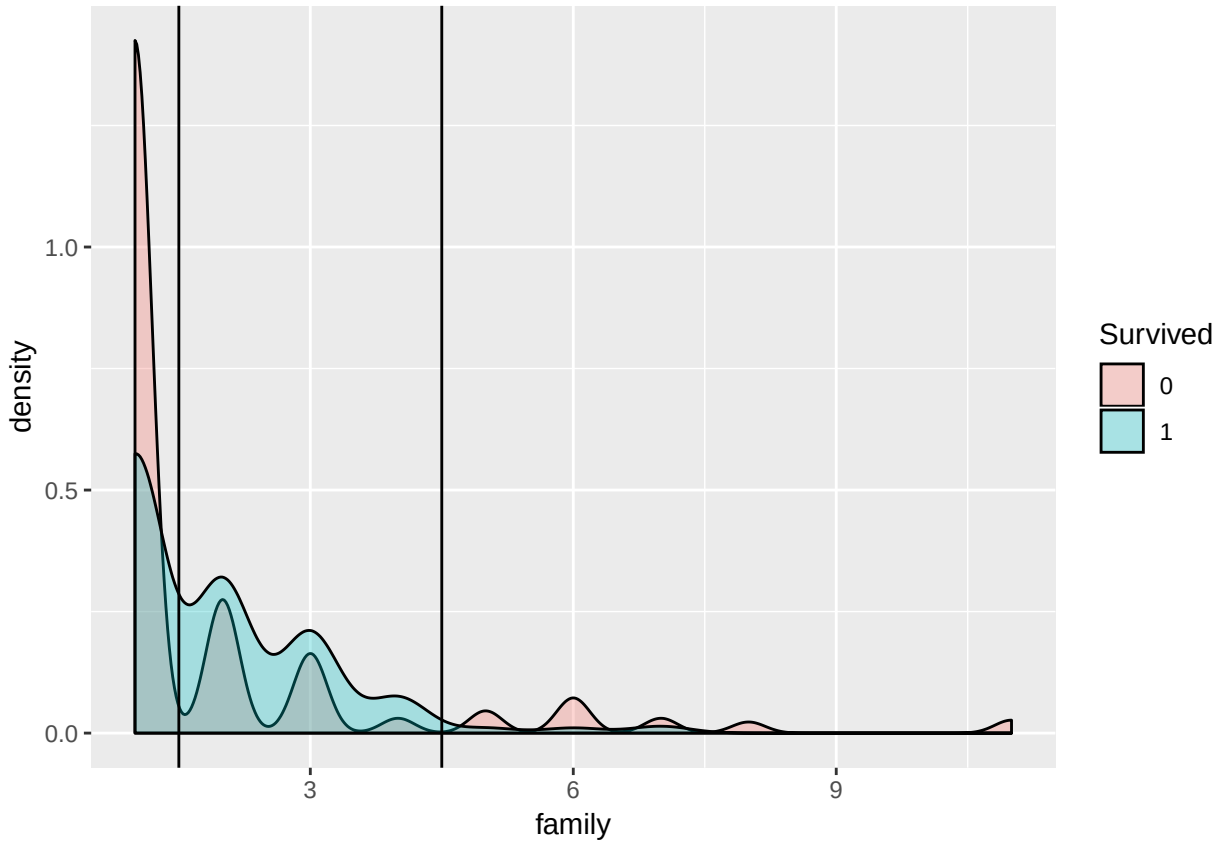
```
## n=891 (418 observations deleted due to missingness)
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 891 342 0 (0.6161616 0.3838384)
##   2) family< 1.5 537 163 0 (0.6964618 0.3035382) *
##   3) family>=1.5 354 175 1 (0.4943503 0.5056497)
##     6) family>=4.5 62  10 0 (0.8387097 0.1612903) *
##     7) family< 4.5 292 123 1 (0.4212329 0.5787671) *
```

```
rpart.plot(tr)
```




```
dat %>% filter(!is.na(Survived)) %>%
  ggplot(aes(x=family, fill=Survived))+
  geom_density(alpha = 0.3)+
```

```
  geom_vline(xintercept=c(1.5, 4.5))
```




```
#1.5 and 4.5

dat$familyGroup <- as.factor(ifelse(dat$family < 1.5, "alone",
                                    ifelse(dat$family > 1.5 & dat$family < 4.5, "small fam", "large fam

table(dat$familyGroup)
```

```
##
##     alone large fam small fam
##       790        82       437
```

```
variables <- colnames(dat)[2:ncol(dat)]
test.data <- data.frame(cols = variables)
test.data
```

```
##                cols
## 1          Survived
## 2            Pclass
## 3               Sex
## 4               Age
## 5              Fare
## 6          Embarked
## 7          Cabin.ox
## 8         deck.surv
## 9   cabin.freq.surv
## 10     ticket.alone
```

```
## 11   ticket.let.surv
## 12            family
## 13              name
## 14 surname.freq.surv
## 15       familyGroup
```

```
data.pval <- relationship.test(variables, test.data, dat)
```

```
## Warning in chisq.test(data[, i], data[, j]): Chi-squared approximation may
## be incorrect

## Warning in chisq.test(data[, i], data[, j]): Chi-squared approximation may
## be incorrect

## Warning in chisq.test(data[, i], data[, j]): Chi-squared approximation may
## be incorrect
```

```
data.pval
```

```
##                 cols Survived Pclass   Sex    Age  Fare Embarked Cabin.ox
## 1           Survived    0.000  0.000 0.000  0.031 0.000    0.000    0.000
## 2             Pclass    0.000  0.000 0.000  0.000 0.000    0.000    0.000
## 3                Sex    0.000  0.000 0.000  0.002 0.000    0.000    0.000
## 4                Age    0.031  0.000 0.002  1.000 0.190    0.000    0.000
## 5               Fare    0.000  0.000 0.000  0.190 1.000    0.000    0.000
## 6           Embarked    0.000  0.000 0.000  0.000 0.000    0.000    0.000
## 7           Cabin.ox    0.000  0.000 0.000  0.000 0.000    0.000    0.000
## 8          deck.surv    0.000  0.000 0.000  0.000 0.000    0.000    0.000
## 9    cabin.freq.surv    0.000  0.000 0.000  0.000 0.000    0.000    0.000
## 10      ticket.alone    0.000  0.000 0.000  0.007 0.000    0.000    0.000
## 11   ticket.let.surv    0.000  0.000 0.000  0.000 0.000    0.000    0.000
## 12            family    0.620  0.102 0.000 -0.224 0.227    0.001    0.609
## 13              name    0.000  0.000 0.000  0.000 0.000    0.000    0.000
## 14 surname.freq.surv    0.000  0.000 0.000  0.753 0.000    0.001    0.000
## 15       familyGroup    0.000  0.000 0.000  0.000 0.000    0.000    0.000
##    deck.surv cabin.freq.surv ticket.alone ticket.let.surv family name
## 1      0.000           0.000        0.000           0.000  0.620    0
## 2      0.000           0.000        0.000           0.000  0.102    0
## 3      0.000           0.000        0.000           0.000  0.000    0
## 4      0.000           0.000        0.007           0.000 -0.224    0
## 5      0.000           0.000        0.000           0.000  0.227    0
## 6      0.000           0.000        0.000           0.000  0.001    0
## 7      0.000           0.000        0.000           0.000  0.609    0
## 8      0.000           0.000        0.000           0.000  0.386    0
## 9      0.000           0.000        0.000           0.000  0.601    0
## 10     0.000           0.000        0.000           0.000  0.000    0
## 11     0.000           0.000        0.000           0.000  0.085    0
## 12     0.386           0.601        0.000           0.085  1.000    0
## 13     0.000           0.000        0.000           0.000  0.000    0
## 14     0.000           0.000        0.000           0.014  0.037    0
## 15     0.000           0.000        0.000           0.000  0.000    0
##    surname.freq.surv familyGroup
## 1              0.000           0
## 2              0.000           0
## 3              0.000           0
```

```
## 4              0.753           0
## 5              0.000           0
## 6              0.001           0
## 7              0.000           0
## 8              0.000           0
## 9              0.000           0
## 10             0.000           0
## 11             0.014           0
## 12             0.037           0
## 13             0.000           0
## 14             0.000           0
## 15             0.000           0
```

```
dat <- dat %>% subset(select=-c(PassengerId, family))

summary(dat)
```

```
##  Survived   Pclass  Sex          Age              Fare         Embarked
##  0   :549   1:323   0:843   Min.   : 0.17   Min.   :  0.000   C:272
##  1   :342   2:277   1:466   1st Qu.:21.00   1st Qu.:  7.896   Q:123
##  NA's:418   3:709           Median :28.32   Median : 14.454   S:914
##                             Mean   :29.52   Mean   : 33.281
##                             3rd Qu.:36.50   3rd Qu.: 31.275
##                             Max.   :80.00   Max.   :512.329
##  Cabin.ox deck.surv   cabin.freq.surv ticket.alone ticket.let.surv
##  0:1014   high: 267   high: 289       0:713        high:323
##  1: 295   low :1042   low :1020       1:596        low :986
##
##
##
##
##      name     surname.freq.surv    familyGroup
##  Master: 61   low    :854       alone    :790
##  Miss  :260   unknown:455       large fam: 82
##  Mr    :782                     small fam:437
##  Mrs   :206
##
##
```

Splitting train and test set to start modeling

```
#train / test
training <- dat %>% filter(!is.na(Survived))
testing <- dat %>% filter(is.na(Survived))

summary(training)
```

```
##  Survived Pclass  Sex          Age              Fare        Embarked
##  0:549    1:216   0:577   Min.   : 0.42   Min.   :  0.00   C:170
##  1:342    2:184   1:314   1st Qu.:21.00   1st Qu.:  7.91   Q: 77
##           3:491           Median :28.32   Median : 14.45   S:644
##                           Mean   :29.43   Mean   : 32.20
##                           3rd Qu.:36.75   3rd Qu.: 31.00
##                           Max.   :80.00   Max.   :512.33
```

```
##  Cabin.ox deck.surv  cabin.freq.surv ticket.alone ticket.let.surv
##  0:687    high:184   high:200        0:481        high:219
##  1:204    low :707   low :691        1:410        low :672
##
##
##
##
##      name     surname.freq.surv    familyGroup
##  Master: 40   low    :602       alone    :537
##  Miss  :182   unknown:289       large fam: 62
##  Mr    :537                     small fam:292
##  Mrs   :132
##
##
```

```
summary(testing)
```

```
##  Survived   Pclass  Sex          Age              Fare         Embarked
##  0   :  0   1:107   0:266   Min.   : 0.17   Min.   :  0.000   C:102
##  1   :  0   2: 93   1:152   1st Qu.:22.00   1st Qu.:  7.896   Q: 46
##  NA's:418   3:218           Median :28.32   Median : 14.454   S:270
##                             Mean   :29.70   Mean   : 35.577
##                             3rd Qu.:36.38   3rd Qu.: 31.472
##                             Max.   :76.00   Max.   :512.329
##  Cabin.ox deck.surv  cabin.freq.surv ticket.alone ticket.let.surv
##  0:327    high: 83   high: 89        0:232        high:104
##  1: 91    low :335   low :329        1:186        low :314
##
##
##
##
##      name     surname.freq.surv    familyGroup
##  Master: 21   low    :252       alone    :253
##  Miss  : 78   unknown:166       large fam: 20
##  Mr    :245                     small fam:145
##  Mrs   : 74
##
##
```

```
#we have 14 predictors.
#we might want to remove some predictors that have low importance while modeling
```

From Cabin.. - Cabin.ox : Cabin NA = 0 or Cabin = 1 - deck.surv : extract the first letter of cabin, with the probability of survival for the deck, splitted into 2 groups, which are high / low - cabin.freq.surv : 2 groups by surv rate with cabin frequency

from Ticket.. - ticket.alone : unique ticket = 0 other 1 - ticket.let.surv : with the first letter of ticket, splitted into 2 groups by surv rate of the ticket letter

from Name.. - name : Master / Miss / Mr / Mrs - surname.freq.surv : groups by surv rate with surname frequency

Caret - Cross Validation Creating useful function for modeling ——————

```
#creating function for Caret modeling

model <- function(method, training, control,grid,...){
```

```
  if(is.null(grid)){
    model.fit <- train(Survived~.,
                       data = training,
                       method = method,
                       trControl = control,
                       ...)
    return(model.fit)
  }

  else{
    model.fit <- train(Survived~.,
                       data = training,
                       method = method,
                       trControl = control,
                       tuneGrid = grid,
                       ...)
    return(model.fit)
  }
}

#accuracy of model
acc <- function(pred, act, data){
  return(sum(diag(table(pred, act)))/nrow(data))
}

#10 folds cv
control <- trainControl(method = "cv", number = 10)
```

I will use Random Forest / Gradient Boosting Method / Support Vector Machine with kernel radial

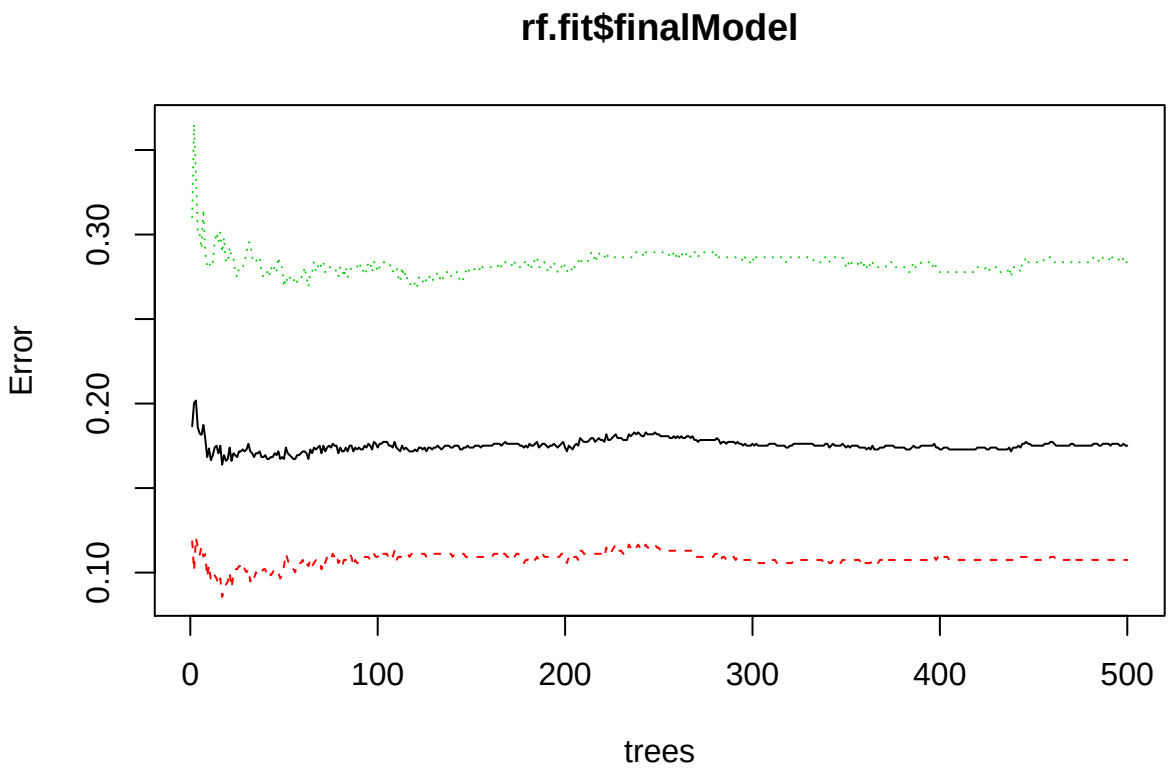
Random Forest

```
#typical mtry in classification = sqrt(# of predictors)
rf.fit <- train(Survived~., data = training,
                method="rf", trControl = control,
                ntree=500, importance = TRUE,
                tuneGrid = expand.grid(mtry = round(sqrt(ncol(training)-1))))

rf.fit
```

```
## Random Forest
##
## 891 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 802, 803, 801, 802, 802, 802, ...
## Resampling results:
##
##   Accuracy  Kappa
```

```
##   0.818198  0.605844
##
## Tuning parameter 'mtry' was held constant at a value of 4
```

```
plot(rf.fit$finalModel)
```



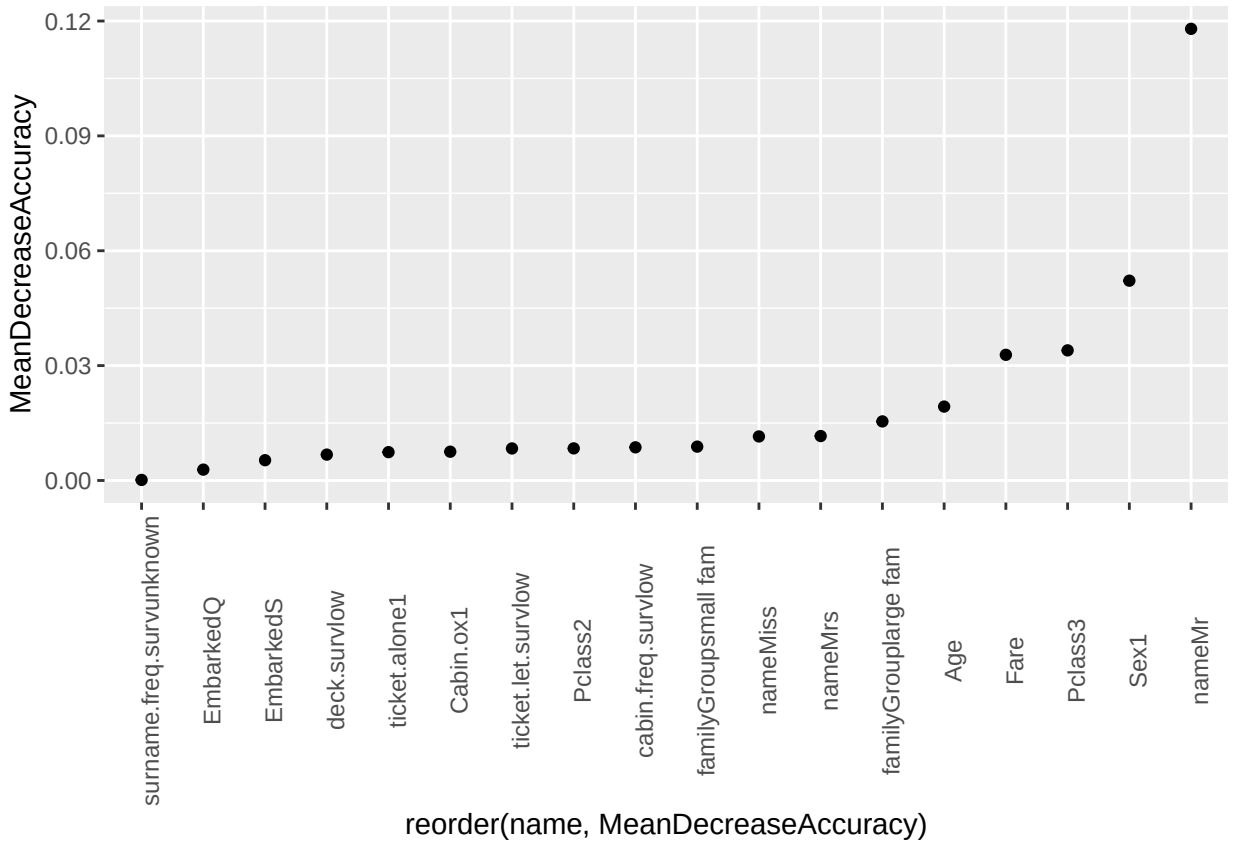

```
varImp(rf.fit)
```

```
## rf variable importance
##
##                           Importance
## nameMr                        100.00
## Pclass3                        65.00
## Fare                           57.79
## Age                            55.01
## Sex1                           53.13
## familyGrouplarge fam           48.89
## Pclass2                        34.99
## familyGroupsmall fam           32.14
## ticket.let.survlow             28.91
## nameMiss                       28.70
## nameMrs                        27.61
## ticket.alone1                  25.73
## EmbarkedS                      25.31
## cabin.freq.survlow             21.83
## Cabin.ox1                      21.51
## deck.survlow                   19.73
## EmbarkedQ                      12.08
## surname.freq.survunknown        0.00
```

```r
rf.fit.result <- data.frame(rf.fit$finalModel$importance[,"MeanDecreaseAccuracy"])
colnames(rf.fit.result) <- "MeanDecreaseAccuracy"

rf.fit.result
```

```
##                           MeanDecreaseAccuracy
## Pclass2                          0.0083833143
## Pclass3                          0.0339847157
## Sex1                             0.0521806205
## Age                              0.0192999328
## Fare                             0.0328228594
## EmbarkedQ                        0.0028322820
## EmbarkedS                        0.0052953266
## Cabin.ox1                        0.0075139351
## deck.survlow                     0.0067570091
## cabin.freq.survlow               0.0086612099
## ticket.alone1                    0.0073954415
## ticket.let.survlow               0.0083645876
## nameMiss                         0.0114980118
## nameMr                           0.1179581941
## nameMrs                          0.0116054104
## surname.freq.survunknown         0.0001445271
## familyGrouplarge fam             0.0154279129
## familyGroupsmall fam             0.0088394701
```

```r
rf.fit.result %>% mutate(name = rownames(rf.fit.result)) %>%
  arrange(MeanDecreaseAccuracy) %>%
  ggplot(aes(x=reorder(name, MeanDecreaseAccuracy),y=MeanDecreaseAccuracy))+
  geom_point()+
  theme(axis.text.x = element_text(angle=90))
```

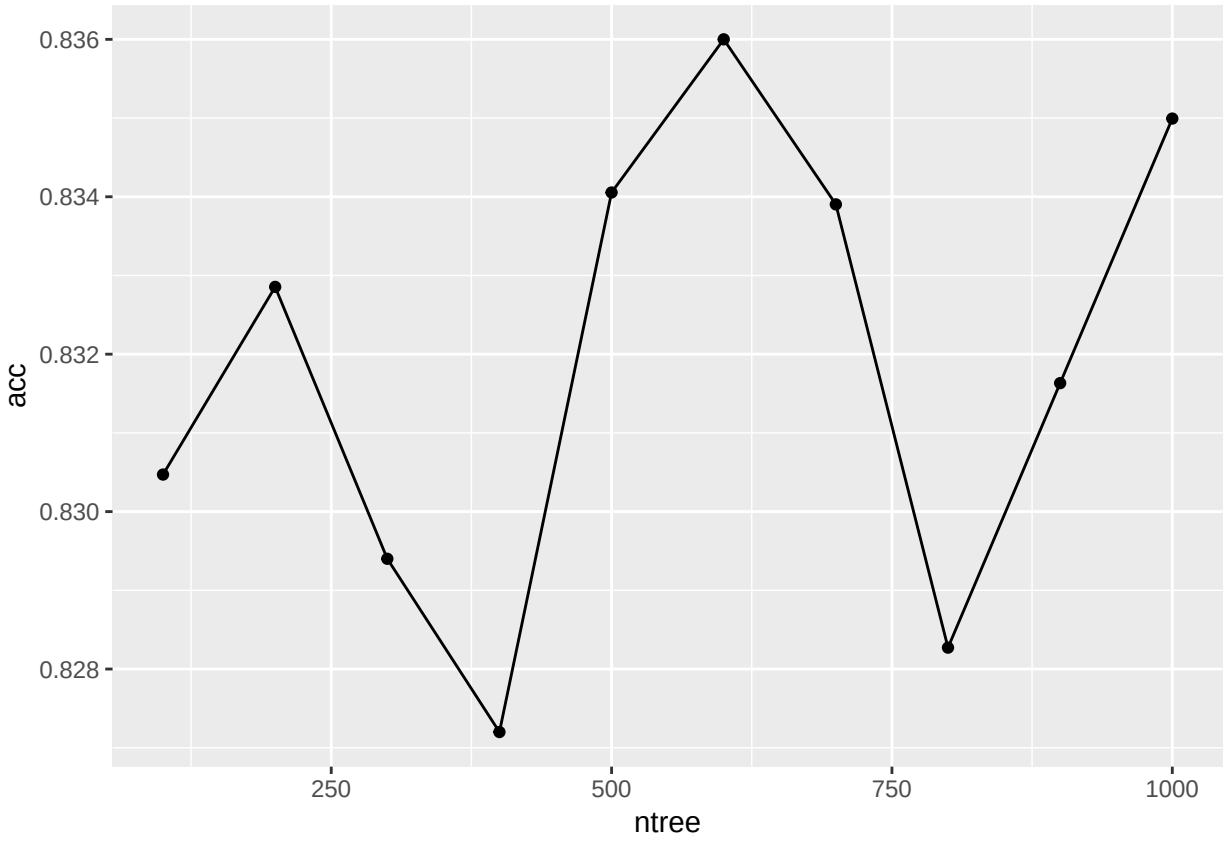



```
#remove Embarked / surname.freq.surv


#tuning parameter mtry and ntree by cross validation
#typical mtry is sqrt(# of predictor)
#ntree: in small dataset -> 100 in large dataset -> 500~1000 sufficient
#larger ntree is more stable, but takes long time
rf.grid <- expand.grid(mtry = seq(2,10, by=2))

rf.acc <- data.frame(ntree = seq(100,1000, by=100), minacc = NA, acc = NA)

for(i in seq(100, 1000, by=100)){
  rf.fit <- train(Survived~., data=training %>% subset(select = -c(Embarked, surname.freq.surv)),
                  method = "rf", trControl = control,
                  ntree=i, tuneGrid = rf.grid, importance = TRUE)
  rf.acc[rf.acc$ntree == i,2] <- max(rf.fit$results$Accuracy) -
    rf.fit$results$AccuracySD[which.max(rf.fit$results$Accuracy)]
  rf.acc[rf.acc$ntree == i,3] <- max(rf.fit$results$Accuracy)
}

rf.acc
```

```
##    ntree    minacc       acc
## 1    100 0.7884610 0.8304707
## 2    200 0.8057981 0.8328550
## 3    300 0.7835773 0.8294007
## 4    400 0.7827001 0.8271998
```

```
## 5    500 0.7931961 0.8340546
## 6    600 0.7994935 0.8360005
## 7    700 0.7869358 0.8339034
## 8    800 0.7918917 0.8282718
## 9    900 0.7937269 0.8316323
## 10  1000 0.7974265 0.8349938
```

```
ggplot(rf.acc, aes(x=ntree, y=acc))+
  geom_line()+
  geom_point()
```




```
g.ntree <- rf.acc$ntree[which.max(rf.acc$minacc)]
g.ntree
```

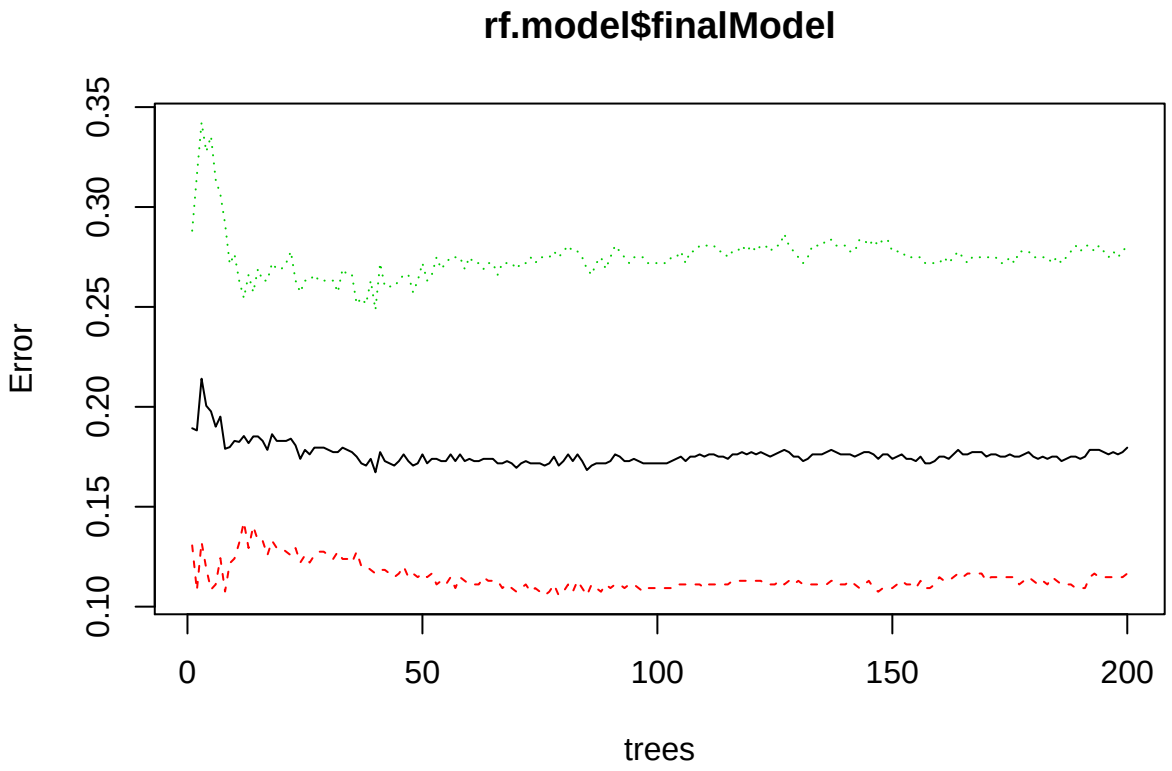
```
## [1] 200
```

```
#I will choose the ntree that has maximum value of minacc = max accuracy - accuracy sd

rf.model <- train(Survived~.,
                  data=training %>% subset(select=-c(Embarked, surname.freq.surv)),
                  method = "rf", trControl = control,
                  ntree=g.ntree, tuneGrid = rf.grid, importance=TRUE)

rf.model
```

```
## Random Forest
##
## 891 samples
##  11 predictor
```

```
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 802, 801, 802, 802, 802, 803, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8306960  0.6353096
##    4    0.8340169  0.6416354
##    6    0.8329307  0.6411699
##    8    0.8283855  0.6321354
##   10    0.8183231  0.6124362
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 4.
```

```
plot(rf.model$finalModel)
```




```
max(rf.model$results$Accuracy)
```

```
## [1] 0.8340169
```

```
#about 83%

varImp(rf.model)
```

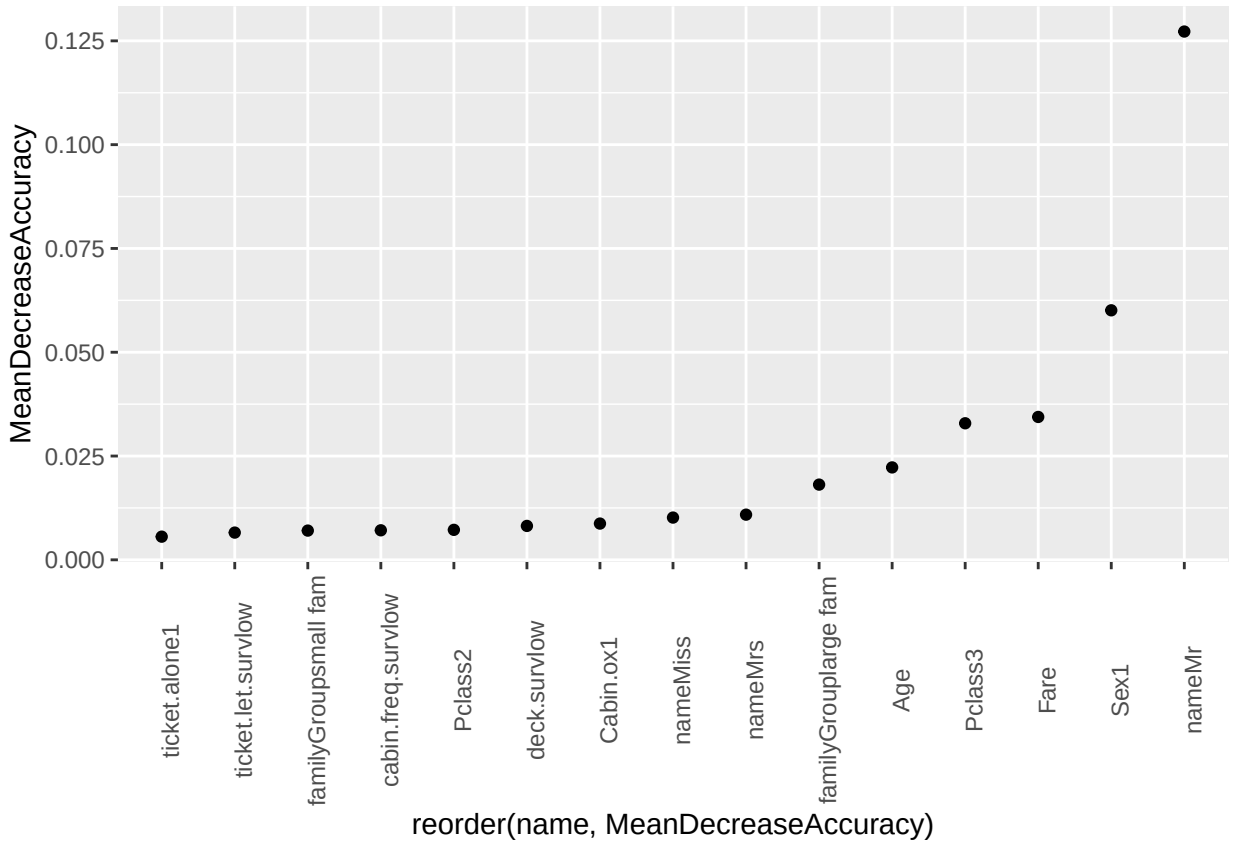
```
## rf variable importance
##
##                      Importance
## nameMr                  100.0000
```

```
## Pclass3                 53.6375
## Fare                    47.1767
## Age                     45.7715
## familyGrouplarge fam    44.9501
## Sex1                    44.8302
## Pclass2                 17.3487
## familyGroupsmall fam     8.8698
## cabin.freq.survlow       7.7383
## ticket.let.survlow       5.9648
## nameMrs                  5.0994
## ticket.alone1            2.9875
## Cabin.ox1                2.5546
## deck.survlow             0.8606
## nameMiss                 0.0000
```

```
rf.model.result <- data.frame(rf.model$finalModel$importance[,"MeanDecreaseAccuracy"])
colnames(rf.model.result) <- "MeanDecreaseAccuracy"

rf.model.result
```

```
##                       MeanDecreaseAccuracy
## Pclass2                         0.007222812
## Pclass3                         0.032887326
## Sex1                            0.060099248
## Age                             0.022261381
## Fare                            0.034414314
## Cabin.ox1                       0.008732015
## deck.survlow                    0.008164329
## cabin.freq.survlow              0.007111068
## ticket.alone1                   0.005574849
## ticket.let.survlow              0.006556810
## nameMiss                        0.010175821
## nameMr                          0.127246900
## nameMrs                         0.010879219
## familyGrouplarge fam            0.018118710
## familyGroupsmall fam            0.007041033
```

```
rf.model.result %>% mutate(name = rownames(rf.model.result)) %>%
  arrange(MeanDecreaseAccuracy) %>%
  ggplot(aes(x=reorder(name, MeanDecreaseAccuracy),y=MeanDecreaseAccuracy))+
  geom_point()+
  theme(axis.text.x = element_text(angle=90))
```




```
rf.minacc <- max(rf.model$results$Accuracy) -
  rf.model$results$AccuracySD[which.max(rf.model$results$Accuracy)]
rf.minacc
```

```
## [1] 0.804023
```

```
#about 80%

#predict on real test
rf.pred <- predict(rf.model, training)

confusionMatrix(rf.pred, training$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 520  60
##          1  29 282
##
##                Accuracy : 0.9001
##                  95% CI : (0.8785, 0.919)
##     No Information Rate : 0.6162
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7852
##
##  Mcnemar's Test P-Value : 0.001473
```

```
##
##             Sensitivity : 0.9472
##             Specificity : 0.8246
##          Pos Pred Value : 0.8966
##          Neg Pred Value : 0.9068
##              Prevalence : 0.6162
##          Detection Rate : 0.5836
##    Detection Prevalence : 0.6510
##       Balanced Accuracy : 0.8859
##
##        'Positive' Class : 0
##
```

```
#93.15%

#training accuracy - cv accuracy
acc(rf.pred, training$Survived, training) - max(rf.model$results$Accuracy)
```

```
## [1] 0.06609538
```

```
#0.0987
```

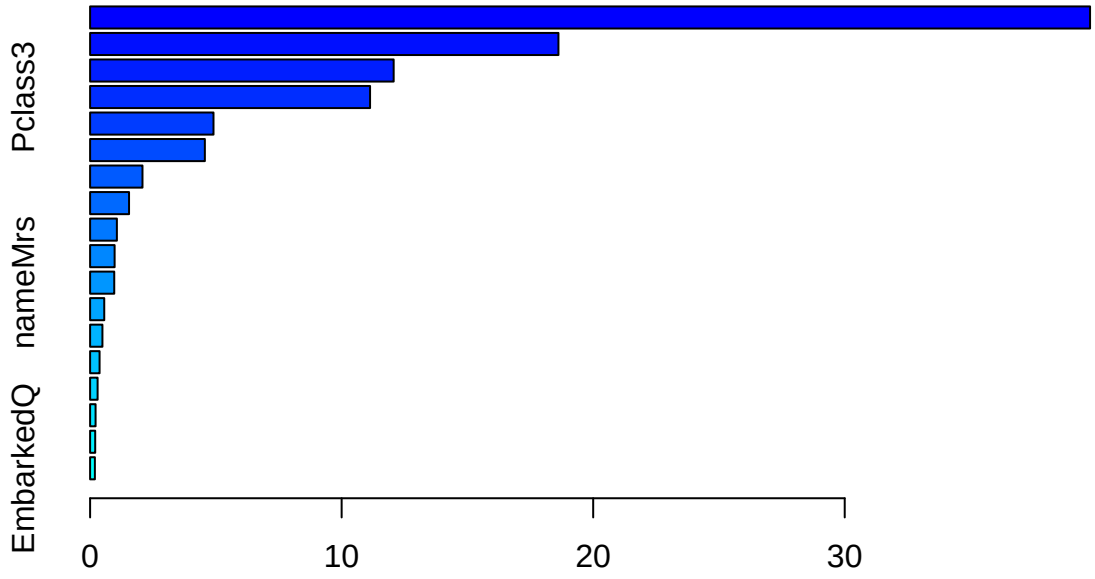
Gradient Boosting Method

```
#modeling without tuning parameter
boost.model <- train(Survived~.,
                     data = training,
                     method = "gbm",
                     verbose = FALSE,
                     trControl = control,
                     tuneGrid = NULL)

boost.model
```

```
## Stochastic Gradient Boosting
##
## 891 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 803, 802, 802, 802, 802, 801, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.8317580  0.6360874
##   1                  100      0.8305834  0.6394458
##   1                  150      0.8305961  0.6410029
##   2                   50      0.8272251  0.6300760
##   2                  100      0.8294220  0.6348922
##   2                  150      0.8327676  0.6406885
##   3                   50      0.8261145  0.6256818
##   3                  100      0.8305326  0.6366158
```

```
##   3                  150      0.8372370  0.6524961
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
summary(boost.model$finalModel)
```




```
##                                             var    rel.inf
## nameMr                                   nameMr 39.7532763
## Fare                                       Fare 18.6188959
## Age                                         Age 12.0676146
## Pclass3                                 Pclass3 11.1299420
## familyGrouplarge fam       familyGrouplarge fam  4.9057741
## ticket.let.survlow           ticket.let.survlow  4.5615743
## cabin.freq.survlow           cabin.freq.survlow  2.0822052
## EmbarkedS                             EmbarkedS  1.5483528
## Sex1                                       Sex1  1.0640973
## deck.survlow                       deck.survlow  0.9743494
## nameMrs                                 nameMrs  0.9591543
## familyGroupsmall fam       familyGroupsmall fam  0.5619401
## nameMiss                               nameMiss  0.4893777
## ticket.alone1                     ticket.alone1  0.3761116
## Cabin.ox1                             Cabin.ox1  0.2978196
## Pclass2                                 Pclass2  0.2168543
## surname.freq.survunknown surname.freq.survunknown  0.2024787
## EmbarkedQ                             EmbarkedQ  0.1901817
```

```
#surname.freq.surv / Embarked
```

```r
#Grid Search
#I put relatively large value of shrinkage to prevent overfitting
boost.grid <- expand.grid(n.trees = seq(100,6000, by=150),
                          interaction.depth = c(1,2,3,4),
                          shrinkage = c(0.01,0.1),
                          n.minobsinnode = c(10))

#modeling
boost.model <- train(Survived~.,
                     data = training %>%
                       subset(select = -c(Embarked, surname.freq.surv)),
                     method = "gbm",
                     verbose = FALSE,
                     trControl = control,
                     tuneGrid = boost.grid)


boost.model
```

```
## Stochastic Gradient Boosting
##
## 891 samples
##  11 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 802, 803, 802, 802, 802, 802, ...
## Resampling results across tuning parameters:
##
##   shrinkage  interaction.depth  n.trees  Accuracy   Kappa
##   0.01       1                   100     0.7936537  0.5632148
##   0.01       1                   250     0.8238912  0.6208086
##   0.01       1                   400     0.8317189  0.6374534
##   0.01       1                   550     0.8328175  0.6399802
##   0.01       1                   700     0.8305703  0.6365401
##   0.01       1                   850     0.8339536  0.6457201
##   0.01       1                  1000     0.8328300  0.6438964
##   0.01       1                  1150     0.8283228  0.6347358
##   0.01       1                  1300     0.8305575  0.6394918
##   0.01       1                  1450     0.8316811  0.6420497
##   0.01       1                  1600     0.8327920  0.6448577
##   0.01       1                  1750     0.8327920  0.6454263
##   0.01       1                  1900     0.8327664  0.6452870
##   0.01       1                  2050     0.8327664  0.6452870
##   0.01       1                  2200     0.8327664  0.6453536
##   0.01       1                  2350     0.8338900  0.6480031
##   0.01       1                  2500     0.8361247  0.6521952
##   0.01       1                  2650     0.8361375  0.6517899
##   0.01       1                  2800     0.8350011  0.6495905
##   0.01       1                  2950     0.8338775  0.6470887
##   0.01       1                  3100     0.8350139  0.6492835
##   0.01       1                  3250     0.8327792  0.6448295
```

```
##   0.01       1                3400      0.8350139  0.6491335
##   0.01       1                3550      0.8350139  0.6491335
##   0.01       1                3700      0.8361375  0.6512399
##   0.01       1                3850      0.8372611  0.6538141
##   0.01       1                4000      0.8361375  0.6512399
##   0.01       1                4150      0.8350011  0.6485594
##   0.01       1                4300      0.8327539  0.6436448
##   0.01       1                4450      0.8350011  0.6482306
##   0.01       1                4600      0.8338775  0.6459244
##   0.01       1                4750      0.8327539  0.6440994
##   0.01       1                4900      0.8327539  0.6440994
##   0.01       1                5050      0.8293832  0.6367408
##   0.01       1                5200      0.8305192  0.6390197
##   0.01       1                5350      0.8282593  0.6337258
##   0.01       1                5500      0.8293832  0.6360556
##   0.01       1                5650      0.8304940  0.6377844
##   0.01       1                5800      0.8282343  0.6337219
##   0.01       1                5950      0.8260121  0.6282732
##   0.01       2                 100      0.8283609  0.6295530
##   0.01       2                 250      0.8339411  0.6426976
##   0.01       2                 400      0.8305703  0.6365401
##   0.01       2                 550      0.8271868  0.6316252
##   0.01       2                 700      0.8316684  0.6413005
##   0.01       2                 850      0.8395463  0.6577363
##   0.01       2                1000      0.8372991  0.6524511
##   0.01       2                1150      0.8440282  0.6664737
##   0.01       2                1300      0.8440154  0.6662624
##   0.01       2                1450      0.8451518  0.6691602
##   0.01       2                1600      0.8417555  0.6612676
##   0.01       2                1750      0.8417680  0.6607017
##   0.01       2                1900      0.8406444  0.6577620
##   0.01       2                2050      0.8417680  0.6602108
##   0.01       2                2200      0.8406444  0.6576546
##   0.01       2                2350      0.8406444  0.6576546
##   0.01       2                2500      0.8417555  0.6598438
##   0.01       2                2650      0.8428663  0.6627194
##   0.01       2                2800      0.8406191  0.6588026
##   0.01       2                2950      0.8406191  0.6588026
##   0.01       2                3100      0.8428791  0.6637936
##   0.01       2                3250      0.8428663  0.6636655
##   0.01       2                3400      0.8428663  0.6636655
##   0.01       2                3550      0.8462626  0.6712794
##   0.01       2                3700      0.8473990  0.6742219
##   0.01       2                3850      0.8451390  0.6687208
##   0.01       2                4000      0.8473990  0.6742219
##   0.01       2                4150      0.8485226  0.6768324
##   0.01       2                4300      0.8462626  0.6716788
##   0.01       2                4450      0.8462626  0.6716788
##   0.01       2                4600      0.8462751  0.6717619
##   0.01       2                4750      0.8462751  0.6717619
##   0.01       2                4900      0.8451643  0.6698677
##   0.01       2                5050      0.8451643  0.6698677
##   0.01       2                5200      0.8451643  0.6698677
##   0.01       2                5350      0.8462879  0.6718229
```

```
##   0.01       2                5500     0.8451643  0.6698341
##   0.01       2                5650     0.8440407  0.6672450
##   0.01       2                5800     0.8429043  0.6645347
##   0.01       2                5950     0.8417807  0.6619457
##   0.01       3                 100     0.8294842  0.6316023
##   0.01       3                 250     0.8316939  0.6383214
##   0.01       3                 400     0.8328047  0.6436548
##   0.01       3                 550     0.8384227  0.6550837
##   0.01       3                 700     0.8417935  0.6622909
##   0.01       3                 850     0.8429171  0.6639063
##   0.01       3                1000     0.8440404  0.6655279
##   0.01       3                1150     0.8395460  0.6559796
##   0.01       3                1300     0.8429168  0.6632951
##   0.01       3                1450     0.8462751  0.6706243
##   0.01       3                1600     0.8473735  0.6732346
##   0.01       3                1750     0.8462499  0.6706291
##   0.01       3                1900     0.8462374  0.6707236
##   0.01       3                2050     0.8451263  0.6685019
##   0.01       3                2200     0.8473862  0.6737557
##   0.01       3                2350     0.8473865  0.6738097
##   0.01       3                2500     0.8462754  0.6715879
##   0.01       3                2650     0.8440282  0.6664411
##   0.01       3                2800     0.8429046  0.6642090
##   0.01       3                2950     0.8429046  0.6645911
##   0.01       3                3100     0.8462754  0.6712875
##   0.01       3                3250     0.8451518  0.6690805
##   0.01       3                3400     0.8451518  0.6687002
##   0.01       3                3550     0.8440282  0.6660039
##   0.01       3                3700     0.8451518  0.6686468
##   0.01       3                3850     0.8440282  0.6663638
##   0.01       3                4000     0.8417810  0.6610475
##   0.01       3                4150     0.8429046  0.6637209
##   0.01       3                4300     0.8429046  0.6638973
##   0.01       3                4450     0.8451518  0.6688896
##   0.01       3                4600     0.8417682  0.6613580
##   0.01       3                4750     0.8372863  0.6518095
##   0.01       3                4900     0.8384099  0.6538973
##   0.01       3                5050     0.8372863  0.6513100
##   0.01       3                5200     0.8361628  0.6492064
##   0.01       3                5350     0.8350392  0.6466191
##   0.01       3                5500     0.8395335  0.6563622
##   0.01       3                5650     0.8372863  0.6513966
##   0.01       3                5800     0.8384099  0.6539509
##   0.01       3                5950     0.8372863  0.6517440
##   0.01       4                 100     0.8260887  0.6211006
##   0.01       4                 250     0.8305828  0.6358433
##   0.01       4                 400     0.8429171  0.6634408
##   0.01       4                 550     0.8440532  0.6659644
##   0.01       4                 700     0.8395713  0.6565727
##   0.01       4                 850     0.8406696  0.6590177
##   0.01       4                1000     0.8406696  0.6590177
##   0.01       4                1150     0.8406444  0.6589848
##   0.01       4                1300     0.8451138  0.6682091
##   0.01       4                1450     0.8496212  0.6783057
```

```
##   0.01       4                 1600      0.8496337  0.6787016
##   0.01       4                 1750      0.8473990  0.6742729
##   0.01       4                 1900      0.8485226  0.6764799
##   0.01       4                 2050      0.8429046  0.6643919
##   0.01       4                 2200      0.8429046  0.6642904
##   0.01       4                 2350      0.8417810  0.6616875
##   0.01       4                 2500      0.8440282  0.6665700
##   0.01       4                 2650      0.8417685  0.6614135
##   0.01       4                 2800      0.8417685  0.6609863
##   0.01       4                 2950      0.8406322  0.6583443
##   0.01       4                 3100      0.8406322  0.6585618
##   0.01       4                 3250      0.8350267  0.6460305
##   0.01       4                 3400      0.8350142  0.6460407
##   0.01       4                 3550      0.8327795  0.6417207
##   0.01       4                 3700      0.8294212  0.6353376
##   0.01       4                 3850      0.8294212  0.6353642
##   0.01       4                 4000      0.8305448  0.6376815
##   0.01       4                 4150      0.8316684  0.6401617
##   0.01       4                 4300      0.8305323  0.6379873
##   0.01       4                 4450      0.8316559  0.6404018
##   0.01       4                 4600      0.8282976  0.6326460
##   0.01       4                 4750      0.8282976  0.6328453
##   0.01       4                 4900      0.8294212  0.6357258
##   0.01       4                 5050      0.8282976  0.6335455
##   0.01       4                 5200      0.8305448  0.6380879
##   0.01       4                 5350      0.8282976  0.6334263
##   0.01       4                 5500      0.8282976  0.6334339
##   0.01       4                 5650      0.8294212  0.6358914
##   0.01       4                 5800      0.8294212  0.6359262
##   0.01       4                 5950      0.8271740  0.6314382
##   0.10       1                  100      0.8328047  0.6445756
##   0.10       1                  250      0.8305065  0.6402588
##   0.10       1                  400      0.8327415  0.6433424
##   0.10       1                  550      0.8338398  0.6446692
##   0.10       1                  700      0.8293454  0.6344697
##   0.10       1                  850      0.8293832  0.6349325
##   0.10       1                 1000      0.8304946  0.6385821
##   0.10       1                 1150      0.8260246  0.6276367
##   0.10       1                 1300      0.8305320  0.6379225
##   0.10       1                 1450      0.8260379  0.6283371
##   0.10       1                 1600      0.8260124  0.6296000
##   0.10       1                 1750      0.8305068  0.6390861
##   0.10       1                 1900      0.8282723  0.6338161
##   0.10       1                 2050      0.8237777  0.6244891
##   0.10       1                 2200      0.8237652  0.6242074
##   0.10       1                 2350      0.8271612  0.6317699
##   0.10       1                 2500      0.8271615  0.6325594
##   0.10       1                 2650      0.8249137  0.6273754
##   0.10       1                 2800      0.8260124  0.6290980
##   0.10       1                 2950      0.8226793  0.6229507
##   0.10       1                 3100      0.8237652  0.6242289
##   0.10       1                 3250      0.8271487  0.6323014
##   0.10       1                 3400      0.8249268  0.6272966
##   0.10       1                 3550      0.8260254  0.6288249
```

```
##   0.10       1                3700      0.8271360  0.6324021
##   0.10       1                3850      0.8192955  0.6148407
##   0.10       1                4000      0.8215302  0.6197339
##   0.10       1                4150      0.8192958  0.6152710
##   0.10       1                4300      0.8204196  0.6167322
##   0.10       1                4450      0.8159502  0.6075716
##   0.10       1                4600      0.8193210  0.6145655
##   0.10       1                4750      0.8215305  0.6195395
##   0.10       1                4900      0.8226288  0.6227076
##   0.10       1                5050      0.8226413  0.6214420
##   0.10       1                5200      0.8192958  0.6158666
##   0.10       1                5350      0.8249013  0.6280463
##   0.10       1                5500      0.8215679  0.6203335
##   0.10       1                5650      0.8204443  0.6183122
##   0.10       1                5800      0.8170860  0.6101468
##   0.10       1                5950      0.8215430  0.6196669
##   0.10       2                 100      0.8440154  0.6660839
##   0.10       2                 250      0.8474112  0.6730046
##   0.10       2                 400      0.8417807  0.6619221
##   0.10       2                 550      0.8451390  0.6687584
##   0.10       2                 700      0.8383975  0.6536862
##   0.10       2                 850      0.8417555  0.6612032
##   0.10       2                1000      0.8339153  0.6445766
##   0.10       2                1150      0.8305320  0.6387097
##   0.10       2                1300      0.8283098  0.6335442
##   0.10       2                1450      0.8249013  0.6262152
##   0.10       2                1600      0.8305317  0.6390597
##   0.10       2                1750      0.8282596  0.6347609
##   0.10       2                1900      0.8271360  0.6327316
##   0.10       2                2050      0.8271485  0.6317034
##   0.10       2                2200      0.8271485  0.6317787
##   0.10       2                2350      0.8226666  0.6220318
##   0.10       2                2500      0.8271737  0.6331855
##   0.10       2                2650      0.8215554  0.6197003
##   0.10       2                2800      0.8193207  0.6156273
##   0.10       2                2950      0.8204194  0.6164674
##   0.10       2                3100      0.8193083  0.6146201
##   0.10       2                3250      0.8159499  0.6073616
##   0.10       2                3400      0.8181847  0.6112685
##   0.10       2                3550      0.8170608  0.6102279
##   0.10       2                3700      0.8148139  0.6047687
##   0.10       2                3850      0.8136903  0.6016777
##   0.10       2                4000      0.8159499  0.6081036
##   0.10       2                4150      0.8170611  0.6097059
##   0.10       2                4300      0.8159499  0.6080508
##   0.10       2                4450      0.8148011  0.6048848
##   0.10       2                4600      0.8159375  0.6090339
##   0.10       2                4750      0.8170486  0.6106319
##   0.10       2                4900      0.8148014  0.6057887
##   0.10       2                5050      0.8148014  0.6061774
##   0.10       2                5200      0.8125667  0.6010796
##   0.10       2                5350      0.8103445  0.5953160
##   0.10       2                5500      0.8159375  0.6077410
##   0.10       2                5650      0.8103320  0.5970509
```

```
##   0.10       2                5800     0.8114556  0.5993502
##   0.10       2                5950     0.8103320  0.5964987
##   0.10       3                 100     0.8474370  0.6744876
##   0.10       3                 250     0.8496714  0.6783144
##   0.10       3                 400     0.8451390  0.6692738
##   0.10       3                 550     0.8440030  0.6675088
##   0.10       3                 700     0.8439777  0.6669297
##   0.10       3                 850     0.8372608  0.6529906
##   0.10       3                1000     0.8305695  0.6379274
##   0.10       3                1150     0.8260373  0.6281384
##   0.10       3                1300     0.8226793  0.6227373
##   0.10       3                1450     0.8249262  0.6266918
##   0.10       3                1600     0.8137152  0.6018075
##   0.10       3                1750     0.8193460  0.6146965
##   0.10       3                1900     0.8170863  0.6094098
##   0.10       3                2050     0.8171113  0.6086253
##   0.10       3                2200     0.8193460  0.6140868
##   0.10       3                2350     0.8137280  0.6028419
##   0.10       3                2500     0.8159752  0.6080767
##   0.10       3                2650     0.8159752  0.6078186
##   0.10       3                2800     0.8182349  0.6118903
##   0.10       3                2950     0.8103694  0.5962781
##   0.10       3                3100     0.8103822  0.5968371
##   0.10       3                3250     0.8114805  0.5984587
##   0.10       3                3400     0.8148513  0.6063689
##   0.10       3                3550     0.8047512  0.5854723
##   0.10       3                3700     0.8081350  0.5919172
##   0.10       3                3850     0.8092583  0.5928699
##   0.10       3                4000     0.8070111  0.5898314
##   0.10       3                4150     0.8092331  0.5940059
##   0.10       3                4300     0.8058623  0.5870373
##   0.10       3                4450     0.8036151  0.5823450
##   0.10       3                4600     0.8036276  0.5831142
##   0.10       3                4750     0.8013804  0.5778372
##   0.10       3                4900     0.8047512  0.5853322
##   0.10       3                5050     0.8047512  0.5865208
##   0.10       3                5200     0.8047387  0.5852454
##   0.10       3                5350     0.8002568  0.5750047
##   0.10       3                5500     0.8025165  0.5796618
##   0.10       3                5650     0.8013929  0.5781300
##   0.10       3                5800     0.8013929  0.5789816
##   0.10       3                5950     0.7991582  0.5737374
##   0.10       4                 100     0.8362002  0.6502630
##   0.10       4                 250     0.8361755  0.6508341
##   0.10       4                 400     0.8372988  0.6516561
##   0.10       4                 550     0.8327917  0.6435465
##   0.10       4                 700     0.8249515  0.6271206
##   0.10       4                 850     0.8249262  0.6286422
##   0.10       4                1000     0.8170735  0.6112602
##   0.10       4                1150     0.8215804  0.6205173
##   0.10       4                1300     0.8182099  0.6121368
##   0.10       4                1450     0.8227040  0.6226709
##   0.10       4                1600     0.8182349  0.6130646
##   0.10       4                1750     0.8115058  0.5986282
```

```
##   0.10       4                  1900     0.8171238  0.6111703
##   0.10       4                  2050     0.8148766  0.6061489
##   0.10       4                  2200     0.8160127  0.6091493
##   0.10       4                  2350     0.8148891  0.6059448
##   0.10       4                  2500     0.8137657  0.6043172
##   0.10       4                  2650     0.8092711  0.5952445
##   0.10       4                  2800     0.8103572  0.5981942
##   0.10       4                  2950     0.8103822  0.5980832
##   0.10       4                  3100     0.8047642  0.5864820
##   0.10       4                  3250     0.8047642  0.5860478
##   0.10       4                  3400     0.8058878  0.5883095
##   0.10       4                  3550     0.8036406  0.5838100
##   0.10       4                  3700     0.8036281  0.5840120
##   0.10       4                  3850     0.8036531  0.5840990
##   0.10       4                  4000     0.8025295  0.5804073
##   0.10       4                  4150     0.8014059  0.5781266
##   0.10       4                  4300     0.7968988  0.5685645
##   0.10       4                  4450     0.7980348  0.5708785
##   0.10       4                  4600     0.7979846  0.5699907
##   0.10       4                  4750     0.7979846  0.5714908
##   0.10       4                  4900     0.7980351  0.5717250
##   0.10       4                  5050     0.8047392  0.5863036
##   0.10       4                  5200     0.8024918  0.5813963
##   0.10       4                  5350     0.8013931  0.5793365
##   0.10       4                  5500     0.8058753  0.5888091
##   0.10       4                  5650     0.7957752  0.5668205
##   0.10       4                  5800     0.8013682  0.5784832
##   0.10       4                  5950     0.7991085  0.5731928
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 250,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

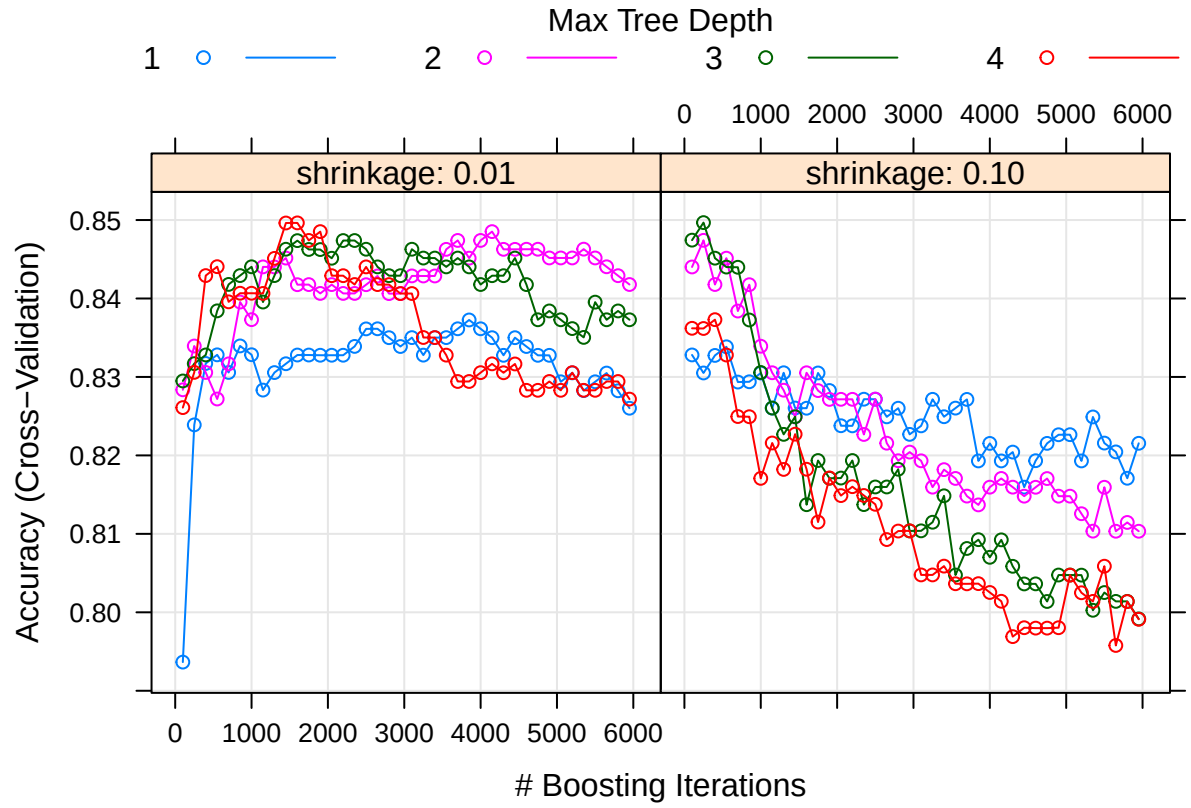
```
max(boost.model$results$Accuracy)
```

```
## [1] 0.8496714
```

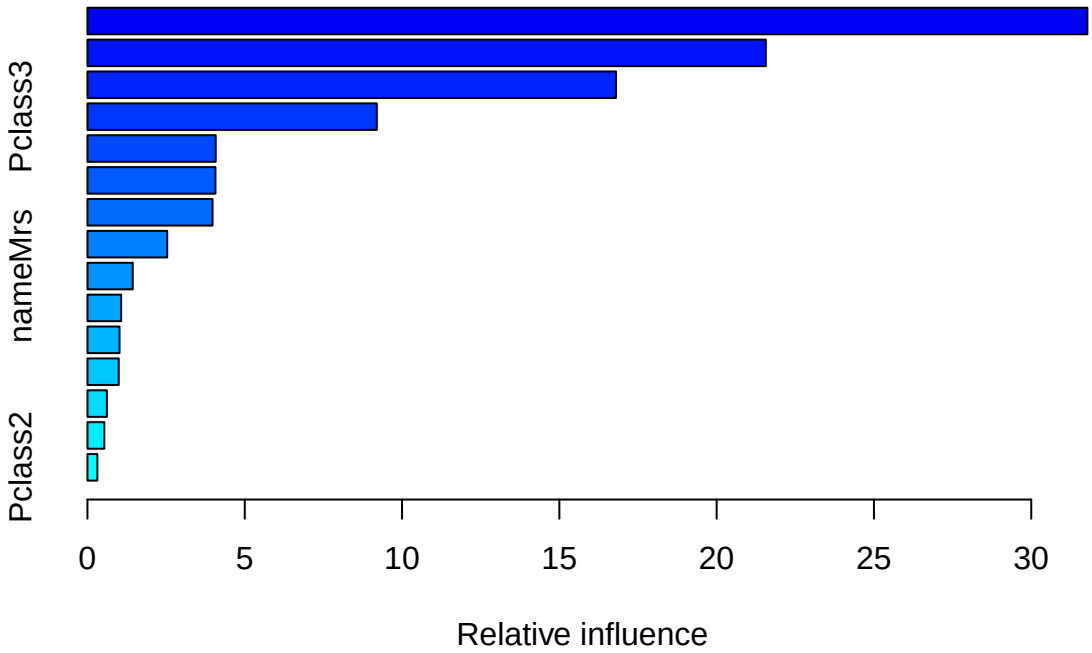
```
#84.44%

boost.minacc <- max(boost.model$results$Accuracy) -
  boost.model$results$AccuracySD[which.max(boost.model$results$Accuracy)]
boost.minacc
```

```
## [1] 0.8137223
```

```
#81.28%

plot(boost.model)
```




```
summary(boost.model$finalModel)
```




```
##                                 var    rel.inf
## nameMr                       nameMr 31.7842778
## Fare                           Fare 21.5639163
## Age                             Age 16.8001393
## Pclass3                     Pclass3  9.1984825
```

```
## ticket.let.survlow     ticket.let.survlow  4.0768864
## familyGrouplarge fam familyGrouplarge fam  4.0685765
## Sex1                                 Sex1  3.9759654
## cabin.freq.survlow     cabin.freq.survlow  2.5356483
## nameMrs                         nameMrs  1.4406745
## deck.survlow                 deck.survlow  1.0716527
## familyGroupsmall fam familyGroupsmall fam  1.0196897
## nameMiss                       nameMiss  0.9939019
## ticket.alone1               ticket.alone1  0.6182421
## Cabin.ox1                      Cabin.ox1  0.5369445
## Pclass2                          Pclass2  0.3150022
```

```
boost.model$finalModel$tuneValue$n.trees
```

```
## [1] 250
```

```
#predict on training
boost.pred <- predict(boost.model, training,
                      n.trees=boost.model$finalModel$tuneValue$n.trees)


confusionMatrix(boost.pred, training$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 513  63
##          1  36 279
##
##                Accuracy : 0.8889
##                  95% CI : (0.8664, 0.9088)
##     No Information Rate : 0.6162
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7616
##
##  Mcnemar's Test P-Value : 0.008973
##
##             Sensitivity : 0.9344
##             Specificity : 0.8158
##          Pos Pred Value : 0.8906
##          Neg Pred Value : 0.8857
##              Prevalence : 0.6162
##          Detection Rate : 0.5758
##    Detection Prevalence : 0.6465
##       Balanced Accuracy : 0.8751
##
##        'Positive' Class : 0
##
```

```
#88.78%

acc(boost.pred, training$Survived, training) - max(boost.model$results$Accuracy)
```

```
## [1] 0.03921746
```

```
#0.0437
```

SVM - kernel radial

```
svm.radial <- model("svmRadial", training, control, grid = NULL, tuneLength = 10)
svm.radial
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 891 samples
##  13 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 802, 801, 802, 802, 802, 801, ...
## Resampling results across tuning parameters:
##
##   C       Accuracy   Kappa
##     0.25  0.8294093  0.6335026
##     0.50  0.8294342  0.6307068
##     1.00  0.8293843  0.6280843
##     2.00  0.8226552  0.6153207
##     4.00  0.8170367  0.6055965
##     8.00  0.8147268  0.6006760
##    16.00  0.8080226  0.5865144
##    32.00  0.8057630  0.5832370
##    64.00  0.7956881  0.5607242
##   128.00  0.7946019  0.5594057
##
## Tuning parameter 'sigma' was held constant at a value of 0.05507806
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.05507806 and C = 0.5.
```

```
max(svm.radial$results$Accuracy)
```

```
## [1] 0.8294342
```

```
#83.16%

varImp(svm.radial)
```

```
## ROC curve variable importance
##
##                   Importance
## Sex                  100.000
## Fare                  68.444
## Pclass                63.925
## Cabin.ox              45.132
## cabin.freq.surv       44.666
## deck.surv             43.806
## ticket.let.surv       42.370
## ticket.alone          42.056
## familyGroup           39.028
```

```
## Embarked              20.071
## surname.freq.surv     19.463
## Age                    1.246
## name                   0.000
```

```
#name and Age

#Grid Search for tuning parameter
svm.grid <- expand.grid(sigma = seq(0.01,0.1, by=0.01),
                        C = seq(0.01,2.01,by=0.25))

svm.radial <- model("svmRadial", training %>% subset(select = -c(name, Age)),
                    control,
                    grid = svm.grid)

svm.radial
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 891 samples
##  11 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 801, 802, 803, 802, 802, 801, ...
## Resampling results across tuning parameters:
##
##   sigma  C     Accuracy   Kappa
##   0.01   0.01  0.6161701  0.0000000
##   0.01   0.26  0.8037303  0.5729768
##   0.01   0.51  0.8082247  0.5808712
##   0.01   0.76  0.8082247  0.5808712
##   0.01   1.01  0.8104719  0.5864336
##   0.01   1.26  0.8104719  0.5864336
##   0.01   1.51  0.8104719  0.5864336
##   0.01   1.76  0.8104719  0.5864336
##   0.01   2.01  0.8104719  0.5864336
##   0.02   0.01  0.6161701  0.0000000
##   0.02   0.26  0.8071011  0.5796396
##   0.02   0.51  0.8082247  0.5819127
##   0.02   0.76  0.8093483  0.5842128
##   0.02   1.01  0.8104719  0.5864336
##   0.02   1.26  0.8104719  0.5864336
##   0.02   1.51  0.8104719  0.5864336
##   0.02   1.76  0.8093483  0.5841335
##   0.02   2.01  0.8071011  0.5796136
##   0.03   0.01  0.6161701  0.0000000
##   0.03   0.26  0.8026067  0.5707815
##   0.03   0.51  0.8059775  0.5773928
##   0.03   0.76  0.8026067  0.5708066
##   0.03   1.01  0.8048539  0.5756475
##   0.03   1.26  0.8026067  0.5712820
##   0.03   1.51  0.8048539  0.5756502
##   0.03   1.76  0.8048539  0.5756502
```

```
##   0.03   2.01  0.8048539  0.5756502
##   0.04   0.01  0.6161701  0.0000000
##   0.04   0.26  0.8003595  0.5664411
##   0.04   0.51  0.8014831  0.5691090
##   0.04   0.76  0.8014831  0.5691090
##   0.04   1.01  0.8014831  0.5691090
##   0.04   1.26  0.8025942  0.5717503
##   0.04   1.51  0.7992106  0.5653191
##   0.04   1.76  0.7958143  0.5558104
##   0.04   2.01  0.7946780  0.5535132
##   0.05   0.01  0.6161701  0.0000000
##   0.05   0.26  0.7947412  0.5565897
##   0.05   0.51  0.7981123  0.5628675
##   0.05   0.76  0.7981123  0.5628675
##   0.05   1.01  0.7969635  0.5609917
##   0.05   1.26  0.7924308  0.5496791
##   0.05   1.51  0.7935544  0.5501331
##   0.05   1.76  0.7935544  0.5495047
##   0.05   2.01  0.7912819  0.5420437
##   0.06   0.01  0.6161701  0.0000000
##   0.06   0.26  0.7924813  0.5526895
##   0.06   0.51  0.7947160  0.5561428
##   0.06   0.76  0.7901836  0.5454972
##   0.06   1.01  0.7913072  0.5475547
##   0.06   1.26  0.7913072  0.5451220
##   0.06   1.51  0.7901583  0.5398794
##   0.06   1.76  0.7924180  0.5425542
##   0.06   2.01  0.7890472  0.5342668
##   0.07   0.01  0.6161701  0.0000000
##   0.07   0.26  0.7924813  0.5532147
##   0.07   0.51  0.7947160  0.5564783
##   0.07   0.76  0.7924308  0.5501520
##   0.07   1.01  0.7935419  0.5511574
##   0.07   1.26  0.7912819  0.5424796
##   0.07   1.51  0.7912819  0.5408410
##   0.07   1.76  0.7912944  0.5398198
##   0.07   2.01  0.7957763  0.5503732
##   0.08   0.01  0.6161701  0.0000000
##   0.08   0.26  0.7913577  0.5515541
##   0.08   0.51  0.7924433  0.5505581
##   0.08   0.76  0.7913072  0.5468071
##   0.08   1.01  0.7868000  0.5337435
##   0.08   1.26  0.7912944  0.5397601
##   0.08   1.51  0.7924055  0.5437206
##   0.08   1.76  0.7935291  0.5465587
##   0.08   2.01  0.7957513  0.5509323
##   0.09   0.01  0.6161701  0.0000000
##   0.09   0.26  0.7969376  0.5655970
##   0.09   0.51  0.7913197  0.5479891
##   0.09   0.76  0.7912944  0.5448166
##   0.09   1.01  0.7912944  0.5409335
##   0.09   1.26  0.7912944  0.5415371
##   0.09   1.51  0.7912944  0.5422148
##   0.09   1.76  0.7935166  0.5465883
```

```
##   0.09   2.01  0.7980113  0.5573872
##   0.10   0.01  0.6161701  0.0000000
##   0.10   0.26  0.7980487  0.5679283
##   0.10   0.51  0.7901961  0.5443093
##   0.10   0.76  0.7913069  0.5431966
##   0.10   1.01  0.7912944  0.5414431
##   0.10   1.26  0.7935166  0.5465883
##   0.10   1.51  0.7957766  0.5518517
##   0.10   1.76  0.7980238  0.5580077
##   0.10   2.01  0.7980238  0.5584190
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.02 and C = 1.01.
```

```
max(svm.radial$results$Accuracy)
```

```
## [1] 0.8104719
```

```
#0.8160

#on training
svm.radial.pred <- predict(svm.radial, training)

confusionMatrix(svm.radial.pred, training$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 492 112
##          1  57 230
##
##                Accuracy : 0.8103
##                  95% CI : (0.783, 0.8356)
##     No Information Rate : 0.6162
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5865
##
##  Mcnemar's Test P-Value : 3.269e-05
##
##             Sensitivity : 0.8962
##             Specificity : 0.6725
##          Pos Pred Value : 0.8146
##          Neg Pred Value : 0.8014
##              Prevalence : 0.6162
##          Detection Rate : 0.5522
##    Detection Prevalence : 0.6779
##       Balanced Accuracy : 0.7843
##
##        'Positive' Class : 0
##
```

```
#0.8395

acc(svm.radial.pred, training$Survived, training) - max(svm.radial$results$Accuracy)
```

```
## [1] -0.0001463764
```

```
#0.0235
```

Ensembling models in a dataset

```
#prediction on test

rf.test.pred <- predict(rf.model, testing)
boost.test.pred <- predict(boost.model, testing)
svm.radial.pred <- predict(svm.radial, testing)

ensembled.test <- data.frame(PassengerId = test$PassengerId,
                             rf = rf.test.pred,
                             boost= boost.test.pred,
                             svm = svm.radial.pred)

#Take average of the predicting value by 3 models : Random Forest / Gradient Boosting / SVM - Radial
ensembled.test$mean <- as.factor(round((as.numeric(ensembled.test$rf) +
                                        as.numeric(ensembled.test$boost) +
                                        as.numeric(ensembled.test$svm) - 3)/3))

ensembled.test$PassengerId <- as.character(ensembled.test$PassengerId)

summary(ensembled.test)
```

```
##  PassengerId        rf      boost   svm     mean
##  Length:418         0:260   0:260   0:271   0:259
##  Class :character   1:158   1:158   1:147   1:159
##  Mode  :character
```

Creating submission

```
final.pred <- ensembled.test$mean
final.pred
```

```
##   [1] 0 0 0 0 1 0 1 0 1 0 0 0 1 0 1 1 0 0 1 1 0 1 1 0 1 0 1 0 0 0 0 0 1 1 1
##  [36] 0 1 0 0 1 0 1 0 1 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 0 1
##  [71] 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 1 1 0 1 0 0 0 1 0 1 0 1 0 0 0 1
## [106] 0 0 0 0 0 0 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
## [141] 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 1 1 1 1 1 1 0 0 1 0 0 1 0 0 0 0 0 0
## [176] 1 1 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 1 1 0 1 1 1 0 1 0 0 1 0 1 0
## [211] 0 0 0 1 0 0 1 0 1 0 1 0 1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 1
## [246] 0 1 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0
## [281] 0 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 1
## [316] 1 0 0 0 0 0 0 0 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 0 1 0 0 0 1 1 1 0 1 0 1
## [351] 1 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0 0 0 1 0 0 1 0 0 1 1 0 0 0 0 0 0 1 1 0
## [386] 1 0 0 0 0 0 1 1 0 0 1 0 1 0 0 1 0 1 0 0 0 0 0 1 1 1 1 1 0 1 0 0 1
## Levels: 0 1
```

```
final <- data.frame(PassengerId = test$PassengerId, Survived = final.pred)

head(final)
```

```
##   PassengerId Survived
## 1         892        0
## 2         893        0
## 3         894        0
## 4         895        0
## 5         896        1
## 6         897        0
```

```
#write.csv(final, "/Users/DavidKwon/Desktop/Practice/Kaggle/Titanic/final.csv", row.names = FALSE)
```

Public Score - The public score is different by seed, but it's about 78%