Titanic Survivor Prediction

David (Yongbock) Kwon

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

Importing and Manipulating Data - Feature Engineering

```r
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
```

```r
library(ggplot2)
```

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

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

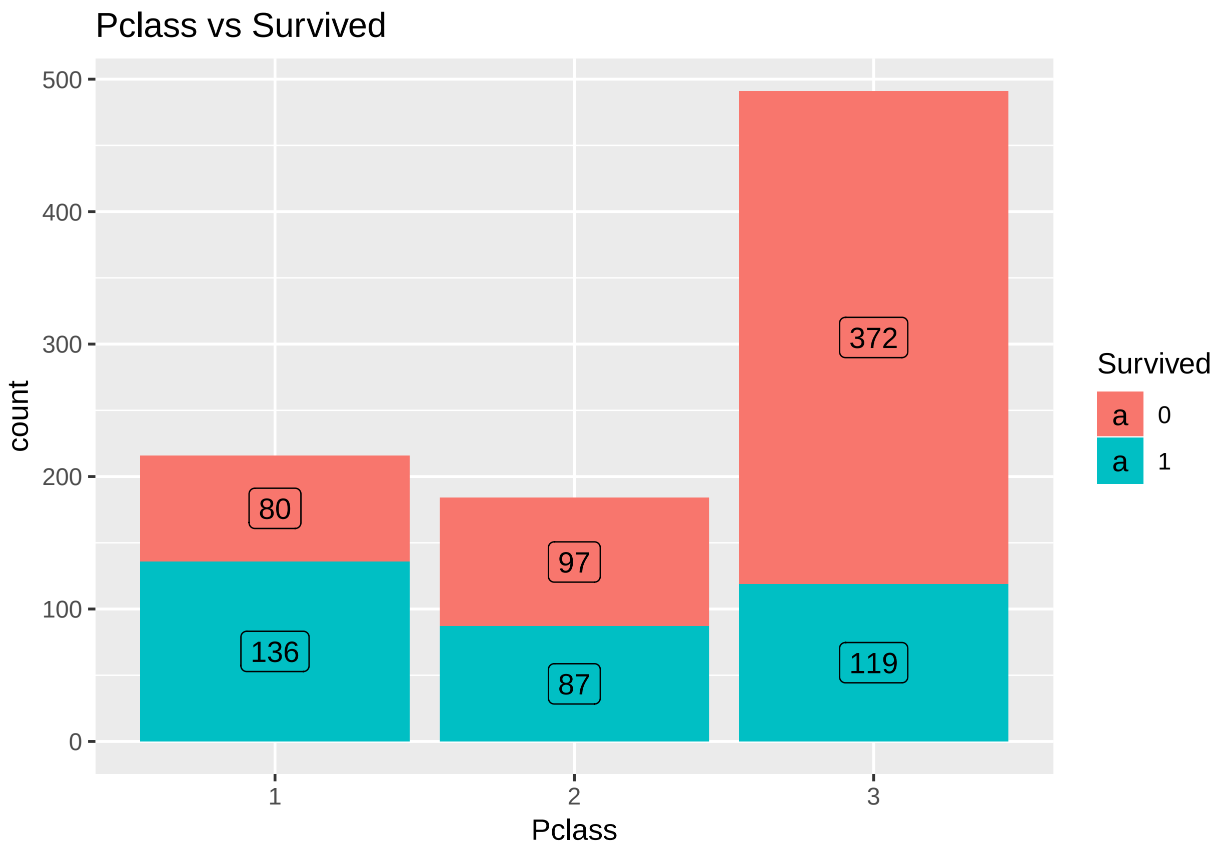
```
## Loading required package: lattice
```

```r
#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)
```

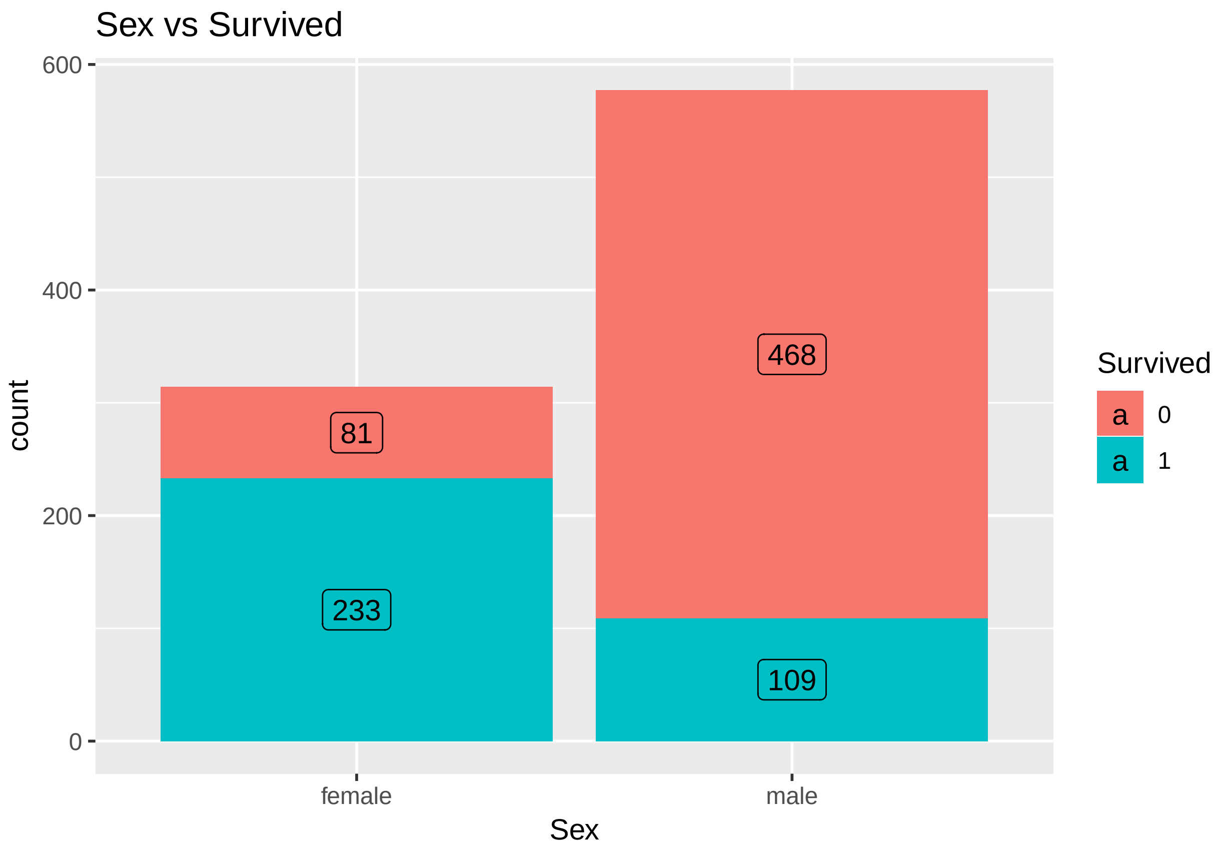
Survived and Pclass

```r
#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)) %>%
```

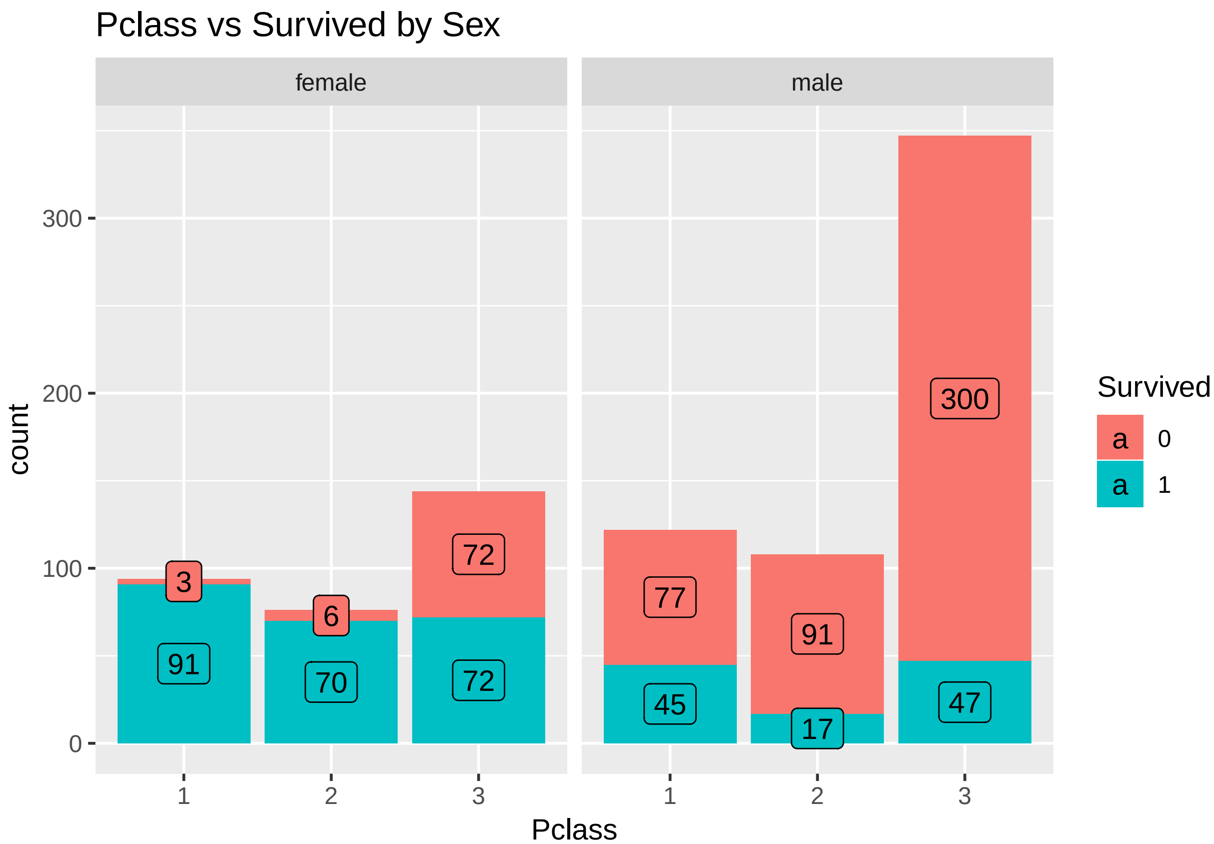
```r
  ggplot(aes(x=Pclass, fill=Survived))+
  geom_bar()+
  geom_label(stat="count",
             position=position_stack(0.5),
             aes(label=..count..))+
  ggtitle("Pclass vs Survived")
```




```r
#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")
```




```r
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)
```



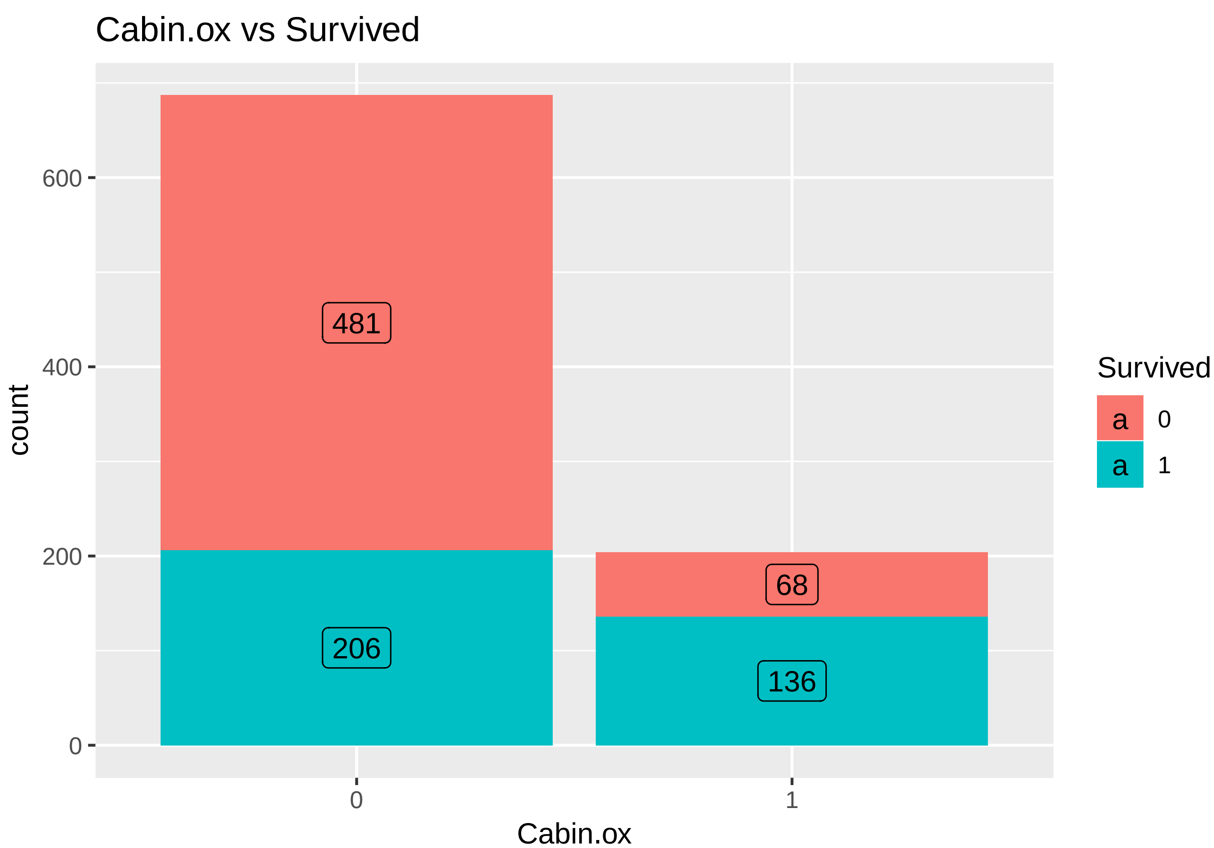

```
#In Pclass 1 and 2, obviously male mostly not survived and female survived
#In Pclass 3, male mostly not survived, but female hard to predict whether surv or not
```

From Cabin, Cabin.ox

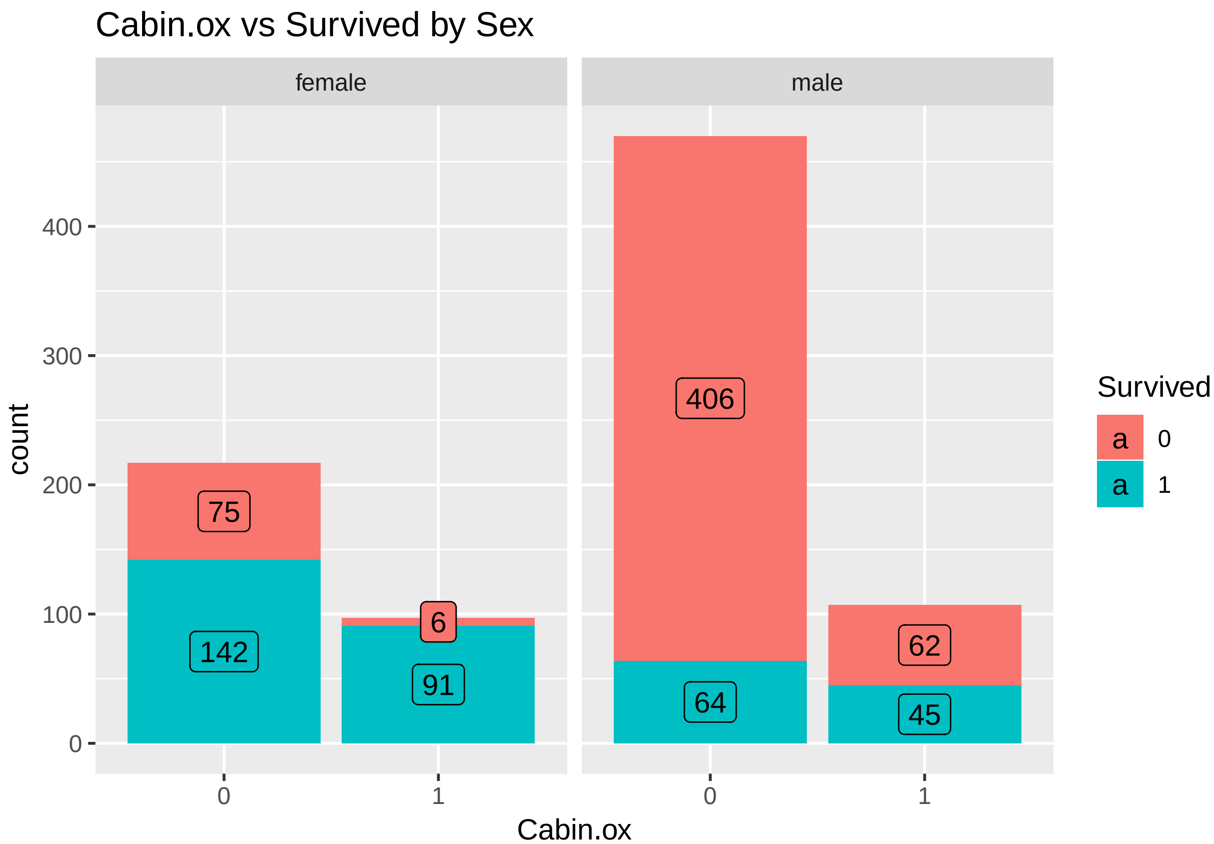
```
#Cabin NA values -> 0, otherwise 1
dat$Cabin.ox <- as.factor(ifelse(is.na(dat$Cabin), 0, 1))
table(dat$Cabin.ox)
```

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

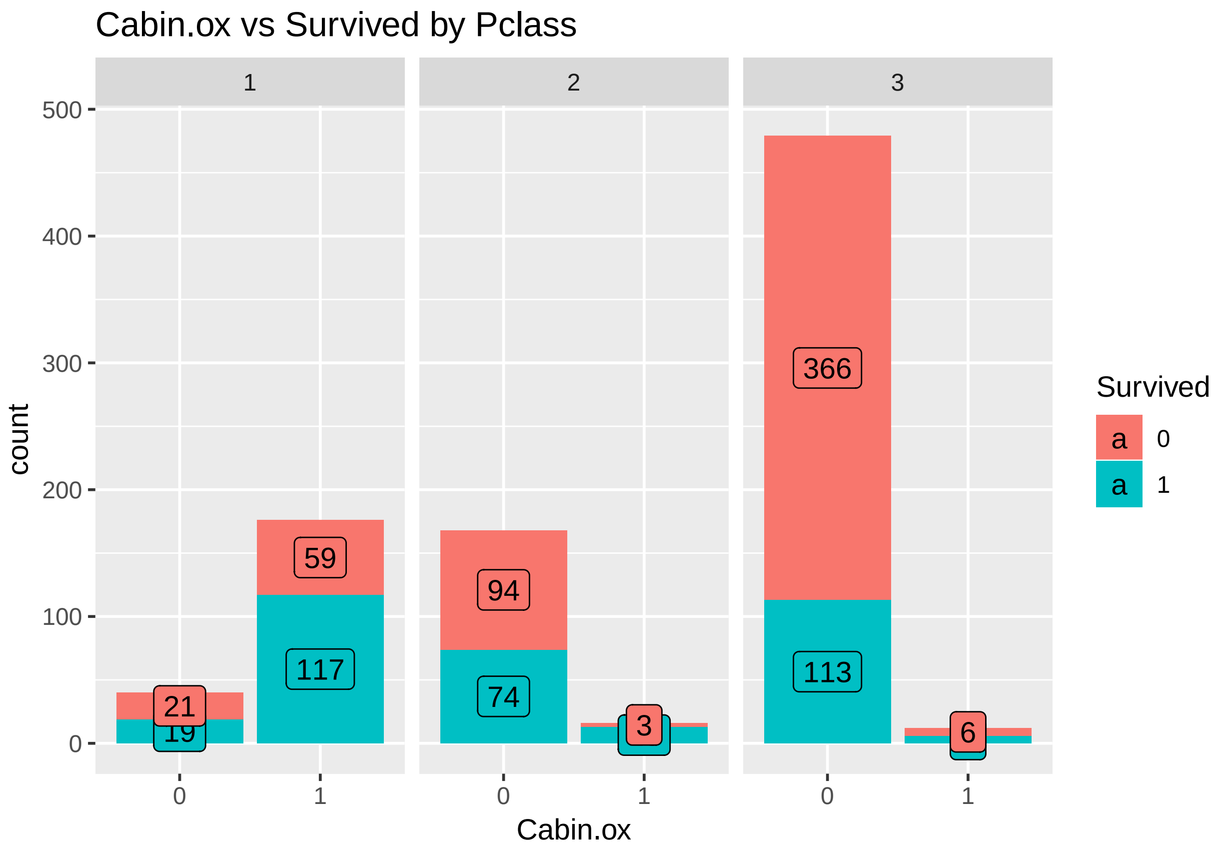
```
#no cabin : 0 / cabin : 1

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")
```



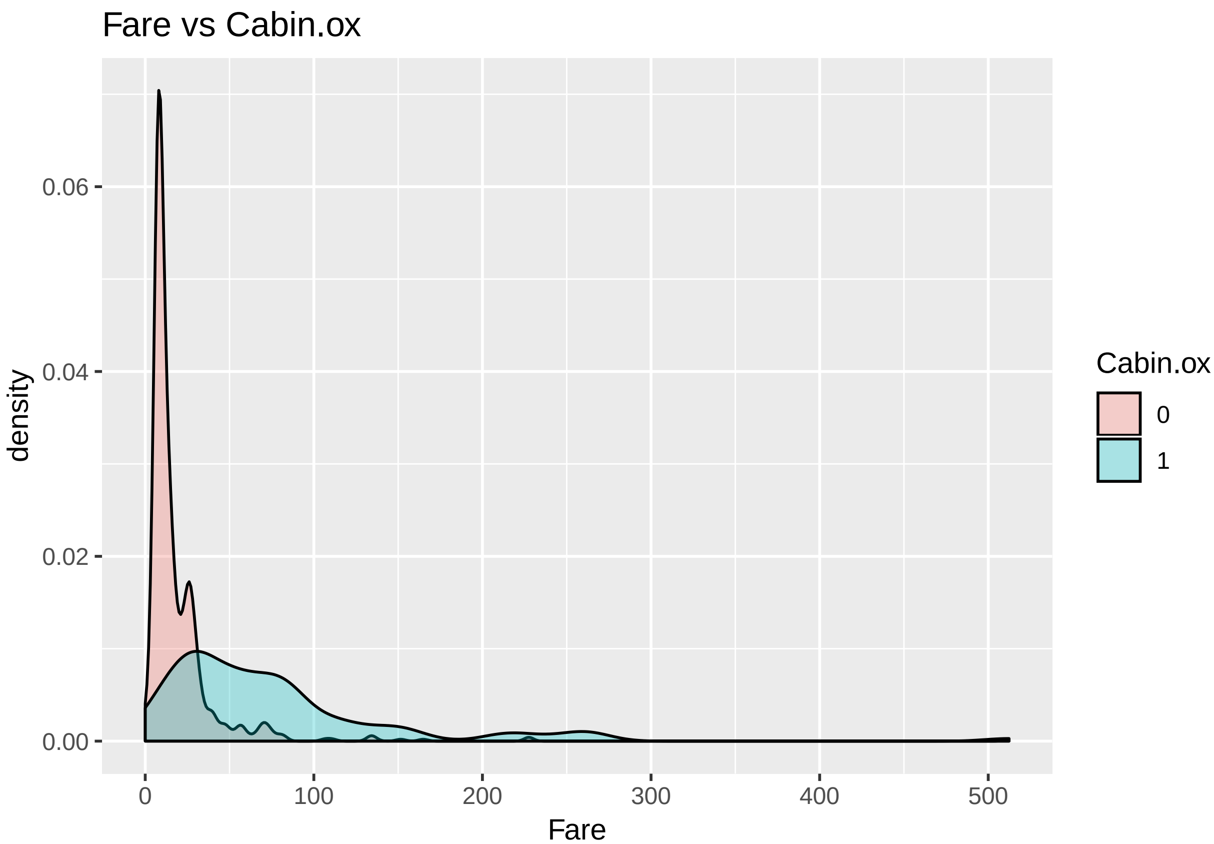

```r
dat %>% filter(!is.na(Survived)) %>%
  ggplot(aes(x=Cabin.ox, fill=Survived))+
  geom_bar()+
  geom_label(stat="count",
             position=position_stack(0.5),
             aes(label=..count..))+
  ggtitle("Cabin.ox vs Survived by Sex")+
  facet_grid(~Sex)
```



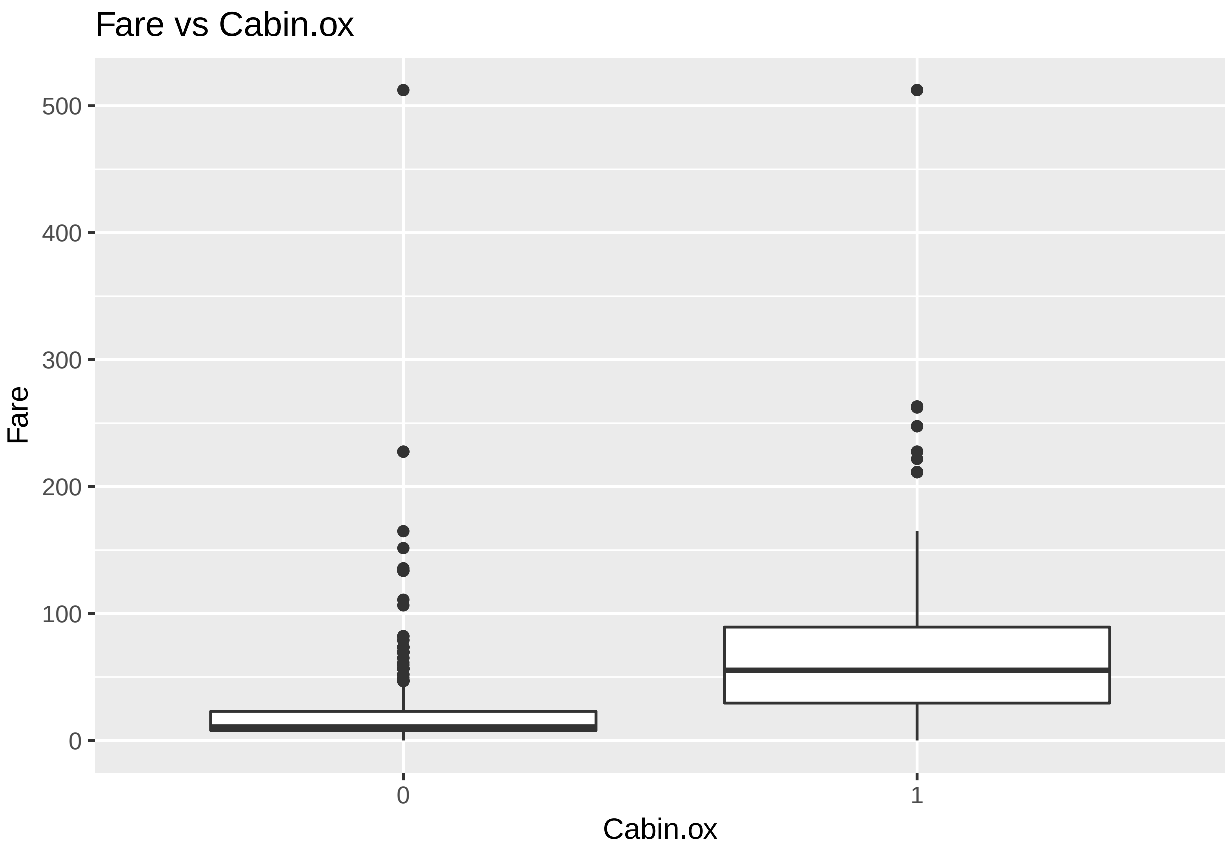

```r
#If Cabin is not NA, then more likely survived
#no cabin likely not survived

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")
```




```r
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
```

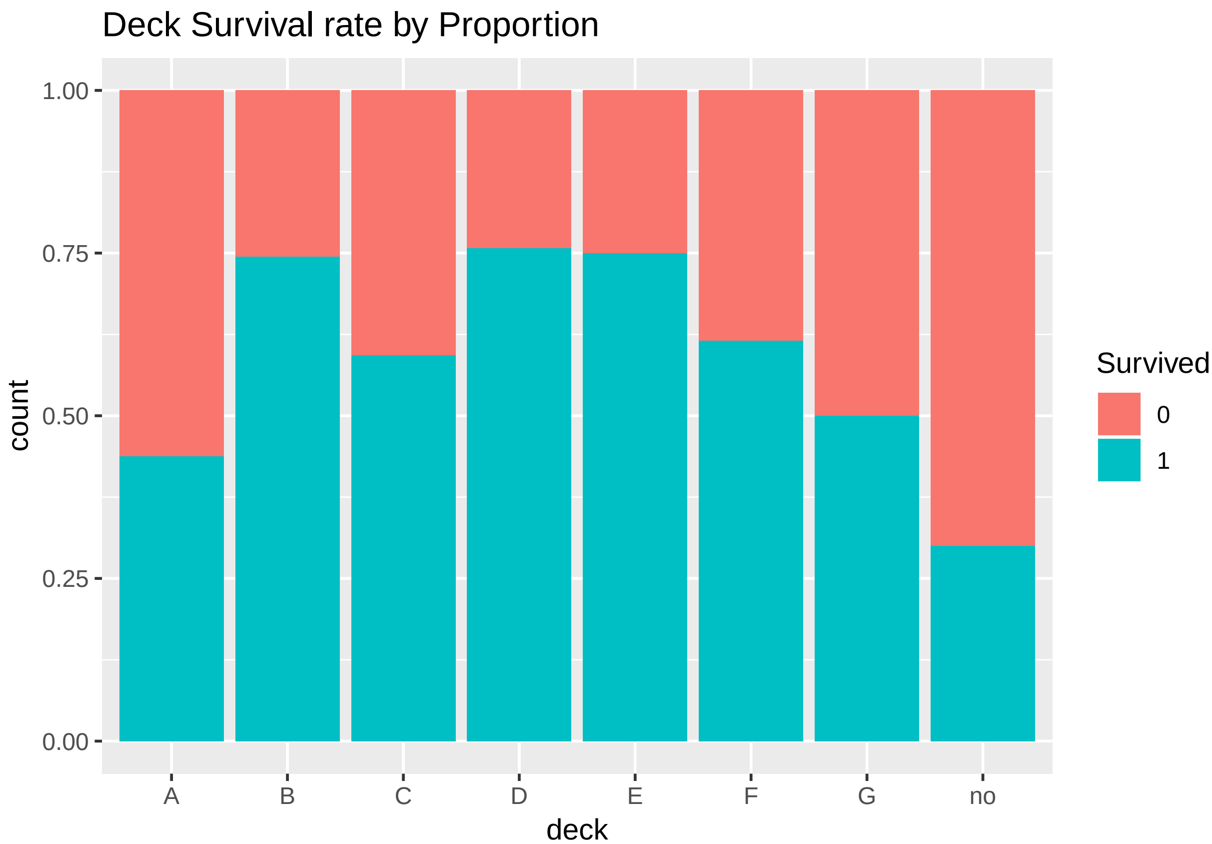
```r
dat %>%
  subset(select = -c(PassengerId)) %>%
  filter(!is.na(Survived)) %>%
  group_by(deck) %>%
  summarise(count = n(),
            mean = mean(Fare))
```

```
## # 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
```

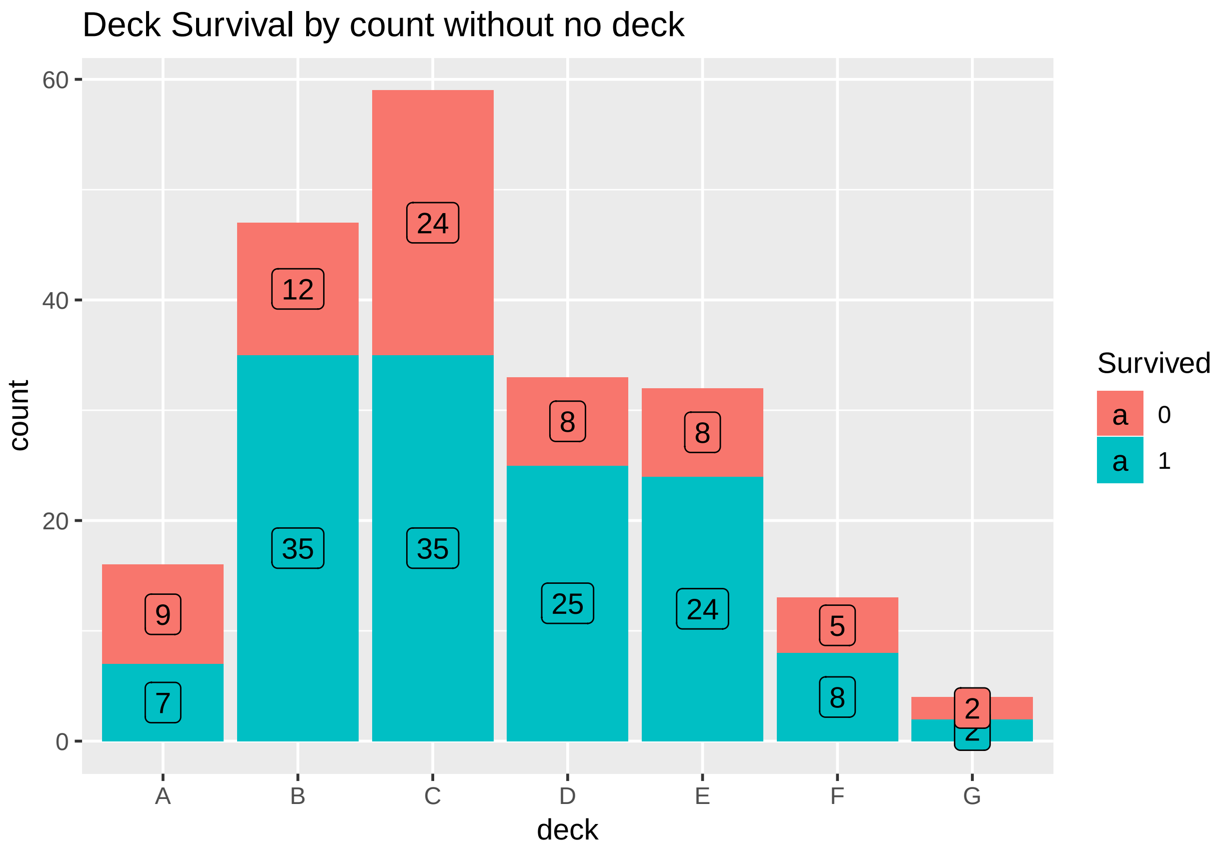
```r
#mean of Fare for deck "T" is close to the mean of Fare for deck "A"
#replace "T" to "A"

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
```

```r
#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
```

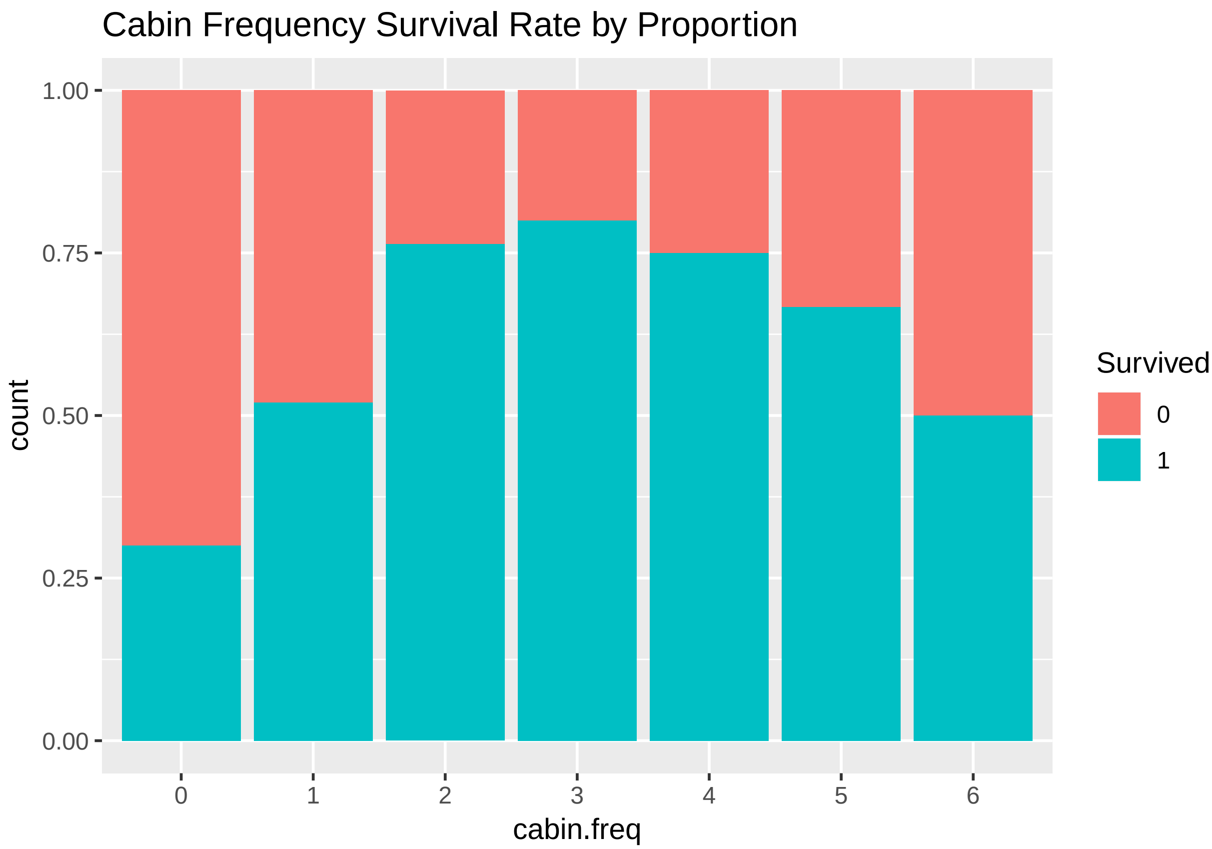
```r
dat$deck.surv <- as.factor(dat$deck.surv)

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

From Cabin, cabin.freq.surv

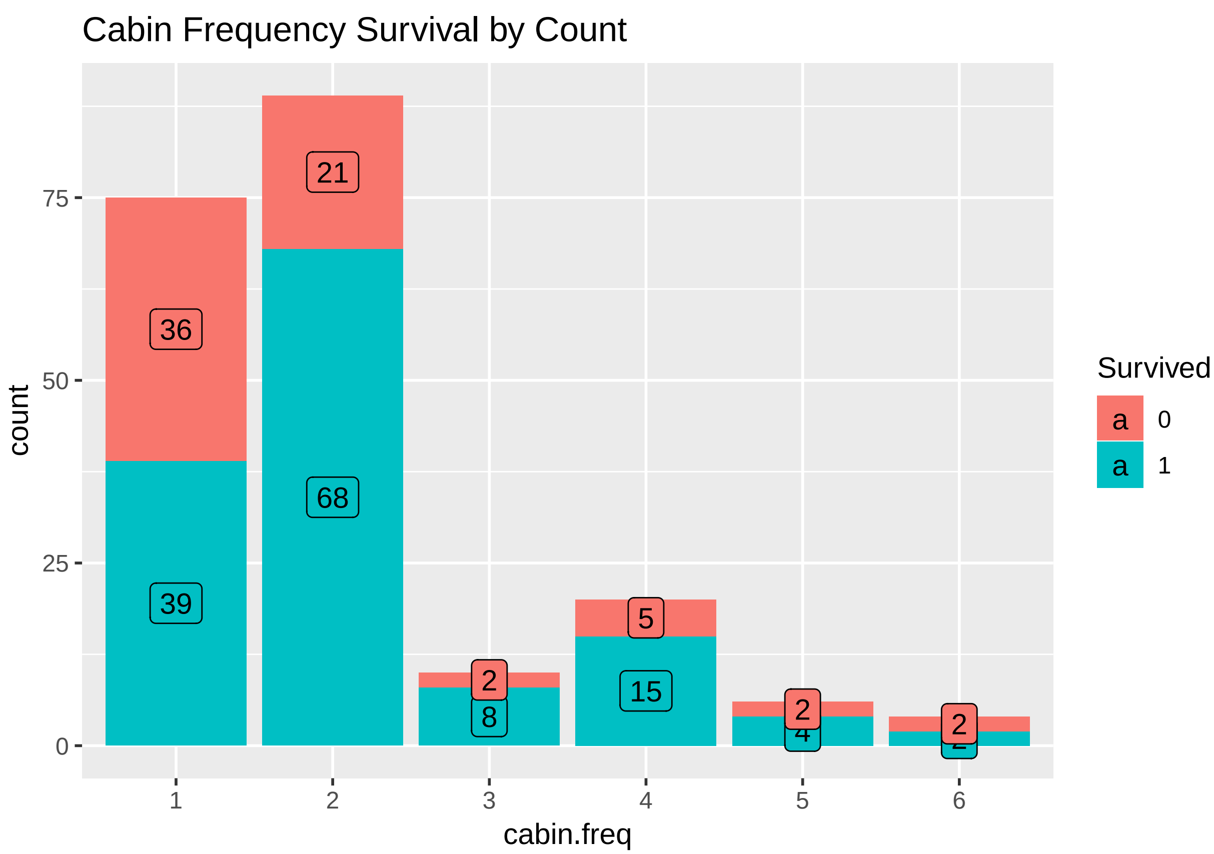
```r
#cabin frequency.. might have relationship between cabin freq
cabin.freq <- data.frame(table(dat$Cabin))


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
```

```r
#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")
```




```r
#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")
```




```r
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
```

```r
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
```

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

```r
#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
```

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

Dealing with NA values in Embarked and Fare

```r
#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
```

```r
#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),
```

```r
            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
```

```r
#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
```

```r
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
```

```r
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
```

```r
#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
```

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

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

From Ticket, ticket.alone

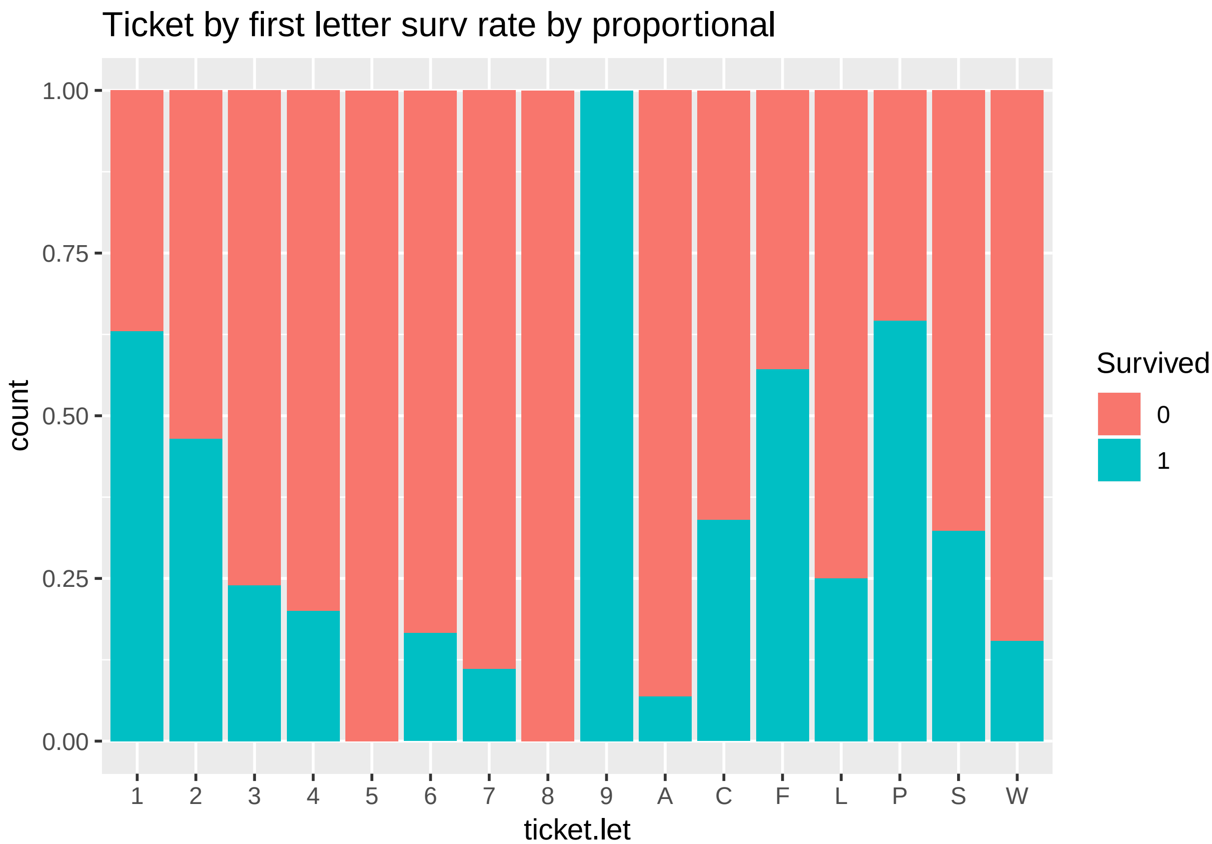
```r
#Ticket
ticket.alone <- data.frame(table(dat$Ticket))

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
```

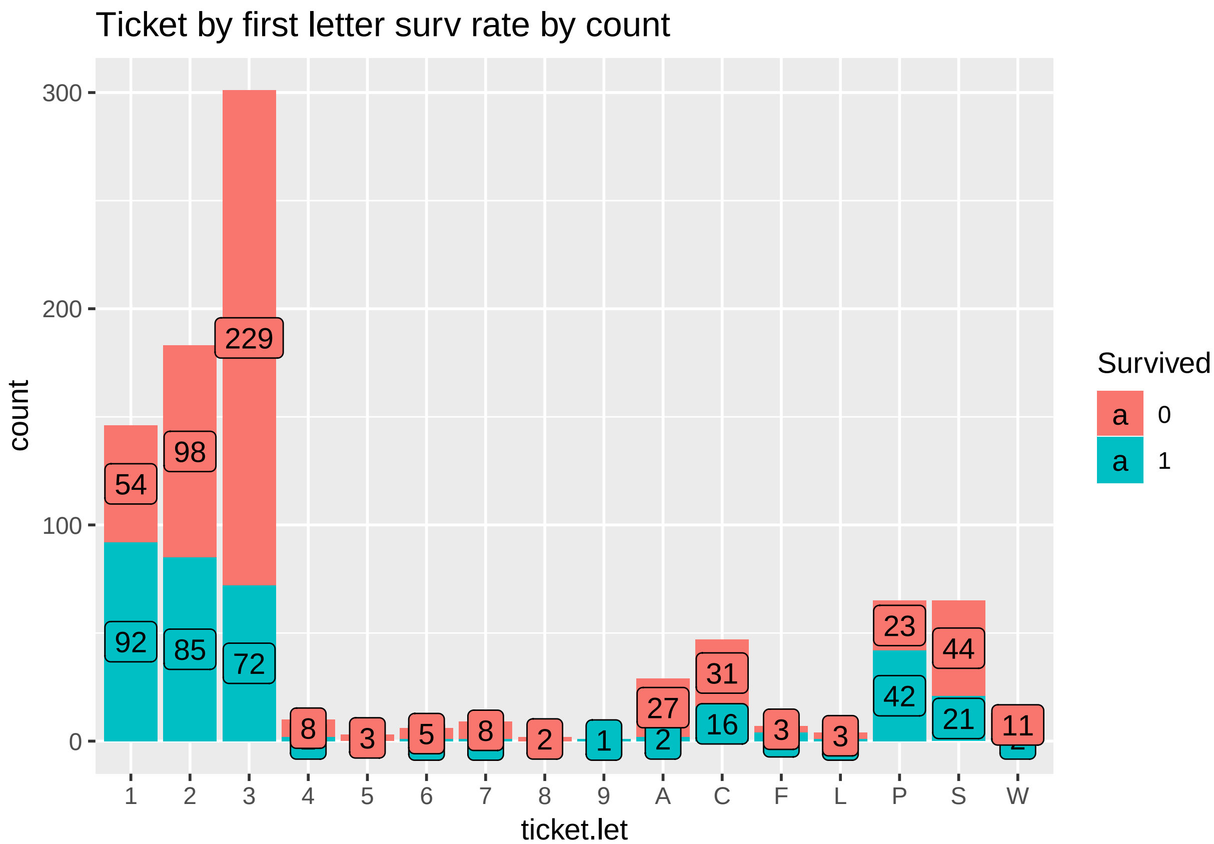
```r
dat$ticket.alone <- as.factor(dat$ticket.alone)
```

From Ticket, ticket.let.surv

```r
#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")
```




```r
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
```

```r
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))
```

```r
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
```

```r
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
```

```
#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
```

```
#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
```

```
#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
```

```r
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
```

```r
#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
```

```r
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))
```

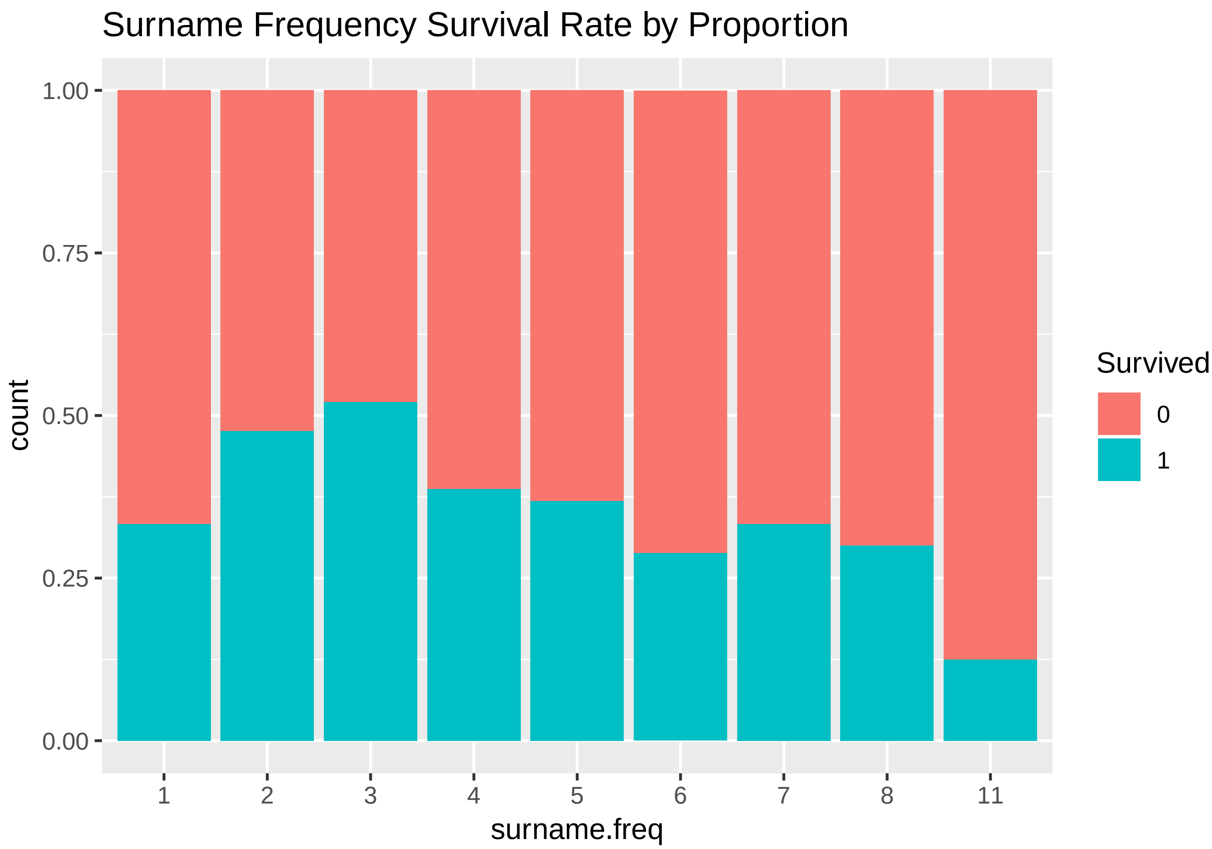
```
## # A tibble: 4 x 5
##   name   count  mean   min   max
##   <chr>  <int> <dbl> <dbl> <dbl>
## 1 Master    61  5.56  0.33  14.5
## 2 Miss     260 21.0   0.17  63
## 3 Mr       757 31.9  11     80
## 4 Mrs      197 36.9  14     76
```

```
#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)){
```

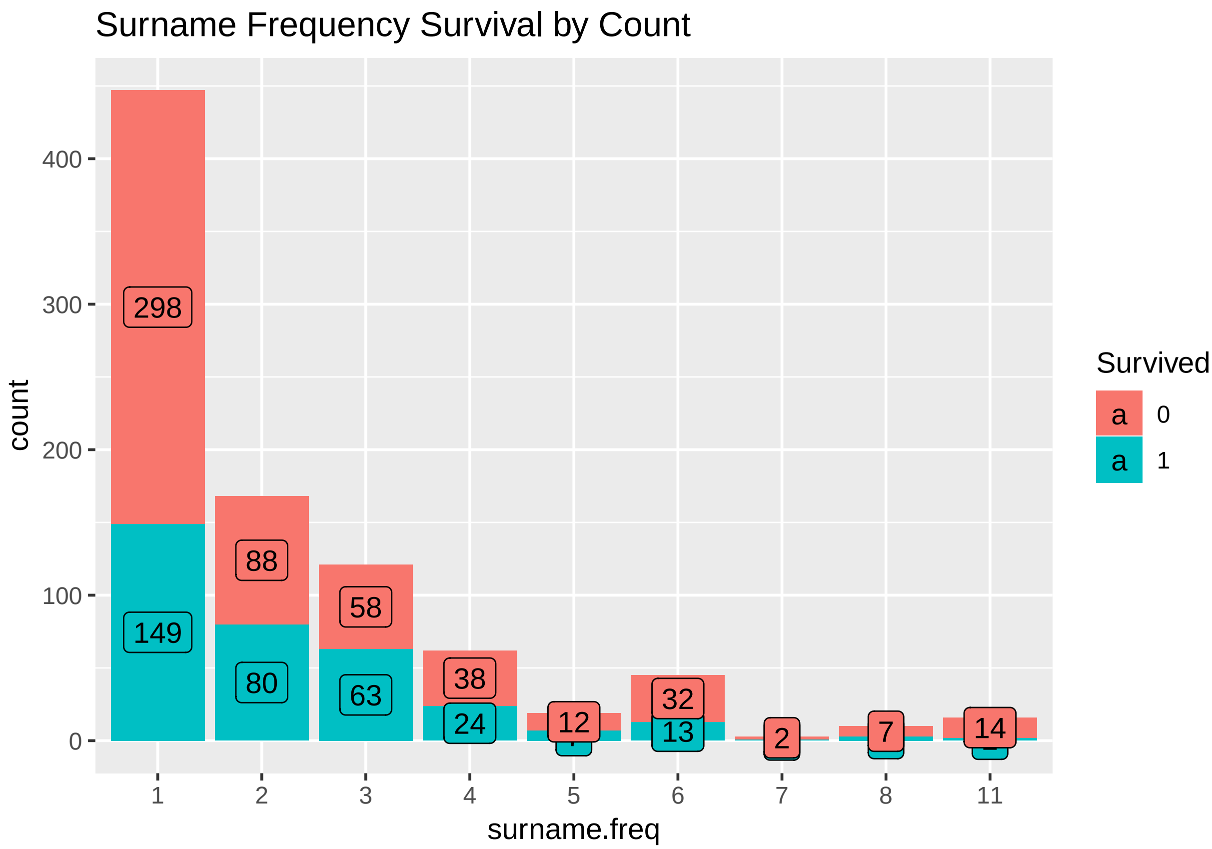
```r
  if(!is.na(dat$Age[i])){
    if(!dat$name[i] %in% c("Mr", "Miss", "Mrs", "Master")){
      if(dat$Sex[i] == 0 & dat$Age[i] <= 14.5){
      dat$name[i] = "Master"
      }
      if(dat$Sex[i] == 0 & dat$Age[i] > 14.5){
      dat$name[i] <- "Mr"
      }
      if(dat$Sex[i] == 1 & dat$Age[i] < 14){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

```r
#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")
```




```r
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")
```




```r
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
```

```r
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

```r
#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)]
```

```r
#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
```

```r
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
```

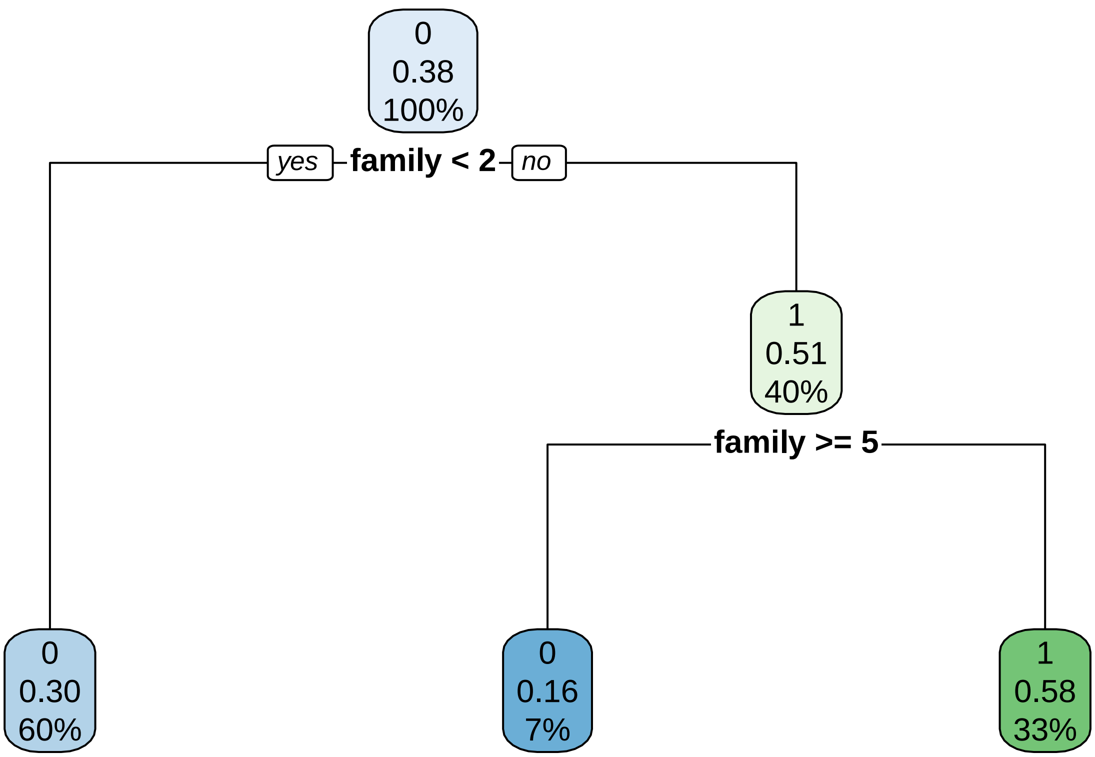
```r
#factor vs factor : if <0.05 (p value), highly dependent, if not, independent
#factor vs numeric : if <0.05, at least one factor has different mean than others.
#if not, all factor has similar mean (non linear)
#numeric vs numeric : if <0.5, low correlation, if not, high correlation
```

Creating familyGroup from investigation of relationship between each variables

```r
#Lets make family to be better predictor

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

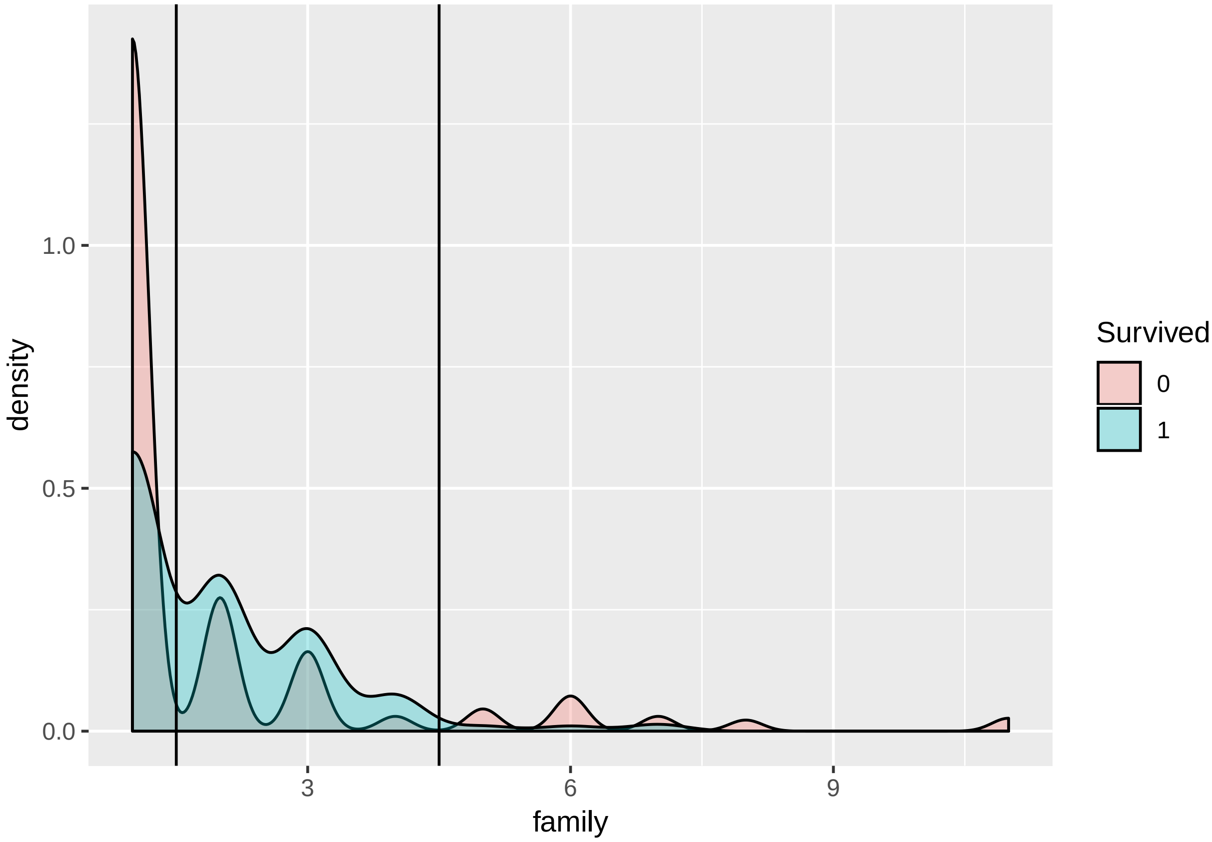
```
## n=891 (418 observations deleted due to missingness)
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 891 342 0 (0.6161616 0.3838384)
##   2) family< 1.5 537 163 0 (0.6964618 0.3035382) *
##   3) family>=1.5 354 175 1 (0.4943503 0.5056497)
##     6) family>=4.5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

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




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

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




```r
#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
```

```r
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
```

```r
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
```

```r
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
##
##
```

```r
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
##
##
```

```r
#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 ——————

```r
#creating function for Caret modeling

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

```r
  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

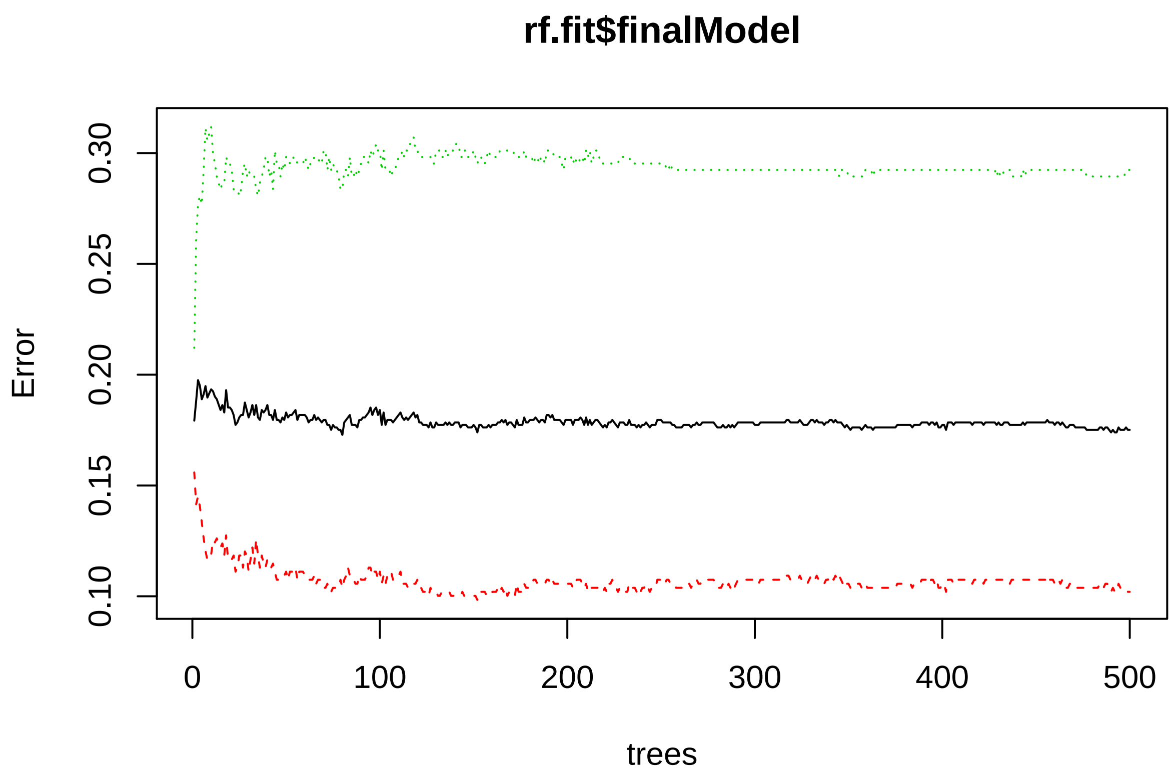
Random Forest

```r
#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, 802, 802, 802, 802, 802, ...
## Resampling results:
##
##   Accuracy   Kappa
```

```
##   0.8271536  0.6253848
##
## Tuning parameter 'mtry' was held constant at a value of 4
```

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



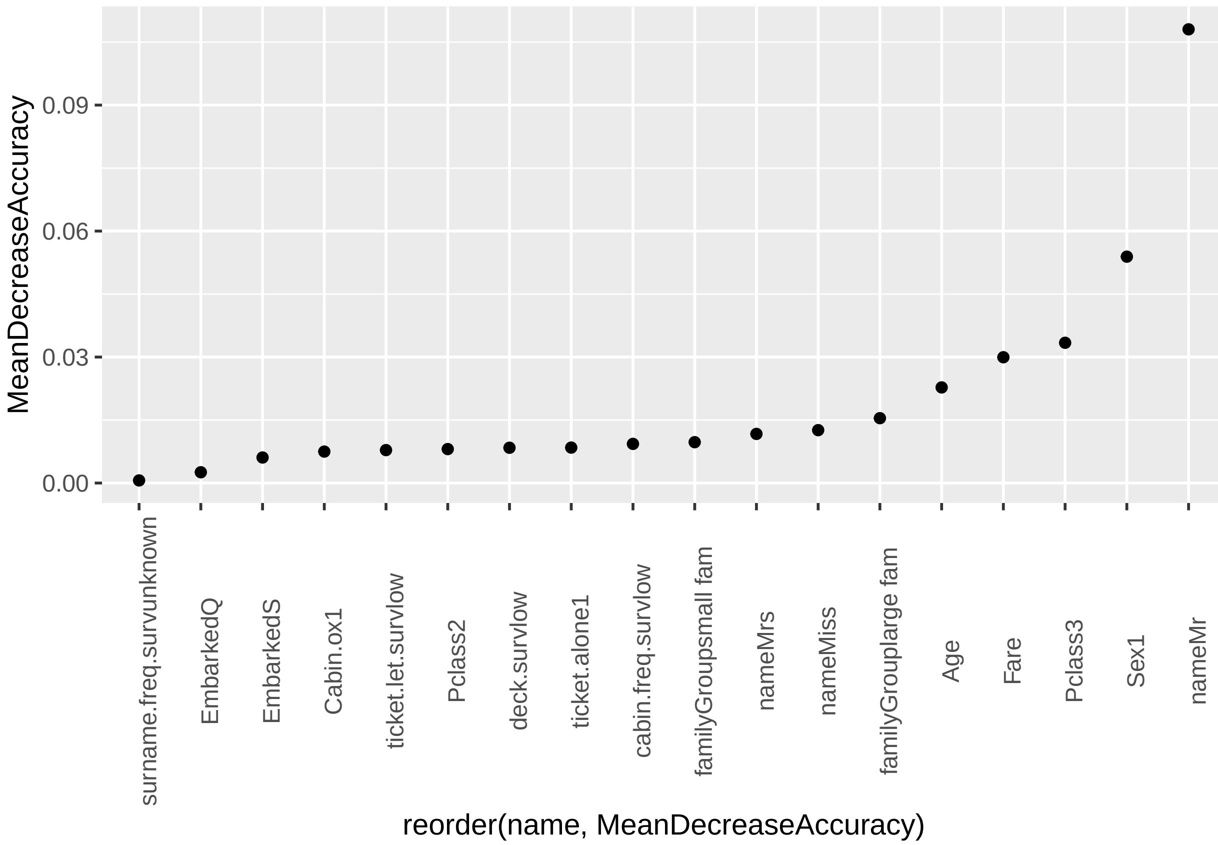

```r
varImp(rf.fit)
```

```
## rf variable importance
##
##                              Importance
## nameMr                           100.00
## Age                               65.44
## Pclass3                           64.74
## Sex1                              61.60
## Fare                              56.59
## familyGrouplarge fam              50.10
## Pclass2                           35.40
## familyGroupsmall fam              33.12
## ticket.alone1                     31.63
## ticket.let.survlow                30.27
## nameMiss                          29.11
## EmbarkedS                         27.53
## nameMrs                           26.90
## cabin.freq.survlow                25.91
## deck.survlow                      21.99
## Cabin.ox1                         19.12
## EmbarkedQ                         11.73
## 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                           0.0080893625
## Pclass3                           0.0334008612
## Sex1                              0.0539083010
## Age                               0.0227714805
## Fare                              0.0299574876
## EmbarkedQ                         0.0025714457
## EmbarkedS                         0.0060811331
## Cabin.ox1                         0.0074908568
## deck.survlow                      0.0084033227
## cabin.freq.survlow                0.0093253743
## ticket.alone1                     0.0084410920
## ticket.let.survlow                0.0078451648
## nameMiss                          0.0125973034
## nameMr                            0.1080448032
## nameMrs                           0.0117025848
## surname.freq.survunknown          0.0006233589
## familyGrouplarge fam              0.0154473416
## familyGroupsmall fam              0.0097350541
```

```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))
```

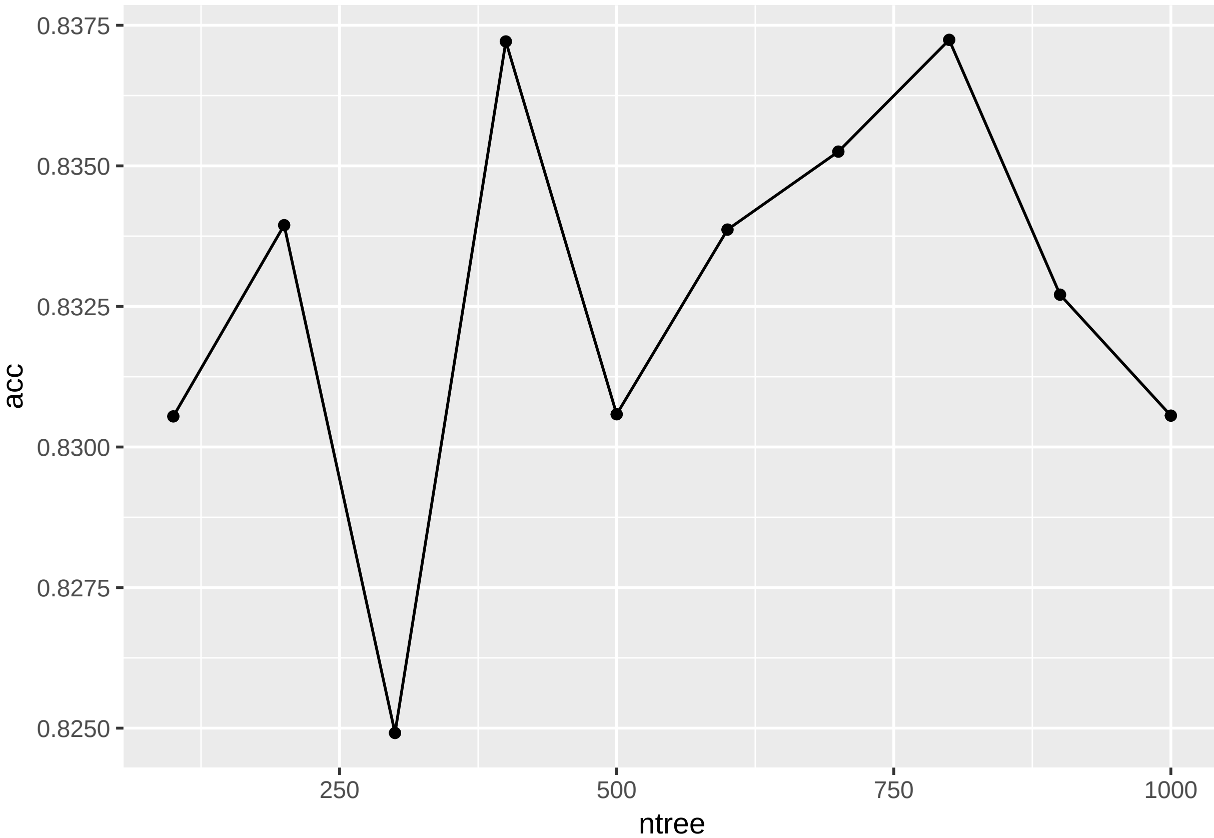



```
#remove Embarked / surname.freq.surv


#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 0.8040275 0.8305445
## 2    200 0.8013228 0.8339451
## 3    300 0.8000004 0.8249146
## 4    400 0.8136984 0.8372120
```

```
## 5    500 0.7922336 0.8305822
## 6    600 0.7956729 0.8338665
## 7    700 0.7801742 0.8352534
## 8    800 0.7940556 0.8372409
## 9    900 0.7896787 0.8327091
## 10  1000 0.8051695 0.8305578
```

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




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

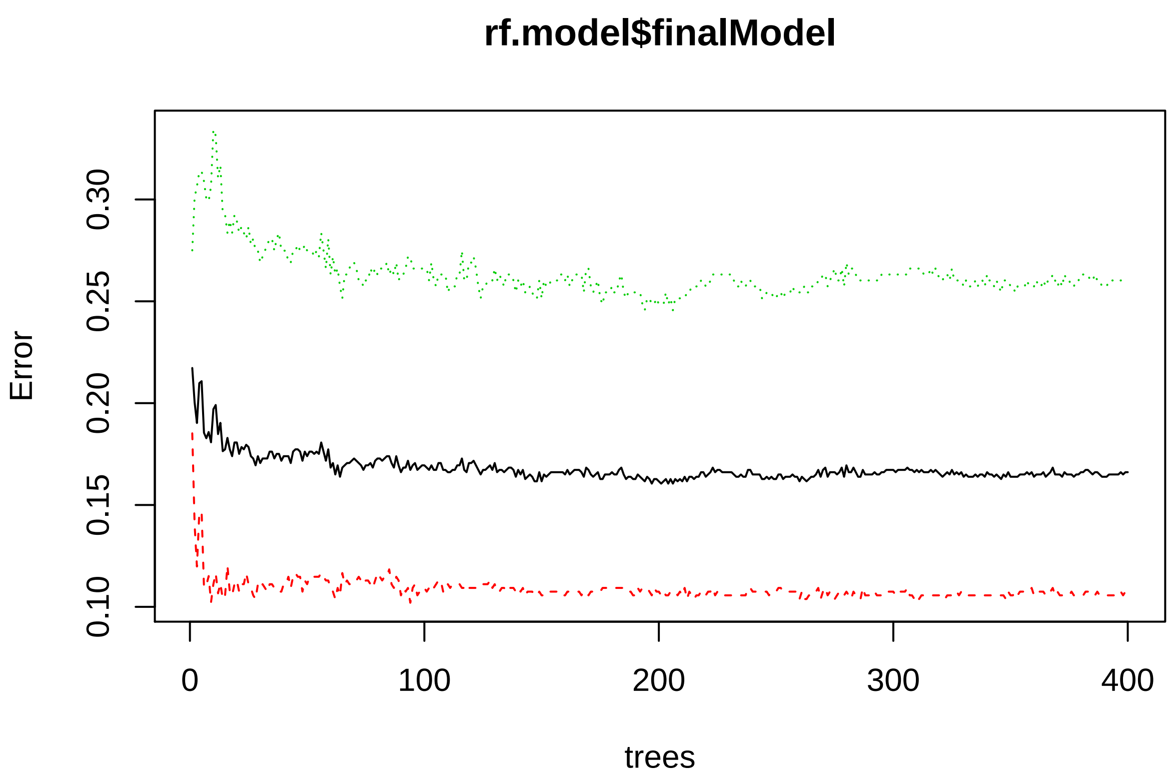
```
## [1] 400
```

```
#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: 801, 801, 803, 802, 802, 802, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.8316800  0.6371979
##    4    0.8249640  0.6243701
##    6    0.8339153  0.6430540
##    8    0.8327670  0.6403319
##   10    0.8215430  0.6190578
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 6.
```

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




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

```
## [1] 0.8339153
```

```r
#about 83%

varImp(rf.model)
```

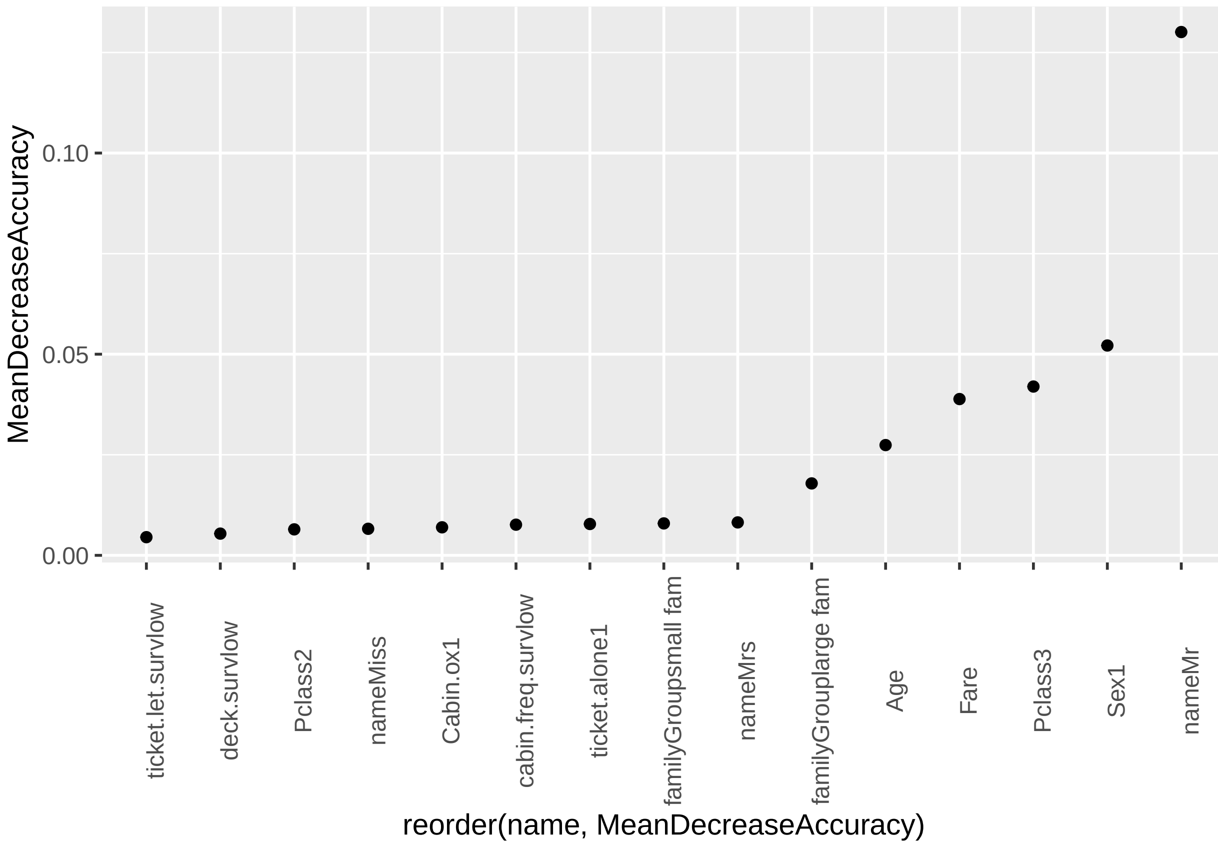
```
## rf variable importance
##
##                         Importance
## nameMr                     100.000
```

```
## Pclass3                   69.758
## Age                       58.984
## Fare                      55.590
## familyGrouplarge fam      44.323
## Sex1                      42.481
## familyGroupsmall fam      17.229
## Pclass2                   16.398
## ticket.alone1             12.730
## ticket.let.survlow        12.391
## cabin.freq.survlow         9.777
## Cabin.ox1                  9.326
## nameMrs                    7.239
## nameMiss                   3.017
## deck.survlow               0.000
```

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

rf.model.result
```

```
##                        MeanDecreaseAccuracy
## Pclass2                         0.006465612
## Pclass3                         0.041962803
## Sex1                            0.052163754
## Age                             0.027413539
## Fare                            0.038850594
## Cabin.ox1                       0.006973273
## deck.survlow                    0.005402401
## cabin.freq.survlow              0.007619640
## ticket.alone1                   0.007793198
## ticket.let.survlow              0.004518084
## nameMiss                        0.006603351
## nameMr                          0.130084443
## nameMrs                         0.008190189
## familyGrouplarge fam            0.017887044
## familyGroupsmall fam            0.007941354
```

```
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))
```




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

```
## [1] 0.8010825
```

```r
#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
##          0 532  43
##          1  17 299
##
##                Accuracy : 0.9327
##                  95% CI : (0.9142, 0.9482)
##     No Information Rate : 0.6162
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8556
##
##  Mcnemar's Test P-Value : 0.001249
```

```
##
##             Sensitivity : 0.9690
##             Specificity : 0.8743
##          Pos Pred Value : 0.9252
##          Neg Pred Value : 0.9462
##              Prevalence : 0.6162
##          Detection Rate : 0.5971
##    Detection Prevalence : 0.6453
##       Balanced Accuracy : 0.9217
##
##        'Positive' Class : 0
##
```

```r
#93.15%

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

```
## [1] 0.09874466
```

```r
#0.0987
```

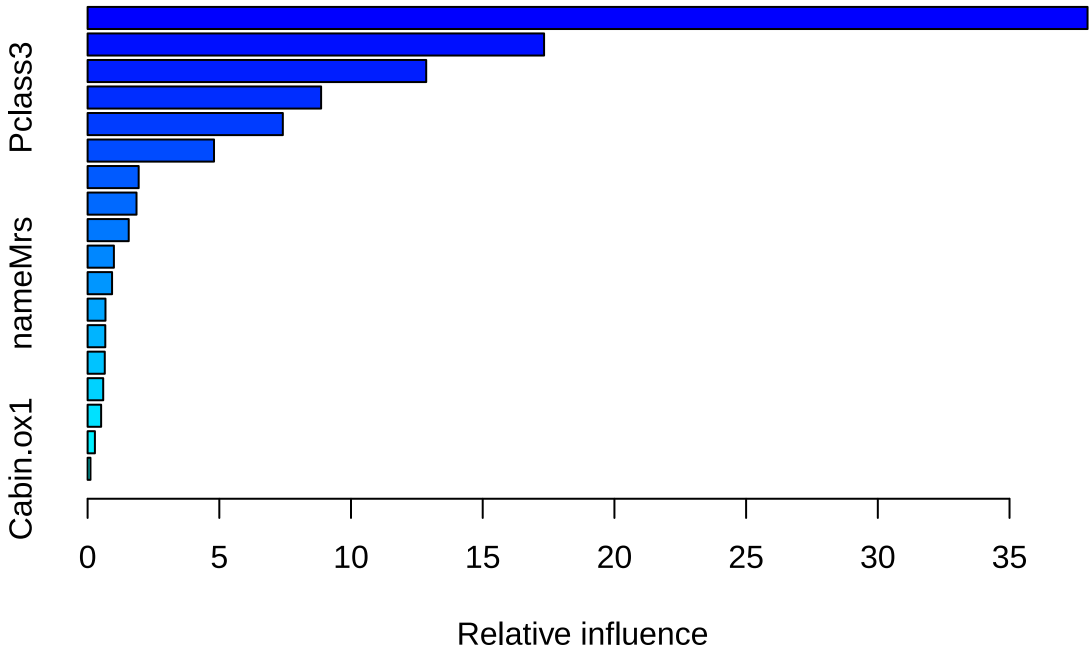
Gradient Boosting Method

```r
#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: 802, 802, 801, 803, 802, 801, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.8306214  0.6340546
##   1                  100      0.8272506  0.6312348
##   1                  150      0.8283617  0.6339705
##   2                   50      0.8339675  0.6445505
##   2                  100      0.8328436  0.6430562
##   2                  150      0.8373130  0.6494927
##   3                   50      0.8373002  0.6521697
##   3                  100      0.8395727  0.6563962
```

```
##   3                  150      0.8429188  0.6623035
##
## 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 37.9625825
## Fare                                        Fare 17.3299360
## Age                                          Age 12.8549703
## Pclass3                                  Pclass3  8.8651563
## familyGrouplarge fam         familyGrouplarge fam  7.4121059
## ticket.let.survlow             ticket.let.survlow  4.7985380
## deck.survlow                         deck.survlow  1.9388728
## EmbarkedS                              EmbarkedS  1.8569618
## cabin.freq.survlow             cabin.freq.survlow  1.5607162
## familyGroupsmall fam         familyGroupsmall fam  0.9982236
## nameMrs                                  nameMrs  0.9287536
## ticket.alone1                      ticket.alone1  0.6781896
## EmbarkedQ                              EmbarkedQ  0.6714772
## Sex1                                        Sex1  0.6518530
## surname.freq.survunknown surname.freq.survunknown  0.5926914
## nameMiss                                nameMiss  0.5131629
## Pclass2                                  Pclass2  0.2773682
## Cabin.ox1                              Cabin.ox1  0.1084408
```

```
#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, 802, 802, 802, 802, 802, ...
## Resampling results across tuning parameters:
##
##   shrinkage  interaction.depth  n.trees  Accuracy   Kappa
##   0.01       1                   100     0.7867915  0.5465762
##   0.01       1                   250     0.8249813  0.6209409
##   0.01       1                   400     0.8294507  0.6311942
##   0.01       1                   550     0.8316979  0.6363860
##   0.01       1                   700     0.8272035  0.6282207
##   0.01       1                   850     0.8249688  0.6244204
##   0.01       1                  1000     0.8305868  0.6374463
##   0.01       1                  1150     0.8328340  0.6435507
##   0.01       1                  1300     0.8350562  0.6481696
##   0.01       1                  1450     0.8373034  0.6533915
##   0.01       1                  1600     0.8339326  0.6464363
##   0.01       1                  1750     0.8316854  0.6420415
##   0.01       1                  1900     0.8328090  0.6441976
##   0.01       1                  2050     0.8328090  0.6442902
##   0.01       1                  2200     0.8339326  0.6463234
##   0.01       1                  2350     0.8328090  0.6443733
##   0.01       1                  2500     0.8294382  0.6372965
##   0.01       1                  2650     0.8328090  0.6441129
##   0.01       1                  2800     0.8350562  0.6486296
##   0.01       1                  2950     0.8384270  0.6552464
##   0.01       1                  3100     0.8395506  0.6574567
##   0.01       1                  3250     0.8406742  0.6596391
```

```
##   0.01       1                       3400      0.8406742  0.6596391
##   0.01       1                       3550      0.8384270  0.6550346
##   0.01       1                       3700      0.8384270  0.6546690
##   0.01       1                       3850      0.8373034  0.6528242
##   0.01       1                       4000      0.8361798  0.6503179
##   0.01       1                       4150      0.8361798  0.6499523
##   0.01       1                       4300      0.8350562  0.6479375
##   0.01       1                       4450      0.8339326  0.6454676
##   0.01       1                       4600      0.8350562  0.6475915
##   0.01       1                       4750      0.8328090  0.6430971
##   0.01       1                       4900      0.8350562  0.6479657
##   0.01       1                       5050      0.8350562  0.6479657
##   0.01       1                       5200      0.8384270  0.6554488
##   0.01       1                       5350      0.8361798  0.6505297
##   0.01       1                       5500      0.8373034  0.6527278
##   0.01       1                       5650      0.8361798  0.6501429
##   0.01       1                       5800      0.8395506  0.6574462
##   0.01       1                       5950      0.8350562  0.6479020
##   0.01       2                        100      0.8283521  0.6285488
##   0.01       2                        250      0.8339451  0.6415837
##   0.01       2                        400      0.8272035  0.6280976
##   0.01       2                        550      0.8294507  0.6347208
##   0.01       2                        700      0.8339326  0.6452259
##   0.01       2                        850      0.8350562  0.6472024
##   0.01       2                       1000      0.8328090  0.6425691
##   0.01       2                       1150      0.8350562  0.6479498
##   0.01       2                       1300      0.8361798  0.6494045
##   0.01       2                       1450      0.8328215  0.6419962
##   0.01       2                       1600      0.8395630  0.6566980
##   0.01       2                       1750      0.8395381  0.6557158
##   0.01       2                       1900      0.8372909  0.6506955
##   0.01       2                       2050      0.8384145  0.6535969
##   0.01       2                       2200      0.8350437  0.6459760
##   0.01       2                       2350      0.8350437  0.6454172
##   0.01       2                       2500      0.8372909  0.6507721
##   0.01       2                       2650      0.8350437  0.6455721
##   0.01       2                       2800      0.8339201  0.6433652
##   0.01       2                       2950      0.8372909  0.6503981
##   0.01       2                       3100      0.8361673  0.6480919
##   0.01       2                       3250      0.8384020  0.6530042
##   0.01       2                       3400      0.8372784  0.6507973
##   0.01       2                       3550      0.8372784  0.6507973
##   0.01       2                       3700      0.8361548  0.6486399
##   0.01       2                       3850      0.8384020  0.6526498
##   0.01       2                       4000      0.8384020  0.6526896
##   0.01       2                       4150      0.8395256  0.6549559
##   0.01       2                       4300      0.8384020  0.6527490
##   0.01       2                       4450      0.8350437  0.6457291
##   0.01       2                       4600      0.8361673  0.6483361
##   0.01       2                       4750      0.8350437  0.6461181
##   0.01       2                       4900      0.8350312  0.6457369
##   0.01       2                       5050      0.8361548  0.6482912
##   0.01       2                       5200      0.8361548  0.6482912
##   0.01       2                       5350      0.8372784  0.6505092
```

```
##   0.01       2                      5500      0.8372784  0.6509619
##   0.01       2                      5650      0.8361548  0.6484076
##   0.01       2                      5800      0.8339076  0.6441437
##   0.01       2                      5950      0.8361548  0.6492887
##   0.01       3                       100      0.8316979  0.6355581
##   0.01       3                       250      0.8316979  0.6366752
##   0.01       3                       400      0.8316854  0.6394731
##   0.01       3                       550      0.8328090  0.6421177
##   0.01       3                       700      0.8327965  0.6420810
##   0.01       3                       850      0.8361673  0.6489817
##   0.01       3                      1000      0.8395381  0.6562217
##   0.01       3                      1150      0.8395256  0.6553732
##   0.01       3                      1300      0.8406492  0.6583537
##   0.01       3                      1450      0.8428964  0.6636912
##   0.01       3                      1600      0.8395256  0.6562806
##   0.01       3                      1750      0.8406492  0.6588255
##   0.01       3                      1900      0.8417728  0.6612311
##   0.01       3                      2050      0.8395256  0.6563723
##   0.01       3                      2200      0.8384020  0.6528228
##   0.01       3                      2350      0.8395256  0.6555013
##   0.01       3                      2500      0.8395256  0.6550030
##   0.01       3                      2650      0.8384020  0.6523821
##   0.01       3                      2800      0.8372909  0.6500850
##   0.01       3                      2950      0.8384020  0.6531616
##   0.01       3                      3100      0.8395256  0.6546807
##   0.01       3                      3250      0.8372784  0.6499034
##   0.01       3                      3400      0.8372659  0.6502534
##   0.01       3                      3550      0.8361423  0.6479803
##   0.01       3                      3700      0.8395131  0.6554598
##   0.01       3                      3850      0.8372659  0.6510252
##   0.01       3                      4000      0.8361423  0.6483917
##   0.01       3                      4150      0.8383895  0.6539460
##   0.01       3                      4300      0.8350187  0.6469341
##   0.01       3                      4450      0.8350187  0.6466067
##   0.01       3                      4600      0.8372659  0.6513352
##   0.01       3                      4750      0.8350312  0.6469585
##   0.01       3                      4900      0.8350312  0.6469585
##   0.01       3                      5050      0.8339076  0.6447663
##   0.01       3                      5200      0.8305368  0.6373304
##   0.01       3                      5350      0.8316604  0.6399403
##   0.01       3                      5500      0.8327840  0.6416950
##   0.01       3                      5650      0.8316604  0.6393054
##   0.01       3                      5800      0.8339076  0.6438017
##   0.01       3                      5950      0.8305493  0.6368333
##   0.01       4                       100      0.8328215  0.6346776
##   0.01       4                       250      0.8294507  0.6317297
##   0.01       4                       400      0.8316604  0.6381775
##   0.01       4                       550      0.8350312  0.6457002
##   0.01       4                       700      0.8395256  0.6556969
##   0.01       4                       850      0.8372784  0.6509671
##   0.01       4                      1000      0.8406492  0.6588304
##   0.01       4                      1150      0.8395256  0.6566544
##   0.01       4                      1300      0.8361548  0.6491894
##   0.01       4                      1450      0.8395256  0.6558215
```

```
##   0.01     4                    1600     0.8384020  0.6538109
##   0.01     4                    1750     0.8372784  0.6516547
##   0.01     4                    1900     0.8372784  0.6512593
##   0.01     4                    2050     0.8361548  0.6488272
##   0.01     4                    2200     0.8384020  0.6532375
##   0.01     4                    2350     0.8383895  0.6540523
##   0.01     4                    2500     0.8372659  0.6514743
##   0.01     4                    2650     0.8350187  0.6464881
##   0.01     4                    2800     0.8372659  0.6512614
##   0.01     4                    2950     0.8372659  0.6509894
##   0.01     4                    3100     0.8383895  0.6531212
##   0.01     4                    3250     0.8372659  0.6509490
##   0.01     4                    3400     0.8383895  0.6534445
##   0.01     4                    3550     0.8383895  0.6535149
##   0.01     4                    3700     0.8406367  0.6579309
##   0.01     4                    3850     0.8383895  0.6535558
##   0.01     4                    4000     0.8372659  0.6507364
##   0.01     4                    4150     0.8372784  0.6507609
##   0.01     4                    4300     0.8384020  0.6530181
##   0.01     4                    4450     0.8372784  0.6503108
##   0.01     4                    4600     0.8361548  0.6486627
##   0.01     4                    4750     0.8339076  0.6436021
##   0.01     4                    4900     0.8361548  0.6480772
##   0.01     4                    5050     0.8361548  0.6482483
##   0.01     4                    5200     0.8327840  0.6410908
##   0.01     4                    5350     0.8339076  0.6430065
##   0.01     4                    5500     0.8305368  0.6353684
##   0.01     4                    5650     0.8339201  0.6428712
##   0.01     4                    5800     0.8316729  0.6380066
##   0.01     4                    5950     0.8283021  0.6304540
##   0.10     1                     100     0.8317104  0.6409639
##   0.10     1                     250     0.8361798  0.6506527
##   0.10     1                     400     0.8350562  0.6470400
##   0.10     1                     550     0.8361798  0.6491662
##   0.10     1                     700     0.8305618  0.6373187
##   0.10     1                     850     0.8294382  0.6342124
##   0.10     1                    1000     0.8305868  0.6373330
##   0.10     1                    1150     0.8339451  0.6448748
##   0.10     1                    1300     0.8283396  0.6326740
##   0.10     1                    1450     0.8272160  0.6303506
##   0.10     1                    1600     0.8305743  0.6364192
##   0.10     1                    1750     0.8294507  0.6342148
##   0.10     1                    1900     0.8317104  0.6390638
##   0.10     1                    2050     0.8249688  0.6252353
##   0.10     1                    2200     0.8238452  0.6226094
##   0.10     1                    2350     0.8272160  0.6297087
##   0.10     1                    2500     0.8294632  0.6343151
##   0.10     1                    2650     0.8238577  0.6226153
##   0.10     1                    2800     0.8261049  0.6280255
##   0.10     1                    2950     0.8261049  0.6270694
##   0.10     1                    3100     0.8238577  0.6227009
##   0.10     1                    3250     0.8294757  0.6355473
##   0.10     1                    3400     0.8238452  0.6225346
##   0.10     1                    3550     0.8249688  0.6239811
```

```
##   0.10       1                   3700      0.8204744  0.6151621
##   0.10       1                   3850      0.8182272  0.6110121
##   0.10       1                   4000      0.8204744  0.6153597
##   0.10       1                   4150      0.8227216  0.6210054
##   0.10       1                   4300      0.8238452  0.6221994
##   0.10       1                   4450      0.8260924  0.6270179
##   0.10       1                   4600      0.8260799  0.6274133
##   0.10       1                   4750      0.8283271  0.6323782
##   0.10       1                   4900      0.8249563  0.6240117
##   0.10       1                   5050      0.8249563  0.6241694
##   0.10       1                   5200      0.8260799  0.6270423
##   0.10       1                   5350      0.8272160  0.6313185
##   0.10       1                   5500      0.8249563  0.6257478
##   0.10       1                   5650      0.8238452  0.6240407
##   0.10       1                   5800      0.8272035  0.6295978
##   0.10       1                   5950      0.8249813  0.6260970
##   0.10       2                    100      0.8361673  0.6500155
##   0.10       2                    250      0.8372909  0.6498635
##   0.10       2                    400      0.8361548  0.6471144
##   0.10       2                    550      0.8372909  0.6518970
##   0.10       2                    700      0.8327840  0.6391798
##   0.10       2                    850      0.8339076  0.6429576
##   0.10       2                   1000      0.8260549  0.6276600
##   0.10       2                   1150      0.8271910  0.6302685
##   0.10       2                   1300      0.8294257  0.6345913
##   0.10       2                   1450      0.8294382  0.6339560
##   0.10       2                   1600      0.8294507  0.6342830
##   0.10       2                   1750      0.8316979  0.6405666
##   0.10       2                   1900      0.8271910  0.6303746
##   0.10       2                   2050      0.8249563  0.6252153
##   0.10       2                   2200      0.8238327  0.6230104
##   0.10       2                   2350      0.8204744  0.6167233
##   0.10       2                   2500      0.8294382  0.6357700
##   0.10       2                   2650      0.8249313  0.6255999
##   0.10       2                   2800      0.8316729  0.6401176
##   0.10       2                   2950      0.8282896  0.6347824
##   0.10       2                   3100      0.8271910  0.6311355
##   0.10       2                   3250      0.8249064  0.6270745
##   0.10       2                   3400      0.8260300  0.6291095
##   0.10       2                   3550      0.8249313  0.6267849
##   0.10       2                   3700      0.8282772  0.6337800
##   0.10       2                   3850      0.8249189  0.6273285
##   0.10       2                   4000      0.8226841  0.6225987
##   0.10       2                   4150      0.8226841  0.6229489
##   0.10       2                   4300      0.8226592  0.6222066
##   0.10       2                   4450      0.8215730  0.6199697
##   0.10       2                   4600      0.8181898  0.6128773
##   0.10       2                   4750      0.8170662  0.6104925
##   0.10       2                   4900      0.8215605  0.6204944
##   0.10       2                   5050      0.8204370  0.6171008
##   0.10       2                   5200      0.8204494  0.6172865
##   0.10       2                   5350      0.8238202  0.6255345
##   0.10       2                   5500      0.8170662  0.6113930
##   0.10       2                   5650      0.8181898  0.6133641
```

```
##   0.10       2                  5800      0.8193009  0.6154645
##   0.10       2                  5950      0.8182022  0.6128703
##   0.10       3                   100      0.8305618  0.6363623
##   0.10       3                   250      0.8361423  0.6472373
##   0.10       3                   400      0.8350062  0.6456816
##   0.10       3                   550      0.8260300  0.6257945
##   0.10       3                   700      0.8260175  0.6257713
##   0.10       3                   850      0.8294007  0.6328573
##   0.10       3                  1000      0.8283021  0.6315169
##   0.10       3                  1150      0.8260300  0.6278975
##   0.10       3                  1300      0.8327591  0.6420083
##   0.10       3                  1450      0.8249189  0.6248758
##   0.10       3                  1600      0.8226592  0.6202514
##   0.10       3                  1750      0.8193258  0.6131084
##   0.10       3                  1900      0.8215481  0.6179386
##   0.10       3                  2050      0.8215481  0.6193613
##   0.10       3                  2200      0.8193009  0.6136205
##   0.10       3                  2350      0.8192884  0.6140253
##   0.10       3                  2500      0.8159675  0.6067770
##   0.10       3                  2650      0.8193009  0.6138976
##   0.10       3                  2800      0.8159301  0.6070844
##   0.10       3                  2950      0.8103121  0.5958668
##   0.10       3                  3100      0.8159301  0.6079462
##   0.10       3                  3250      0.8181773  0.6125156
##   0.10       3                  3400      0.8148065  0.6055967
##   0.10       3                  3550      0.8148065  0.6056973
##   0.10       3                  3700      0.8114482  0.5984542
##   0.10       3                  3850      0.8125593  0.6004527
##   0.10       3                  4000      0.8136829  0.6047528
##   0.10       3                  4150      0.8092010  0.5937283
##   0.10       3                  4300      0.8103121  0.5963320
##   0.10       3                  4450      0.8069663  0.5895958
##   0.10       3                  4600      0.8069538  0.5900700
##   0.10       3                  4750      0.8069538  0.5898983
##   0.10       3                  4900      0.8080774  0.5925458
##   0.10       3                  5050      0.8024594  0.5807261
##   0.10       3                  5200      0.8069413  0.5901315
##   0.10       3                  5350      0.8046941  0.5858492
##   0.10       3                  5500      0.8058177  0.5879645
##   0.10       3                  5650      0.8013358  0.5784079
##   0.10       3                  5800      0.8024594  0.5809573
##   0.10       3                  5950      0.8013358  0.5788231
##   0.10       4                   100      0.8294132  0.6343434
##   0.10       4                   250      0.8383895  0.6532633
##   0.10       4                   400      0.8271660  0.6305048
##   0.10       4                   550      0.8227091  0.6214020
##   0.10       4                   700      0.8204494  0.6184066
##   0.10       4                   850      0.8204494  0.6180296
##   0.10       4                  1000      0.8204619  0.6174917
##   0.10       4                  1150      0.8204619  0.6169888
##   0.10       4                  1300      0.8159551  0.6082242
##   0.10       4                  1450      0.8182022  0.6129729
##   0.10       4                  1600      0.8148065  0.6062573
##   0.10       4                  1750      0.8103371  0.5972998
```

```
##    0.10       4                   1900      0.8159301  0.6078624
##    0.10       4                   2050      0.8080649  0.5920778
##    0.10       4                   2200      0.8102996  0.5983922
##    0.10       4                   2350      0.8102996  0.5980267
##    0.10       4                   2500      0.8024719  0.5812727
##    0.10       4                   2650      0.8024594  0.5801124
##    0.10       4                   2800      0.8080774  0.5921546
##    0.10       4                   2950      0.8058427  0.5888350
##    0.10       4                   3100      0.8069413  0.5905940
##    0.10       4                   3250      0.8069538  0.5908965
##    0.10       4                   3400      0.8047066  0.5861657
##    0.10       4                   3550      0.8047191  0.5860979
##    0.10       4                   3700      0.8069288  0.5905889
##    0.10       4                   3850      0.8069538  0.5917695
##    0.10       4                   4000      0.8058427  0.5888908
##    0.10       4                   4150      0.8103121  0.5979125
##    0.10       4                   4300      0.8069413  0.5903116
##    0.10       4                   4450      0.8058302  0.5886268
##    0.10       4                   4600      0.8058302  0.5888931
##    0.10       4                   4750      0.8046941  0.5861149
##    0.10       4                   4900      0.8058177  0.5888690
##    0.10       4                   5050      0.8002122  0.5771487
##    0.10       4                   5200      0.8046941  0.5870378
##    0.10       4                   5350      0.7990886  0.5742914
##    0.10       4                   5500      0.8046941  0.5864209
##    0.10       4                   5650      0.8035830  0.5838610
##    0.10       4                   5800      0.8080774  0.5938094
##    0.10       4                   5950      0.8058177  0.5895161
##
## 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 = 1450,
##  interaction.depth = 3, shrinkage = 0.01 and n.minobsinnode = 10.
```

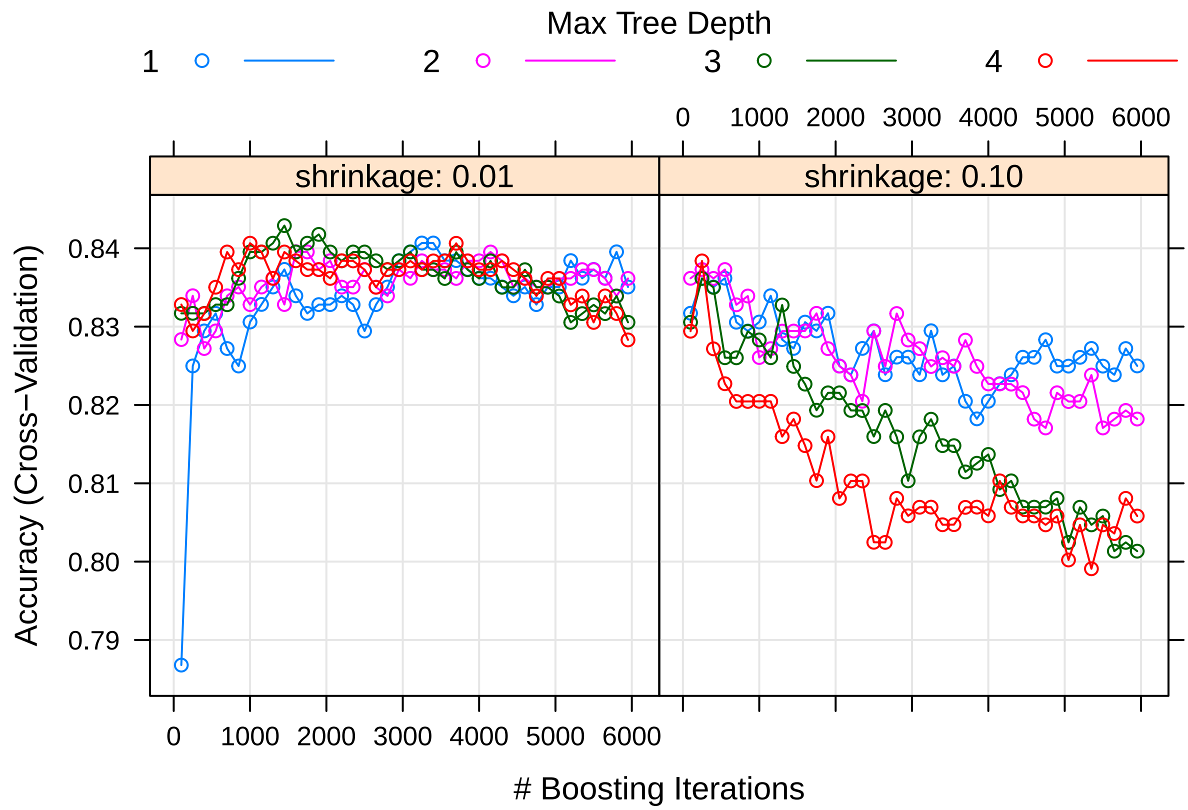
```r
max(boost.model$results$Accuracy)
```

```
## [1] 0.8428964
```

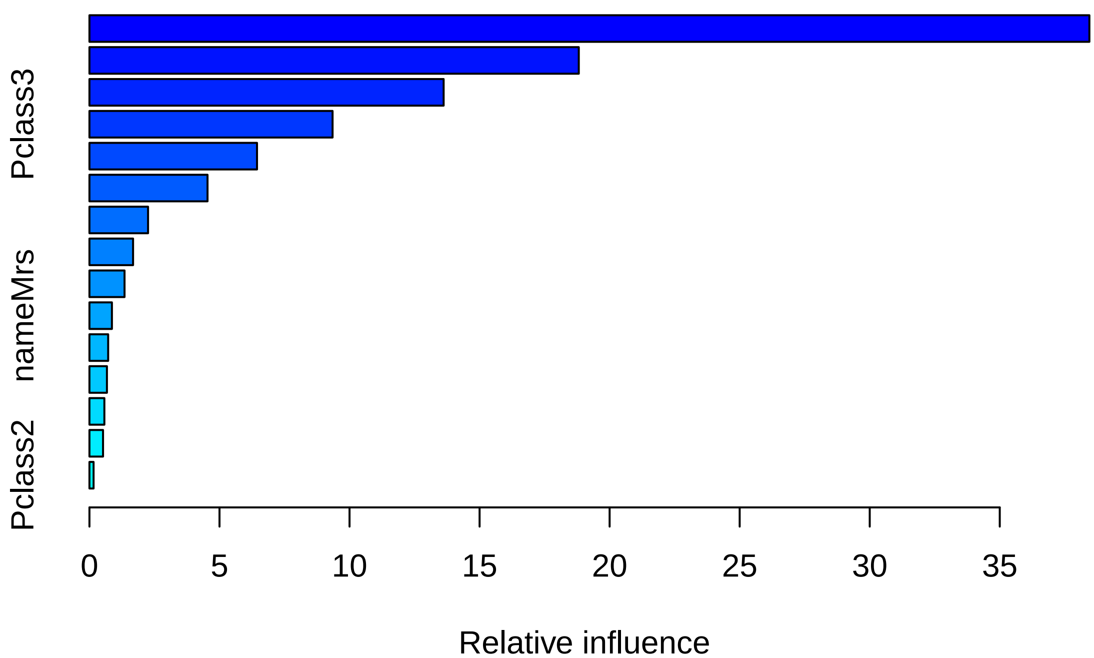
```r
#84.44%

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

```
## [1] 0.8182462
```

```r
#81.28%

plot(boost.model)
```




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




```
##                                       var    rel.inf
## nameMr                             nameMr 38.4456252
## Fare                                 Fare 18.8139829
## Age                                   Age 13.6180307
## Pclass3                           Pclass3  9.3479133
```

```
## familyGrouplarge fam familyGrouplarge fam  6.4429449
## ticket.let.survlow     ticket.let.survlow  4.5398689
## Sex1                                 Sex1  2.2532818
## cabin.freq.survlow     cabin.freq.survlow  1.6773612
## deck.survlow                 deck.survlow  1.3493302
## nameMrs                           nameMrs  0.8642179
## familyGroupsmall fam familyGroupsmall fam  0.7195182
## Cabin.ox1                       Cabin.ox1  0.6722524
## nameMiss                         nameMiss  0.5739130
## ticket.alone1               ticket.alone1  0.5225127
## Pclass2                           Pclass2  0.1592466
```

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

```
## [1] 1450
```

```r
#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
##          0 505  66
##          1  44 276
##
##                Accuracy : 0.8765
##                  95% CI : (0.8531, 0.8974)
##     No Information Rate : 0.6162
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.7358
##
##  Mcnemar's Test P-Value : 0.04526
##
##             Sensitivity : 0.9199
##             Specificity : 0.8070
##          Pos Pred Value : 0.8844
##          Neg Pred Value : 0.8625
##              Prevalence : 0.6162
##          Detection Rate : 0.5668
##    Detection Prevalence : 0.6409
##       Balanced Accuracy : 0.8634
##
##        'Positive' Class : 0
##
```

```r
#88.78%

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

```
## [1] 0.03364683
```

```r
#0.0437
```

SVM - kernel radial

```r
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, 803, ...
## Resampling results across tuning parameters:
##
##   C       Accuracy   Kappa
##     0.25  0.8317441  0.6394228
##     0.50  0.8283609  0.6285819
##     1.00  0.8305706  0.6323720
##     2.00  0.8316942  0.6357776
##     4.00  0.8249520  0.6229609
##     8.00  0.8170738  0.6078849
##    16.00  0.8069862  0.5857883
##    32.00  0.7991085  0.5684094
##    64.00  0.7957499  0.5616529
##   128.00  0.7991590  0.5702028
##
## Tuning parameter 'sigma' was held constant at a value of 0.05485621
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.05485621 and C = 0.25.
```

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

```
## [1] 0.8317441
```

```r
#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
```

```r
#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, 802, 802, 802, 802, ...
## Resampling results across tuning parameters:
##
##   sigma  C     Accuracy   Kappa
##   0.01   0.01  0.6161673  0.0000000
##   0.01   0.26  0.8069913  0.5800724
##   0.01   0.51  0.8103496  0.5866245
##   0.01   0.76  0.8103496  0.5866245
##   0.01   1.01  0.8103496  0.5866245
##   0.01   1.26  0.8103496  0.5866245
##   0.01   1.51  0.8103496  0.5866245
##   0.01   1.76  0.8103496  0.5866245
##   0.01   2.01  0.8103496  0.5866245
##   0.02   0.01  0.6161673  0.0000000
##   0.02   0.26  0.8081024  0.5822589
##   0.02   0.51  0.8103496  0.5866245
##   0.02   0.76  0.8103496  0.5866245
##   0.02   1.01  0.8103496  0.5866245
##   0.02   1.26  0.8103496  0.5866245
##   0.02   1.51  0.8103496  0.5866245
##   0.02   1.76  0.8103496  0.5866245
##   0.02   2.01  0.8103496  0.5866245
##   0.03   0.01  0.6161673  0.0000000
##   0.03   0.26  0.8081024  0.5822589
##   0.03   0.51  0.8092260  0.5844011
##   0.03   0.76  0.8092260  0.5844011
##   0.03   1.01  0.8103496  0.5872464
##   0.03   1.26  0.8114732  0.5894698
##   0.03   1.51  0.8114732  0.5894698
##   0.03   1.76  0.8114856  0.5899984
```

```
##    0.03   2.01  0.8114856  0.5899984
##    0.04   0.01  0.6161673  0.0000000
##    0.04   0.26  0.8024969  0.5712651
##    0.04   0.51  0.8081024  0.5829707
##    0.04   0.76  0.8103496  0.5872464
##    0.04   1.01  0.8103620  0.5877750
##    0.04   1.26  0.8114856  0.5899984
##    0.04   1.51  0.8126092  0.5921765
##    0.04   1.76  0.8103620  0.5869309
##    0.04   2.01  0.8047566  0.5745653
##    0.05   0.01  0.6161673  0.0000000
##    0.05   0.26  0.8013983  0.5695598
##    0.05   0.51  0.8047441  0.5762376
##    0.05   0.76  0.8092385  0.5855797
##    0.05   1.01  0.8092385  0.5850573
##    0.05   1.26  0.8058677  0.5767282
##    0.05   1.51  0.8058801  0.5768927
##    0.05   1.76  0.8069913  0.5775183
##    0.05   2.01  0.8069913  0.5777189
##    0.06   0.01  0.6161673  0.0000000
##    0.06   0.26  0.7991511  0.5674687
##    0.06   0.51  0.8036454  0.5746500
##    0.06   0.76  0.8047566  0.5758718
##    0.06   1.01  0.8036330  0.5723419
##    0.06   1.26  0.8081149  0.5797675
##    0.06   1.51  0.8081149  0.5799705
##    0.06   1.76  0.8047191  0.5727643
##    0.06   2.01  0.8047316  0.5716980
##    0.07   0.01  0.6161673  0.0000000
##    0.07   0.26  0.8002747  0.5710041
##    0.07   0.51  0.8025218  0.5720253
##    0.07   0.76  0.8002747  0.5657454
##    0.07   1.01  0.8081149  0.5797993
##    0.07   1.26  0.8058552  0.5758367
##    0.07   1.51  0.8069663  0.5763337
##    0.07   1.76  0.8058677  0.5745482
##    0.07   2.01  0.8047566  0.5718301
##    0.08   0.01  0.6161673  0.0000000
##    0.08   0.26  0.8013858  0.5753066
##    0.08   0.51  0.7968914  0.5605278
##    0.08   0.76  0.8036205  0.5714836
##    0.08   1.01  0.8024969  0.5688007
##    0.08   1.26  0.8058552  0.5746537
##    0.08   1.51  0.8047441  0.5724907
##    0.08   1.76  0.8047441  0.5719561
##    0.08   2.01  0.8024969  0.5676466
##    0.09   0.01  0.6161673  0.0000000
##    0.09   0.26  0.8013858  0.5754146
##    0.09   0.51  0.7980150  0.5625795
##    0.09   0.76  0.8013733  0.5677303
##    0.09   1.01  0.8024844  0.5673259
##    0.09   1.26  0.8047441  0.5724907
##    0.09   1.51  0.8024969  0.5676466
##    0.09   1.76  0.8002497  0.5636046
```

```
##   0.09   2.01  0.8036080  0.5739394
##   0.10   0.01  0.6161673  0.0000000
##   0.10   0.26  0.8002622  0.5725210
##   0.10   0.51  0.8024969  0.5718688
##   0.10   0.76  0.8013608  0.5667512
##   0.10   1.01  0.8036080  0.5706548
##   0.10   1.26  0.8024969  0.5683120
##   0.10   1.51  0.8013733  0.5671499
##   0.10   1.76  0.8013858  0.5699287
##   0.10   2.01  0.7980150  0.5626456
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.04 and C = 1.51.
```

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

```
## [1] 0.8126092
```

```r
#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
##          0 491 107
##          1  58 235
##
##                Accuracy : 0.8148
##                  95% CI : (0.7877, 0.8398)
##     No Information Rate : 0.6162
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5976
##
##  Mcnemar's Test P-Value : 0.0001864
##
##             Sensitivity : 0.8944
##             Specificity : 0.6871
##          Pos Pred Value : 0.8211
##          Neg Pred Value : 0.8020
##              Prevalence : 0.6162
##          Detection Rate : 0.5511
##    Detection Prevalence : 0.6712
##       Balanced Accuracy : 0.7907
##
##        'Positive' Class : 0
##
```

```r
#0.8395

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

```
## [1] 0.002205576
```

```r
#0.0235
```

Ensembling models in a dataset

```r
#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         0:266   0:257   0:271   0:258
##  Class :character   1:152   1:161   1:147   1:160
##  Mode  :character
```

Creating submission

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

```
##   [1] 0 0 0 0 1 0 0 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 0 0 1 0 1 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 1
##  [71] 1 0 1 1 1 0 0 1 0 1 1 0 0 0 0 0 1 1 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 1 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 1 1 0 1 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 0 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 0 0 1 0 0 1
## Levels: 0 1
```

```r
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%