Titantic Competiton

Observer

October 17, 2016

Included Package

```
library(caret)
library(doMC)
library(randomForest)
library(data.table)
library(DMwR)
```

Set Core

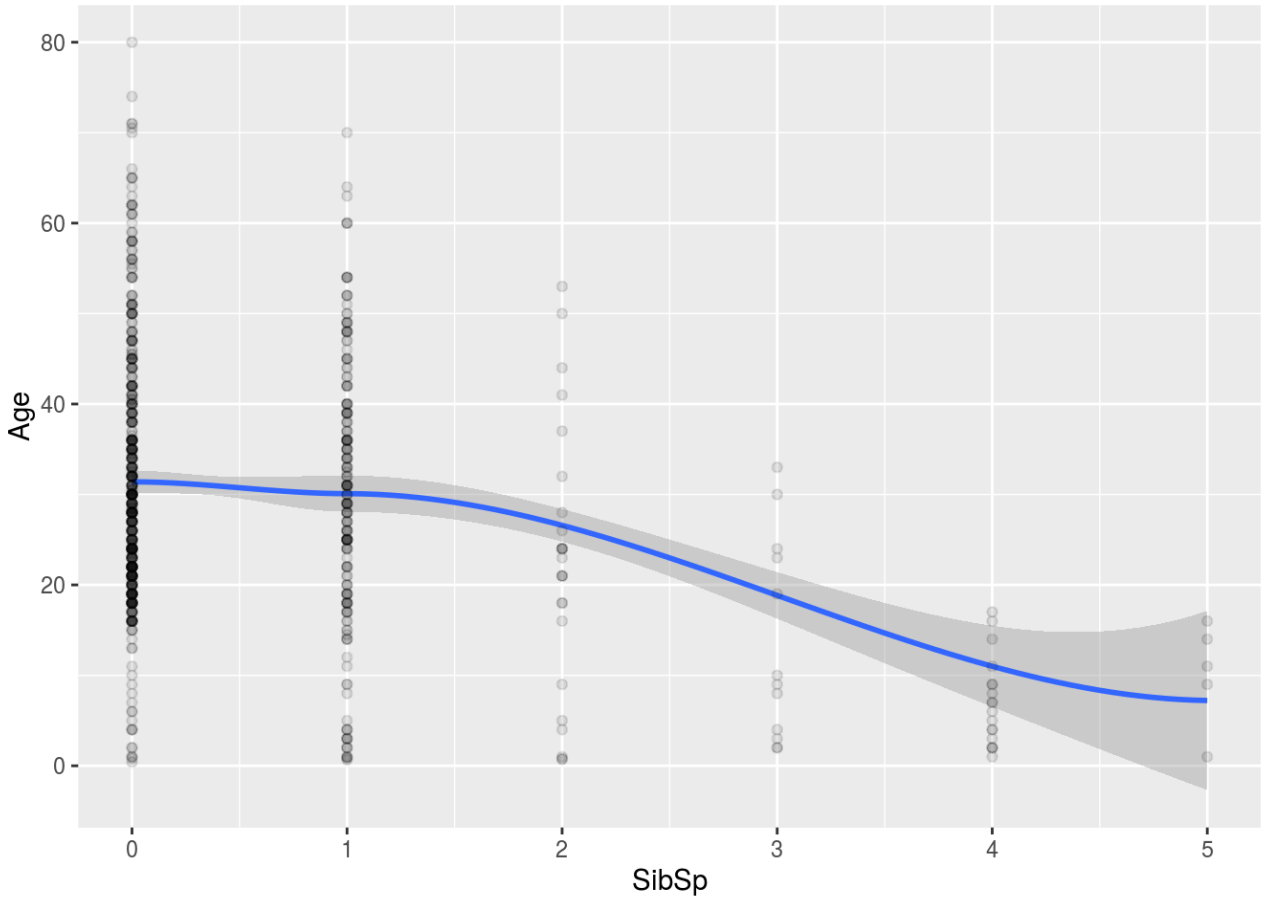
```
# using multiple processing... comment this if it does not work
registerDoMC(cores = 7)
```

Load Data

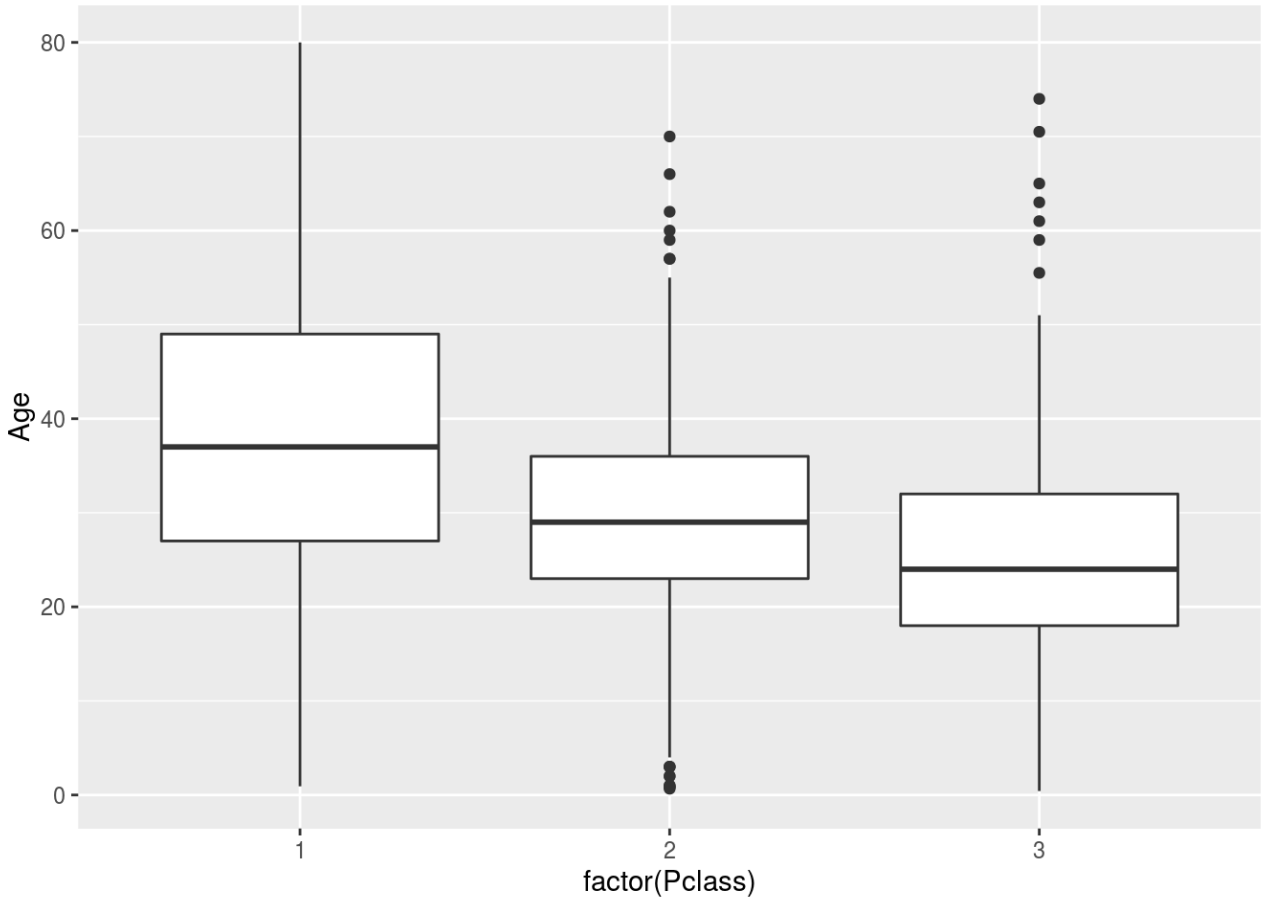
```
dtest<-read.csv("./data/test.csv", header = T)
ds<-read.csv("./data/train.csv", header = T)
dtst = dtest
dtst$Survived=0
dtst$src = 1
ds$src = 0
dall<-rbind(ds,dtst)
dall$Survived = as.factor(dall$Survived)
```

EDA

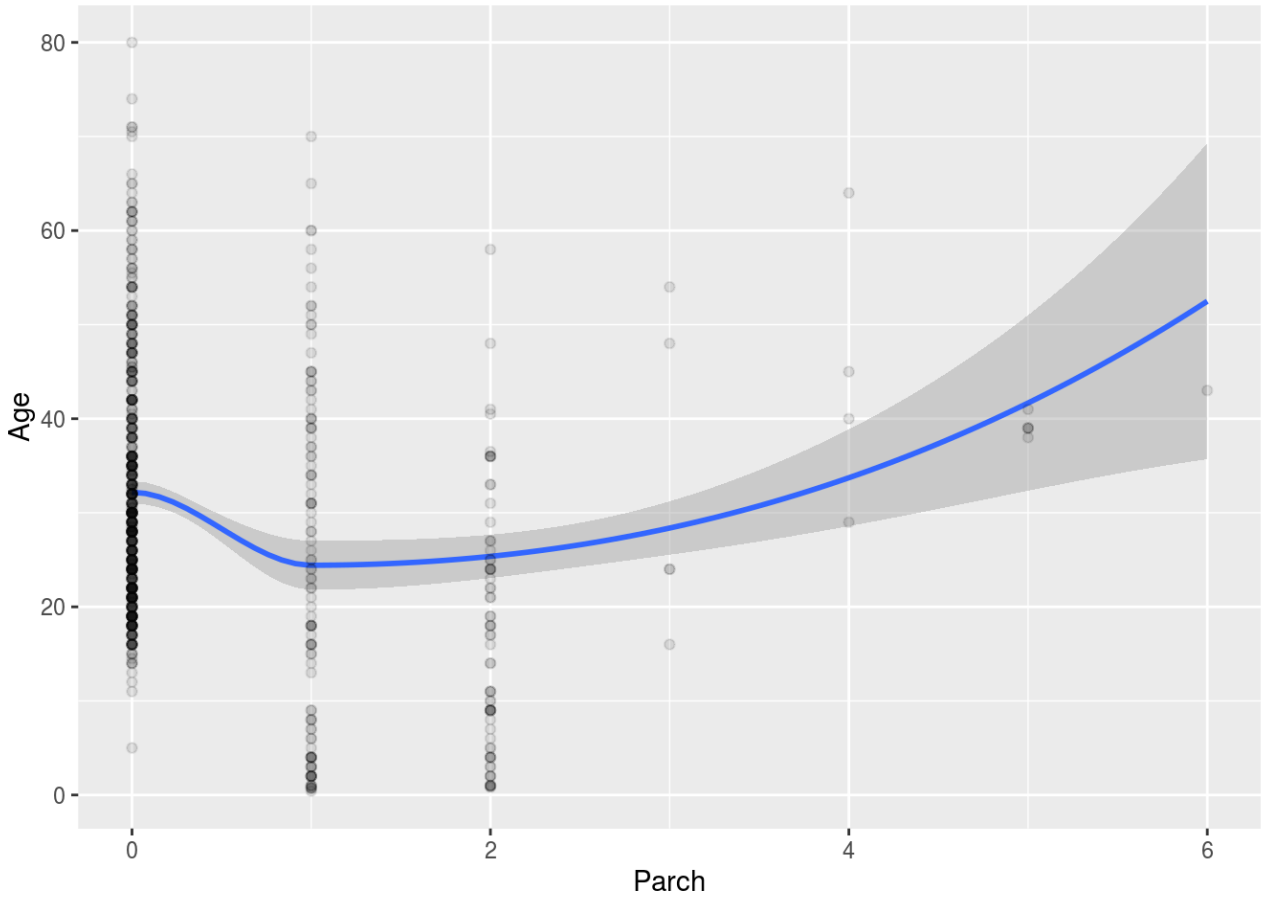
```
# Normalized Fare by Pclass
ds$nFare[ds$Pclass==1] = log(ds$Fare[ds$Pclass==1] +0.01) - log(mean(ds$Fare[
ds$Pclass==1])+0.01)
ds$nFare[ds$Pclass==2] = log(ds$Fare[ds$Pclass==2] +0.01) - log(mean(ds$Fare[
ds$Pclass==2])+0.01)
ds$nFare[ds$Pclass==3] = log(ds$Fare[ds$Pclass==3] +0.01) - log(mean(ds$Fare[
ds$Pclass==3])+0.01)
ds$nFare[is.na(ds$nFare)] = 0

ds_m <- ds[!is.na(ds$Age),]
ds_m$Survived <- as.factor(ds_m$Survived)

ggplot(ds_m, aes(SibSp,Age)) + stat_smooth() + geom_point(alpha=0.1)
```



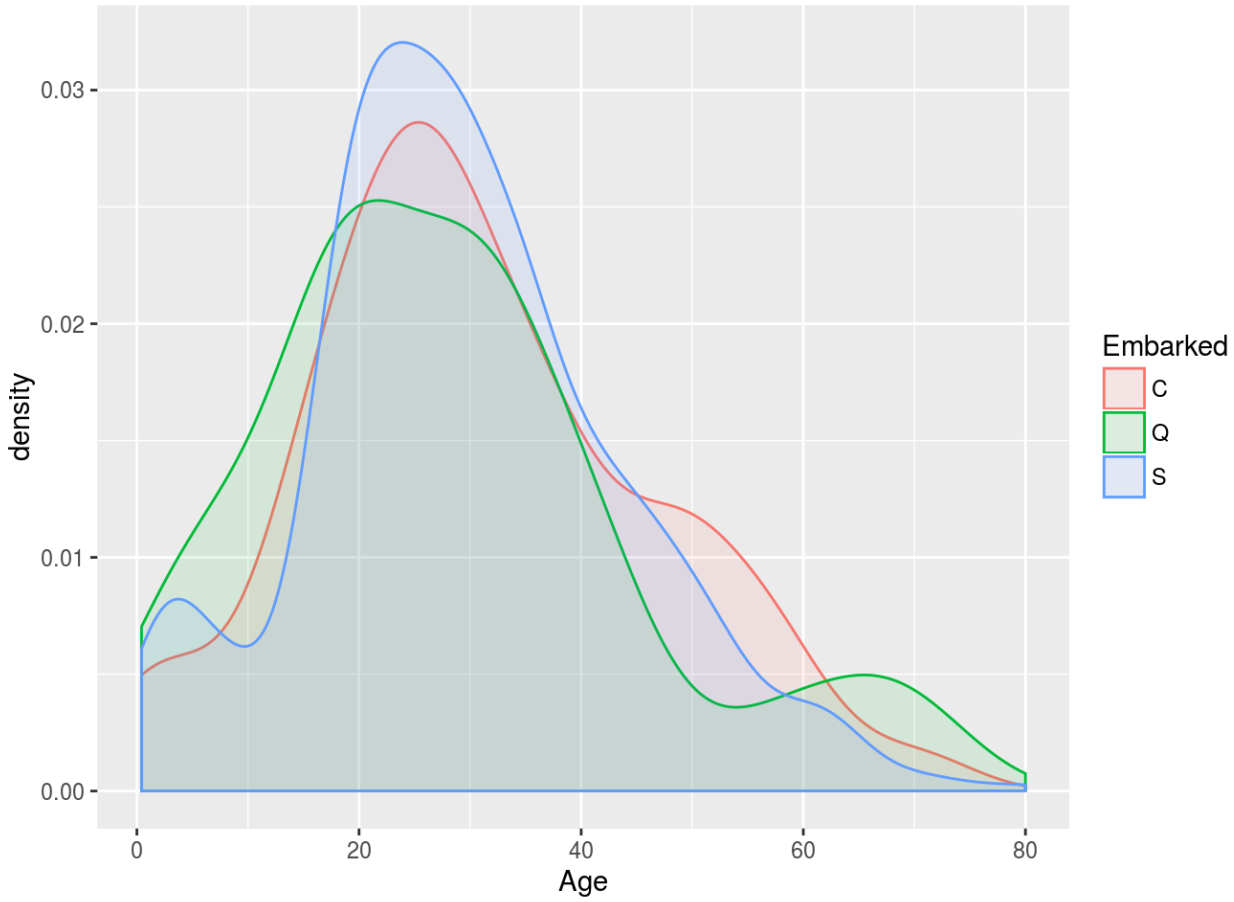

```
ggplot(ds_m, aes(factor(Pclass),Age)) + geom_boxplot()
```



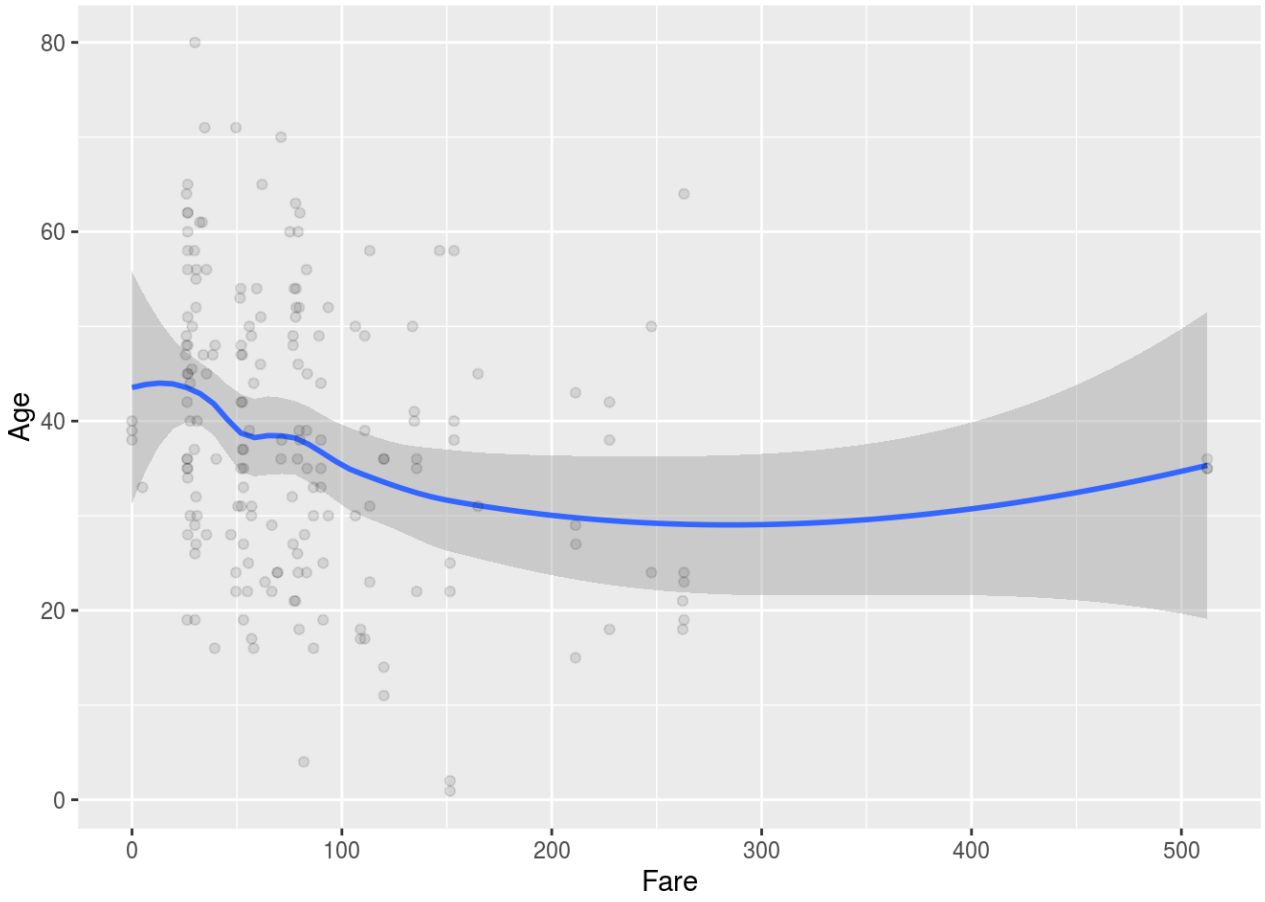

```
ggplot(ds_m, aes(Parch,Age)) + stat_smooth() + geom_point(alpha=0.1)
```



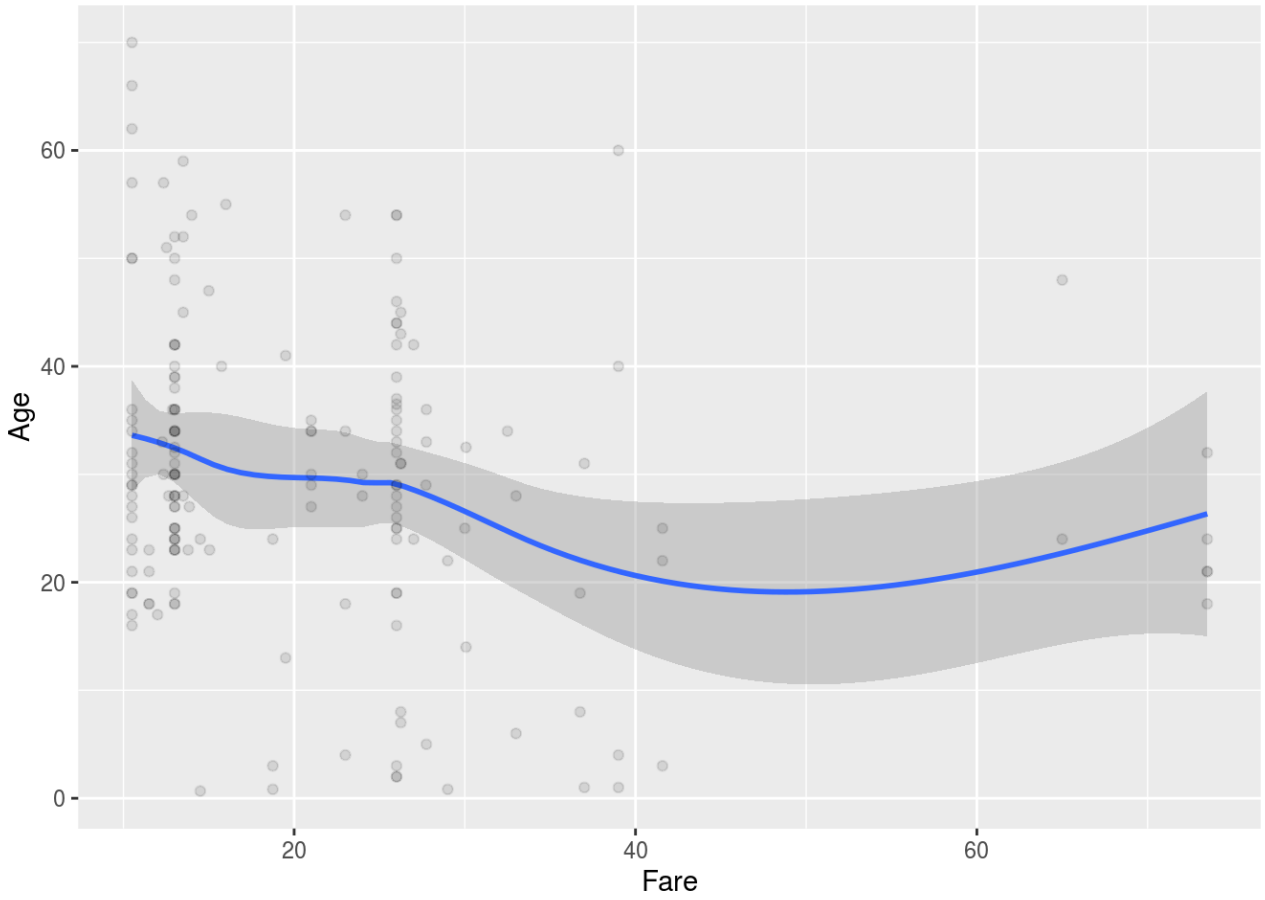

```
ggplot(ds_m[!(ds_m$Embarked==""),], aes(x=Age, fill=Embarked, colour=Embarked
)) + geom_density(adjust=1, alpha=0.1)
```



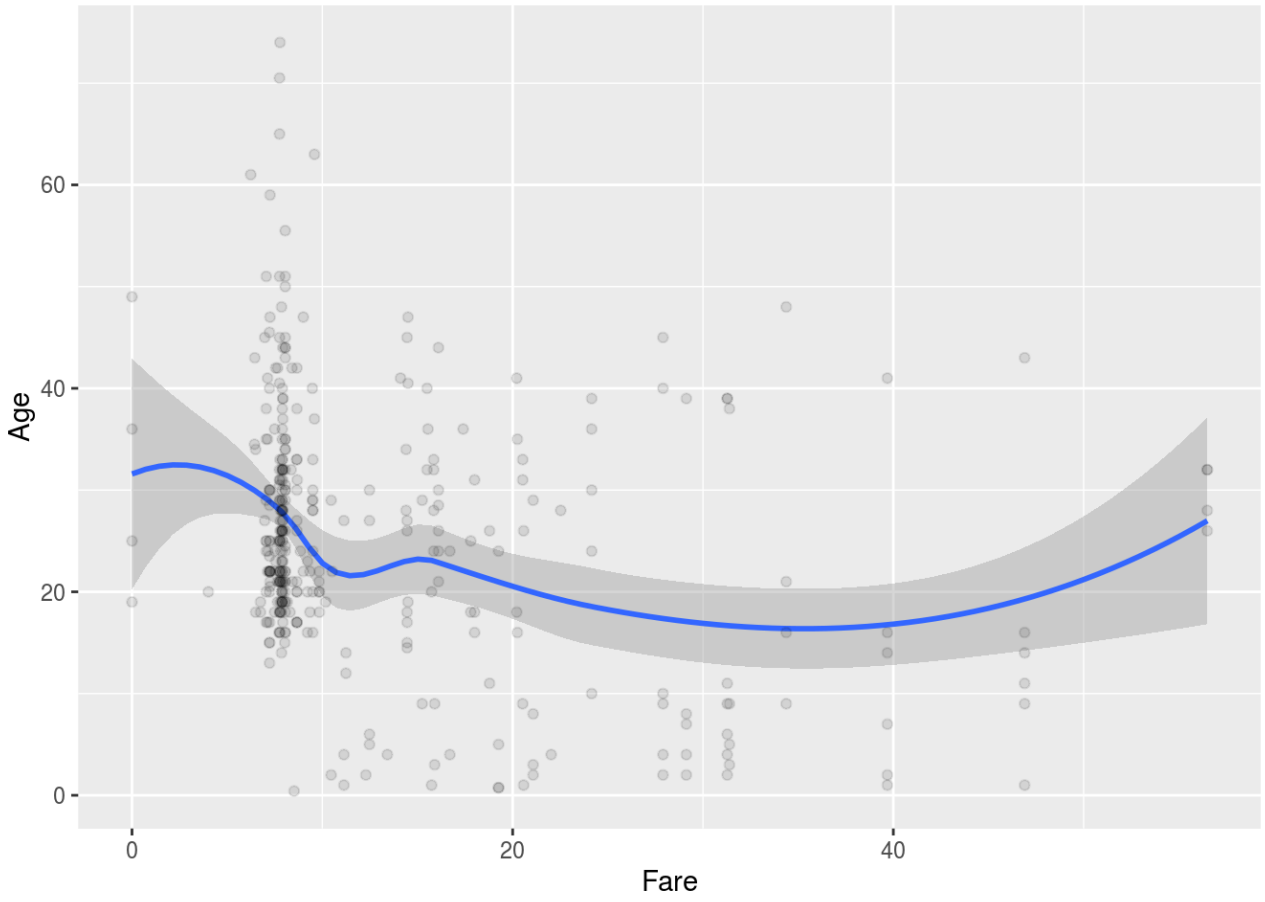

```
ggplot(ds_m[ds_m$Pclass==1,], aes(Fare,Age)) + stat_smooth() + geom_point(alp
ha=0.1)
```



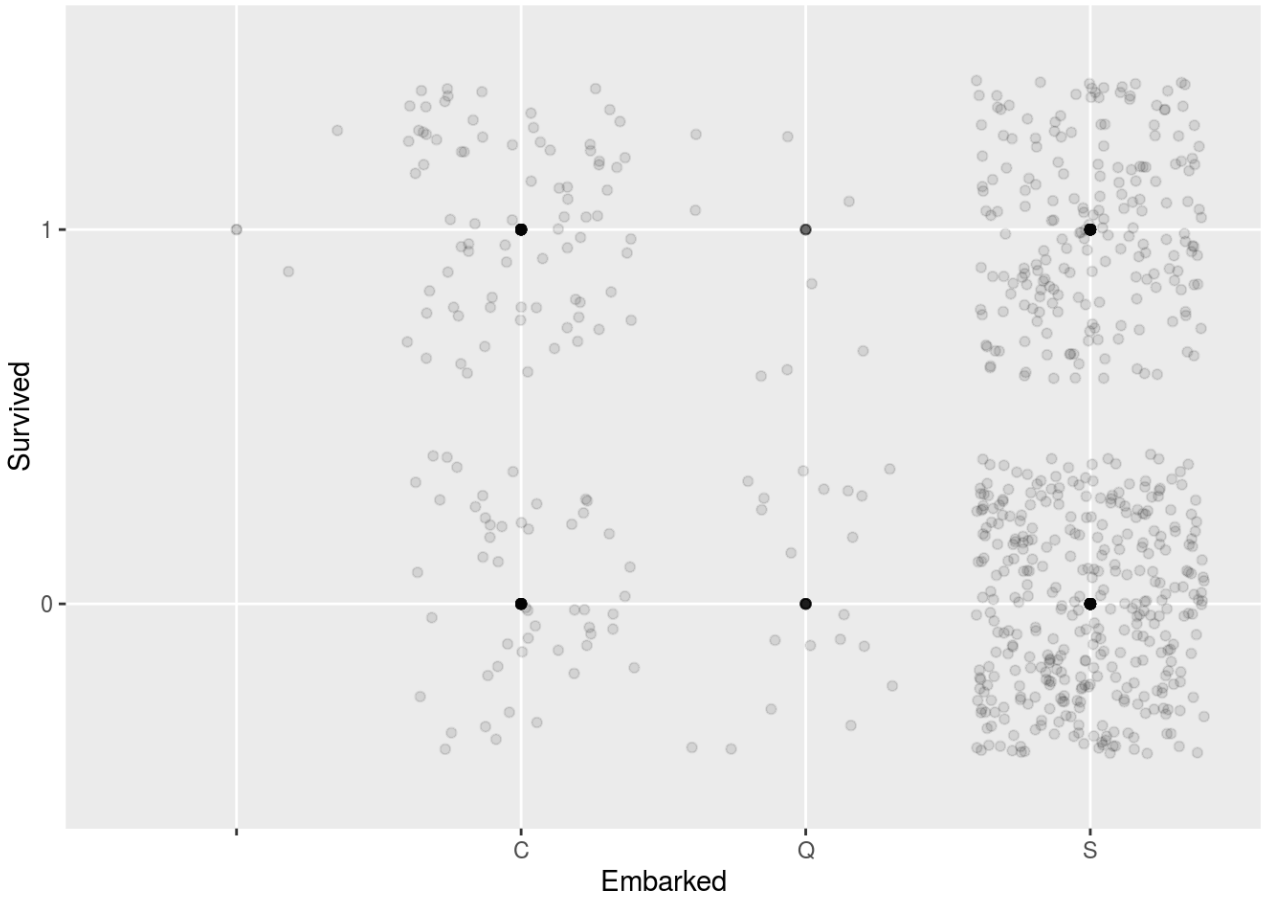

```
ggplot(ds_m[ds_m$Pclass==2,], aes(Fare,Age)) + stat_smooth() + geom_point(alp
ha=0.1)
```



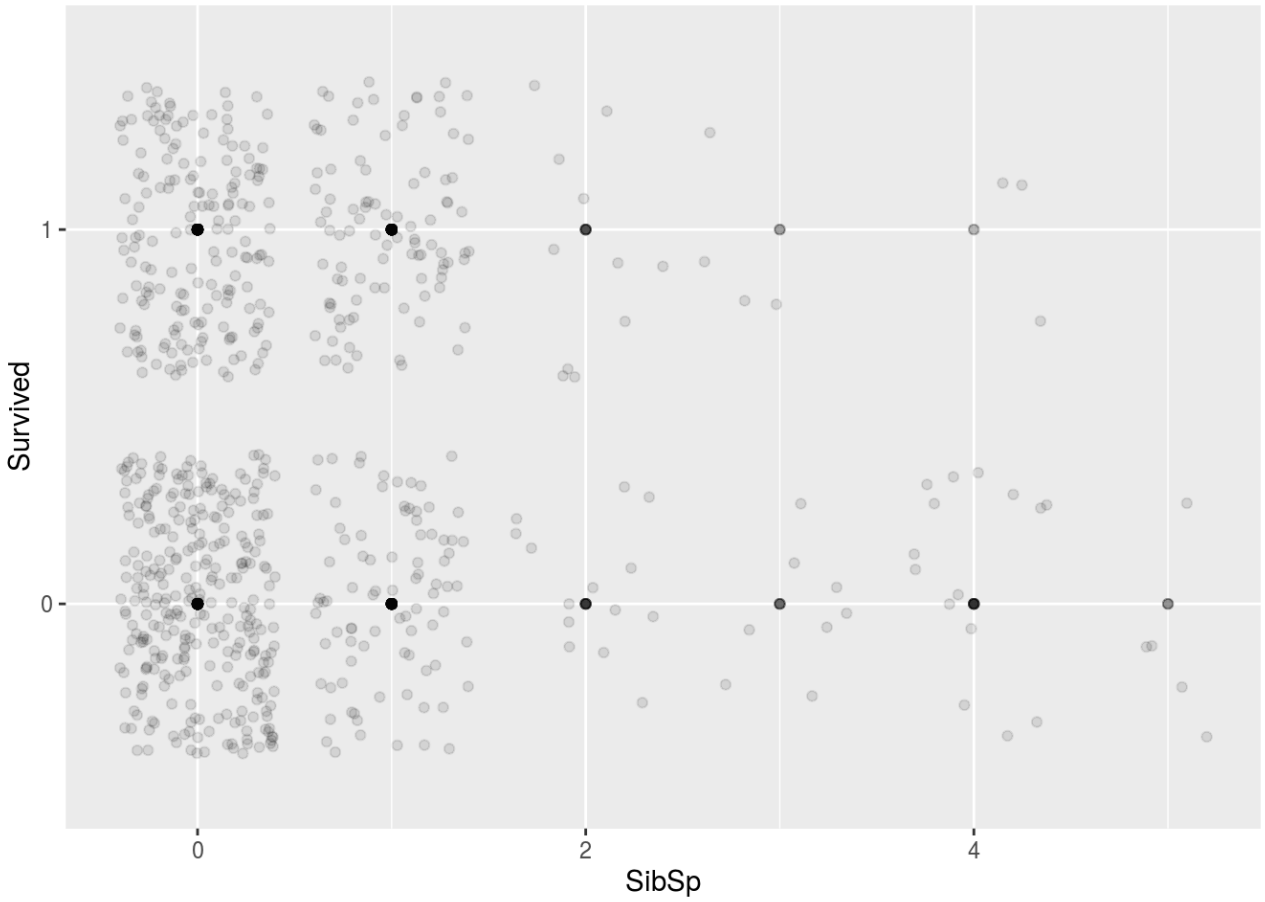

```
ggplot(ds_m[ds_m$Pclass==3,], aes(Fare,Age)) + stat_smooth() + geom_point(alp
ha=0.1)
```



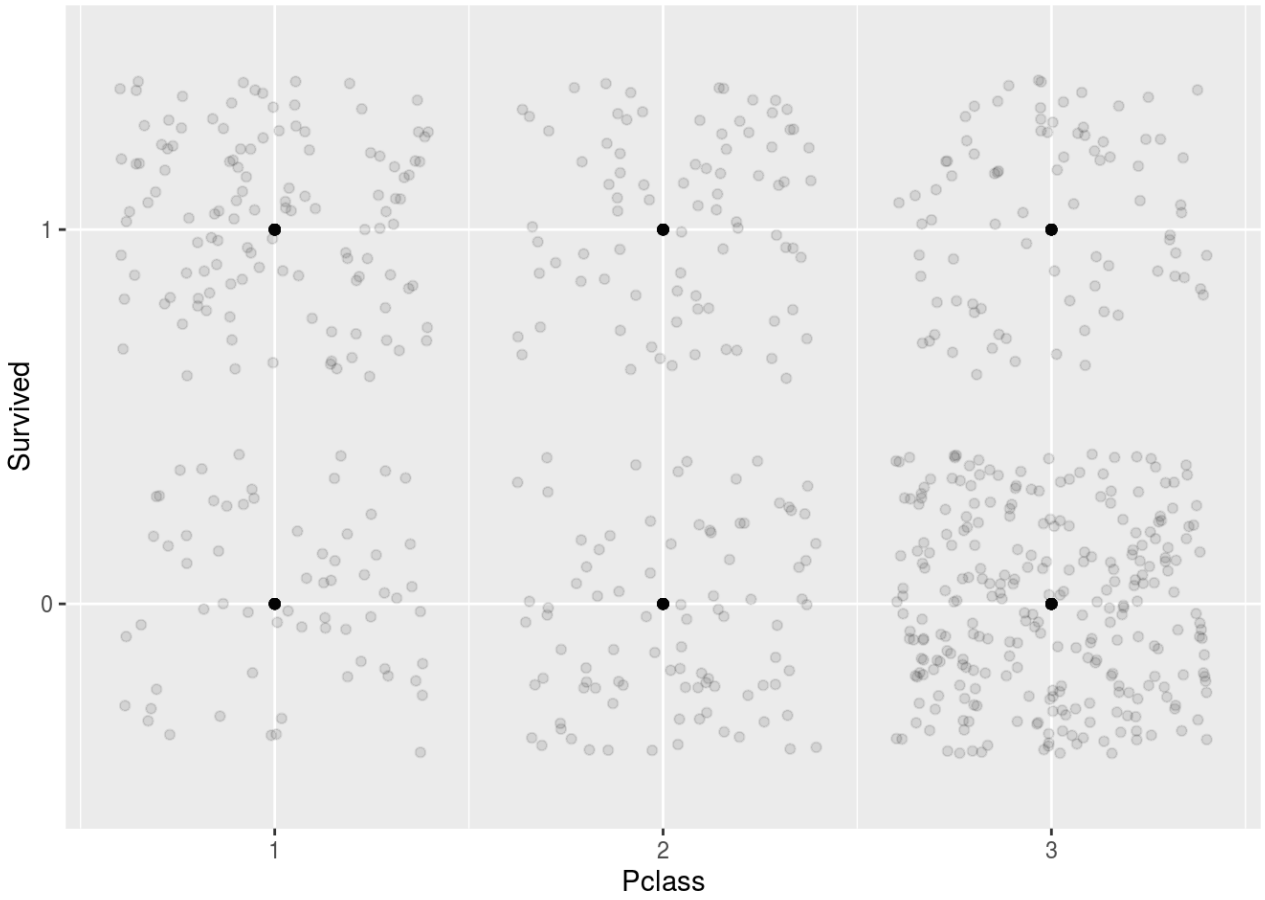

```
ggplot(ds_m, aes(Embarked, Survived)) +geom_jitter(alpha=0.1) + geom_point(al
pha=0.1)
```



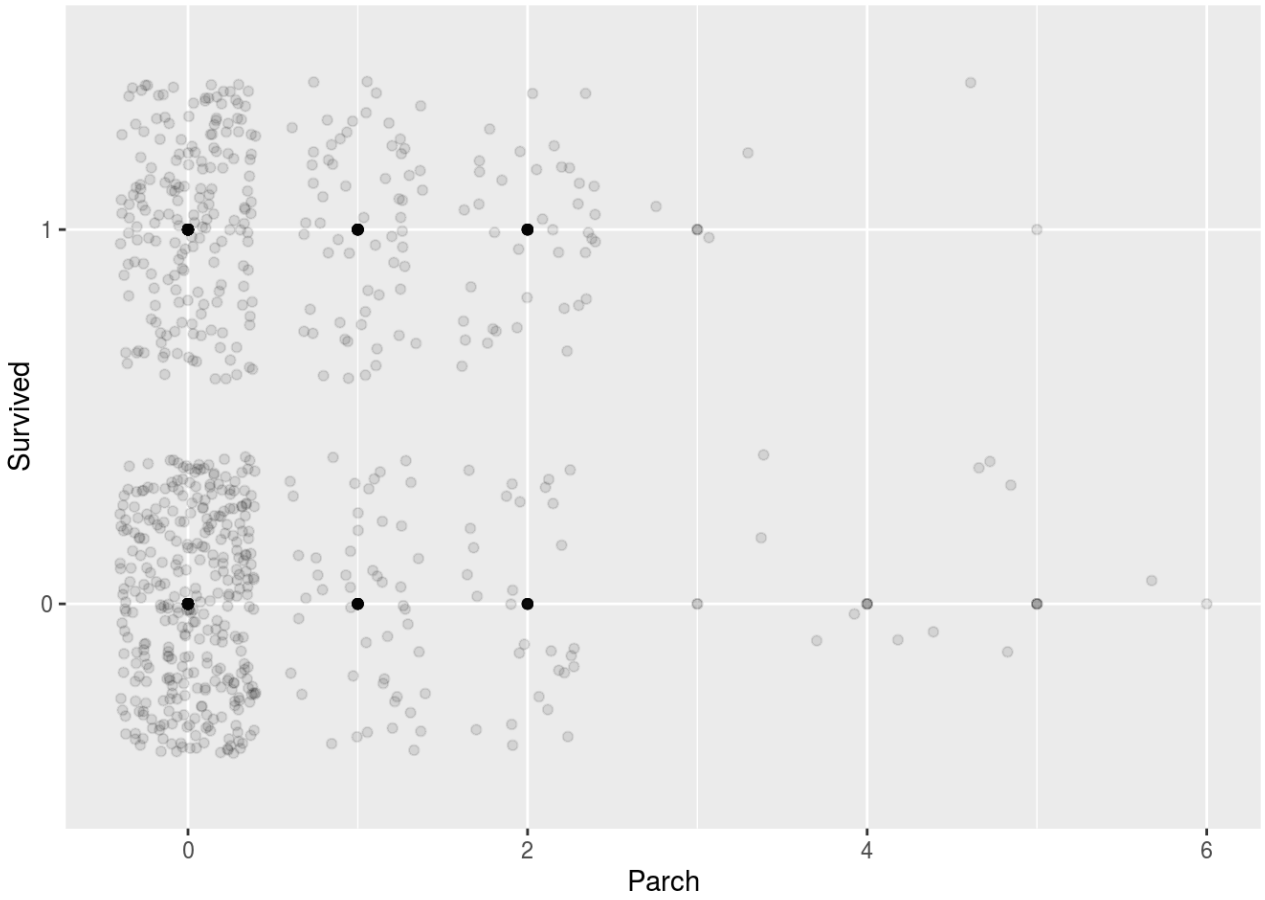

```
ggplot(ds_m, aes(SibSp, Survived)) +geom_jitter(alpha=0.1) + geom_point(alpha
=0.1)
```



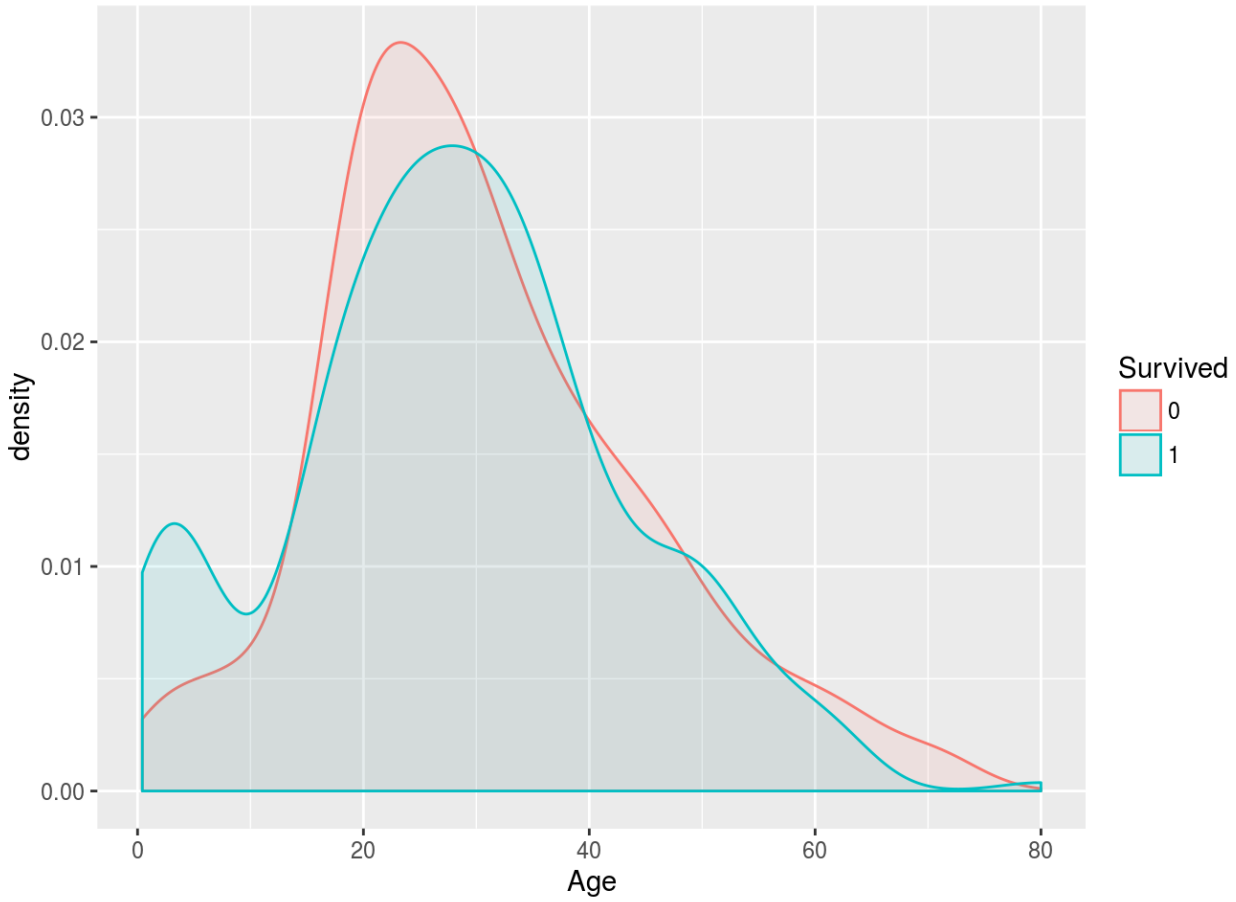

```
ggplot(ds_m, aes(Pclass, Survived)) +geom_jitter(alpha=0.1) + geom_point(alph
a=0.1)
```



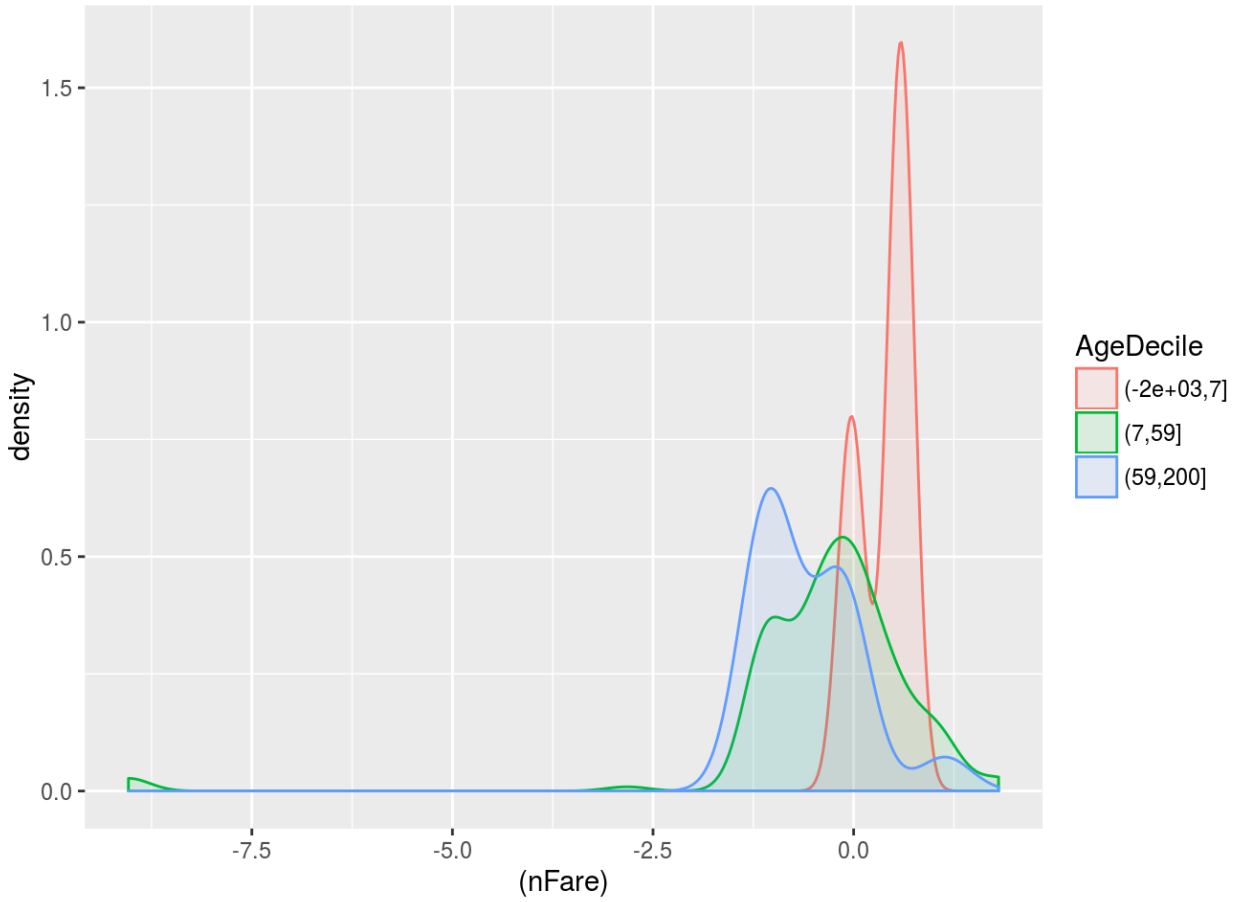

```
ggplot(ds_m, aes(Parch, Survived)) +geom_jitter(alpha=0.1) + geom_point(alpha
=0.1)
```



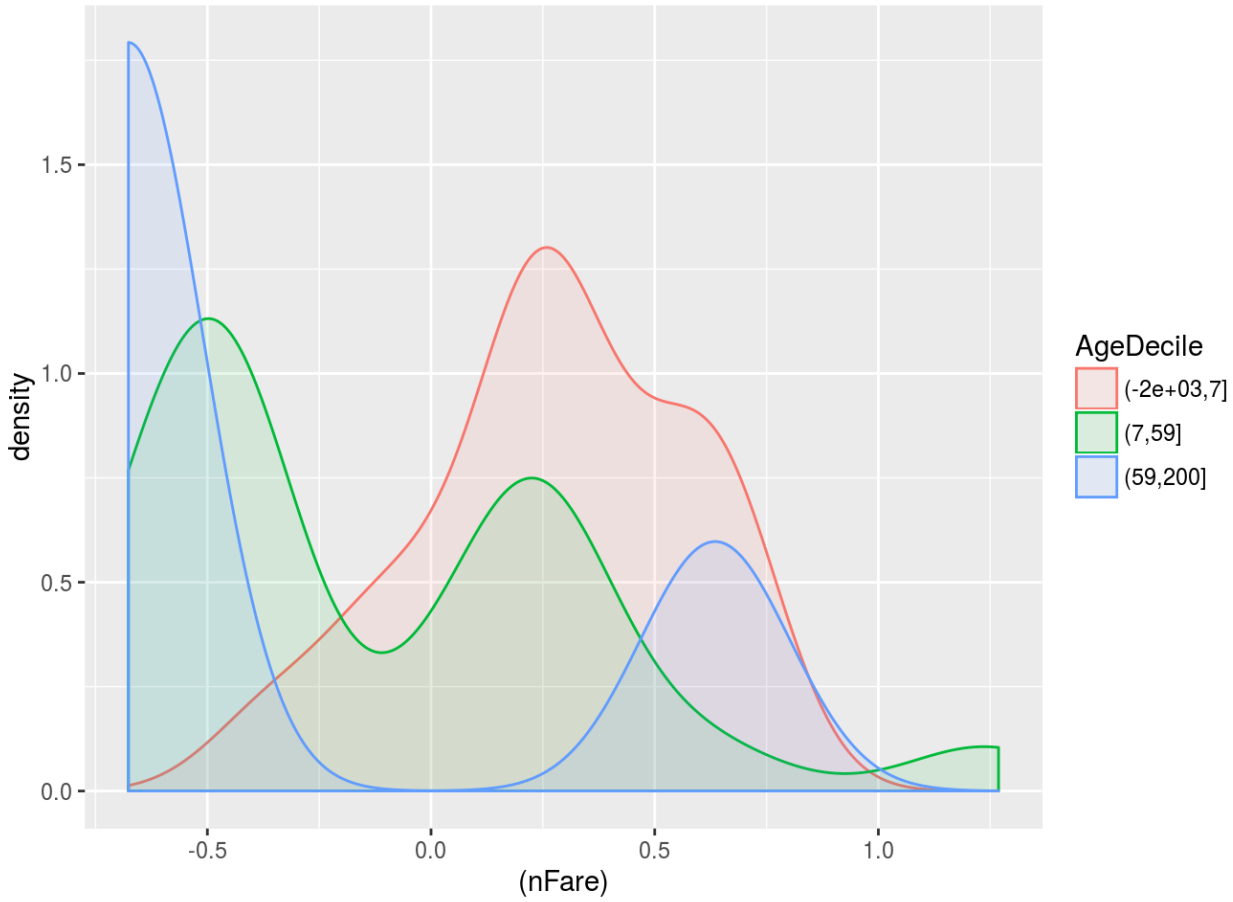

```
ggplot(ds_m, aes(x=Age, fill=Survived, colour=Survived)) + geom_density(adjus
t=1, alpha=0.1)
```



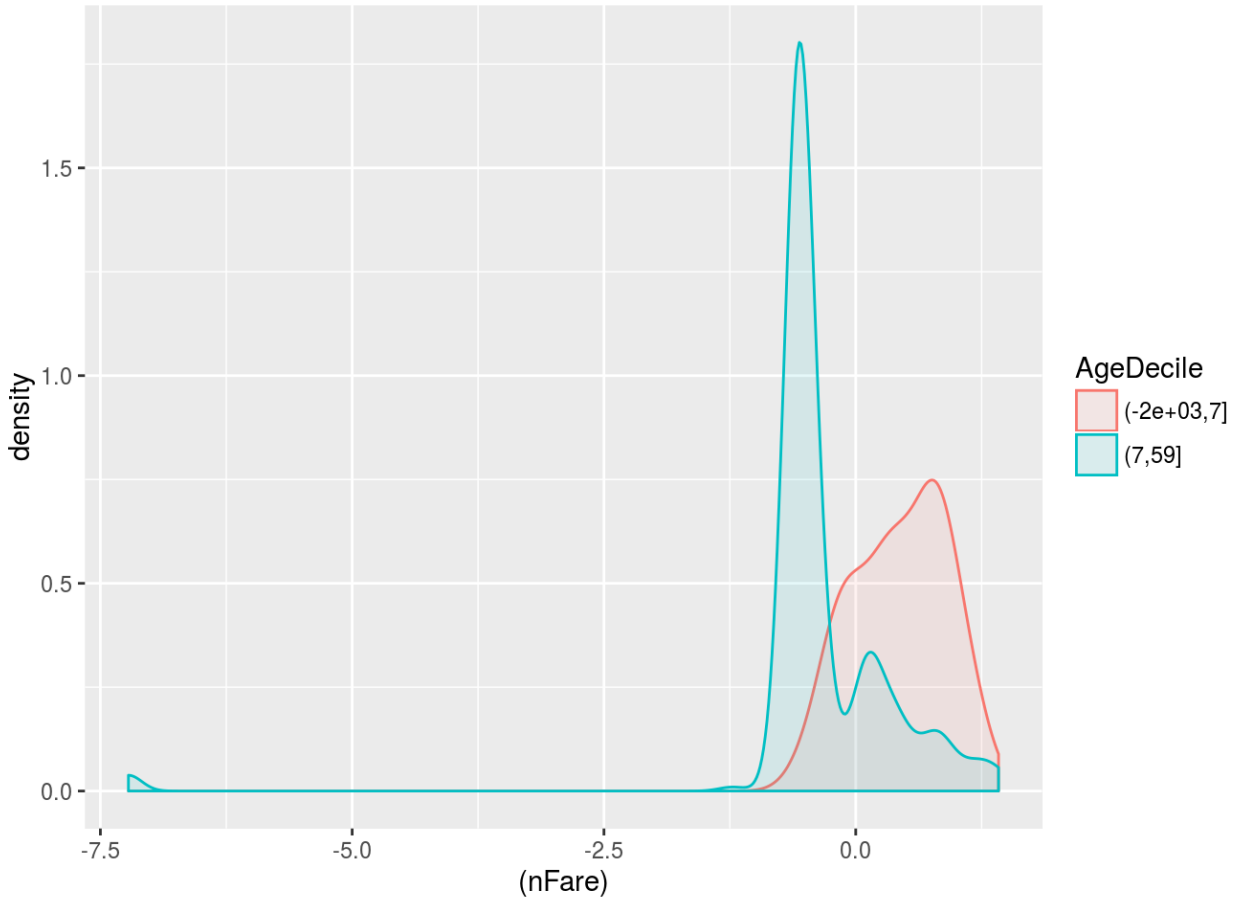

```
ds_m$AgeDecile = cut(ds_m$Age, c(-2000,7,59,200))

ggplot(ds_m[ds_m$Pclass==1,], aes(x=(nFare), fill=AgeDecile, colour=AgeDecile
)) + geom_density(adjust=1, alpha=0.1)
```




```
ggplot(ds_m[ds_m$Pclass==2,], aes(x=(nFare), fill=AgeDecile, colour=AgeDecile
)) + geom_density(adjust=1, alpha=0.1)
```




```
ggplot(ds_m[ds_m$Pclass==3 & as.integer(ds_m$AgeDecile)<3,], aes(x=(nFare), f
ill=AgeDecile, colour=AgeDecile)) + geom_density(adjust=1, alpha=0.1)
```



Supporting Functions

```
#### function to count number of space in the string
getCountSp <- function(str){
  m<-gregexpr("\\s", str, perl=T)
  return(unlist(length(regmatches(str,m)[[1]])))
}

#### function to generate lookup for cnt per ticket
generate_cnt_by_Ticket<-function(dtf){
  # Create variable for prob to Survive within the same cabin: same cabin usu
ally has same outcome
  dtf<-data.frame(Ticket=dtf$Ticket, Survived=as.integer(dtf$Survived) -1)
  dt<-data.table(dtf)
  lookup<-dt[,list(cnt_ticket = .N), by=Ticket]
  return(lookup)
}

#### function to generate samples from Beta distribution; piror dist. = Beta(
1,1) : average = 0.5; posterior = Beta(1+Survived, 1+dead)
generate_BAY_p_ds<-function(i, dtest){
  p <- rbeta(1, dtest$SurvivedTotal[i] + 1, dtest$nsample[i] - dtest$Survived
Total[i] + 1 )
  return (p)
}

#### function to generate lookup for cnt per Grpticket and number of survived
 per GrpTicket
generate_n_Survived_GrpTicket<-function(dtf){
  # Create variable for prob to Survive within the same cabin: same cabin usu
ally has same outcome
  dtf<-data.frame(GrpTicket=dtf$GrpTicket, Survived=as.integer(dtf$Survived)
-1)
  dt<-data.table(dtf)
  lookup<-dt[,list(SurvivedTotal = mean(Survived) * .N, nsample = .N), by=Grp
Ticket]
  return(lookup)
}

#### function to add information in lookup by name into data set
add_info_by_name <- function(dtr, name_, lookup){
  dtr <- merge(x=dtr, y=lookup, by=name_, all.x=T)
  return(dtr)
}

#### function to create variables before the Age prediction
pretreatment <- function(ds){
  ds$Pclass = as.factor(ds$Pclass)
```

```
  # Get the title as predictor for Age
  ds$title=""
  ds$title[grepl("Mr\\.", ds$Name)] = "Mr"
  ds$title[grepl("Mrs\\.", ds$Name)] = "Mrs"
  ds$title[grepl("Miss\\.", ds$Name)] = "Miss"
  ds$title[grepl("Master\\.", ds$Name)] = "Master"
  ds$title[grepl("Dr\\.", ds$Name)] = "Dr"
  ds$title[(ds$Sex=="male") & (ds$Age > 12) & (ds$title=="")]="Mr"
  ds$title[(ds$Sex=="male") & (ds$Age <= 11) & (ds$title=="")]="Master"
  ds$title[(ds$Sex=="female") & (ds$Age <= 11) & (ds$title=="")]="Miss"
  ds$title = as.factor(ds$title)
  # Create dummy variable isSingle...
  ds$noSibSp = as.factor(ds$SibSp == 0)
  ds$isSingle <- as.factor(ds$Parch==0)
  ds$isSinglenoSibSp <- as.factor(ds$SibSp == 0 & ds$Parch==0 )
  ds$hasCabin = as.factor(ds$Cabin=="")
  ds$CabinLet = as.factor(substr(ds$Cabin, 1, 1))
  ds$CabinCnt = as.integer(!ds$Cabin=="") + sapply(ds$Cabin,getCountSp, simpl
ify=T)
  ds$familySz = ds$SibSp + ds$Parch

  ds$ln_Fare = log(ds$Fare+0.01)
  ds$nFare[ds$Pclass==1] = log(ds$Fare[ds$Pclass==1] +0.01) - log(mean(ds$Far
e[ds$Pclass==1])+0.01)
  ds$nFare[ds$Pclass==2] = log(ds$Fare[ds$Pclass==2] +0.01) - log(mean(ds$Far
e[ds$Pclass==2])+0.01)
  ds$nFare[ds$Pclass==3] = log(ds$Fare[ds$Pclass==3] +0.01) - log(mean(ds$Far
e[ds$Pclass==3])+0.01)
  ds$nFare[is.na(ds$nFare)] = 0

  ds$hasNoChildren = (ds$SibSp==1 & ds$Parch==0)
  ds$r_SibSp_Parch = ((ds$SibSp+0.01)/(ds$SibSp+ds$Parch+0.01))

  # Get total number of people per ticket
  lookup<-generate_cnt_by_Ticket(ds)
  ds<-add_info_by_name(ds,"Ticket",lookup)
  # Define the group ticket that has more than 2 people
  ds$GrpTicket = "XXXXXXX"
  ds$GrpTicket[ds$cnt_ticket>1] = as.character(ds$Ticket[ds$cnt_ticket>1])

  return(ds)
}

#### function to do Age Prediction
# first stage using linear regression to deal with the interaction
  # Predict Age for missings
  # if isSingle and title is Miss, could be older
```

```
  # if isSingle and has a lot of SibSp, could be younger
  # if Pclass is controlled, Fare should reflect the age
  # using log(Age) because Age distribution is right-skewed... except the bab
y... but mostly right skewed
lm_Age <- function(ds) {
  ds_m <- ds[!is.na(ds$Age),]
  lr_Age<- lm(I(log(Age+0.0001))~title*isSingle + isSingle*SibSp + Parch*I(Si
bSp>1) + Pclass*nFare + hasCabin*hasNoChildren, data=ds_m)
  print(summary(lr_Age))
  return(lr_Age)
}
# second stage using rf for partition
md_Age <- function(ds) {
  #
  #ds_m <- reSamplebyAge(ds[!is.na(ds$Age),], c(-2000,11,17,30,60,200))
  ds_m <- ds[!is.na(ds$Age),]
  trf_Age<- randomForest(I(log(Age+0.0001))~ title + familySz + pAge_lr + Sib
Sp + Pclass + nFare  + CabinLet + isSinglenoSibSp +  hasNoChildren + r_SibSp_
Parch, data=ds_m, importance=TRUE,proximity=TRUE,ntree=500)
  print(varImp(trf_Age))
  return(trf_Age)
}

#### function to create variables after the Age prediction
psttreatment <- function(lm_Ager, trf_Age, ds) {
  ds_m <- ds
  ds_m$pAge_lr = exp(predict(lm_Ager, ds_m))
  ds_m$AgePred <- exp(predict(trf_Age, ds_m))
  ds_m$AgeDecile = cut(ds_m$Age, c(-2000,11,17,30,60,200))
  ds_m$AgePredDecile = cut(ds_m$AgePred, c(-2000,11,17,30,60,200))
  print(table(ds_m$AgeDecile[!is.na(ds_m$Age)],ds_m$AgePredDecile[!is.na(ds_m
$Age)]))
  # replace NA
  ds_m$Age[is.na(ds_m$Age)]<-ds_m$AgePred[is.na(ds_m$Age)]
  # cut by decile
  ds_m$AgeDecile = cut(ds_m$Age, c(-2000,11,17,30,60,200))

  return(ds_m)
}


#### function to run the prediction
run_prediction <- function(tgbm2, trf, tsvm, dtest){
  ## Run the data through the model
  dtest$pred.tgbm2 = predict(tgbm2, dtest, "raw")
  dtest$pred.rf = predict(trf, dtest, "raw")
  dtest$pred.svm = predict(tsvm, dtest, "raw")
```

```
  dtest$pred.vote = as.factor(
                  as.integer(
                      (as.integer(dtest$pred.rf)-1 + as.integer(dtest$pred.sv
m)-1 + as.integer(dtest$pred.tgbm2)-1)> 1.5
                    )
                  )
  return(dtest)
}
```

Data Manipulation

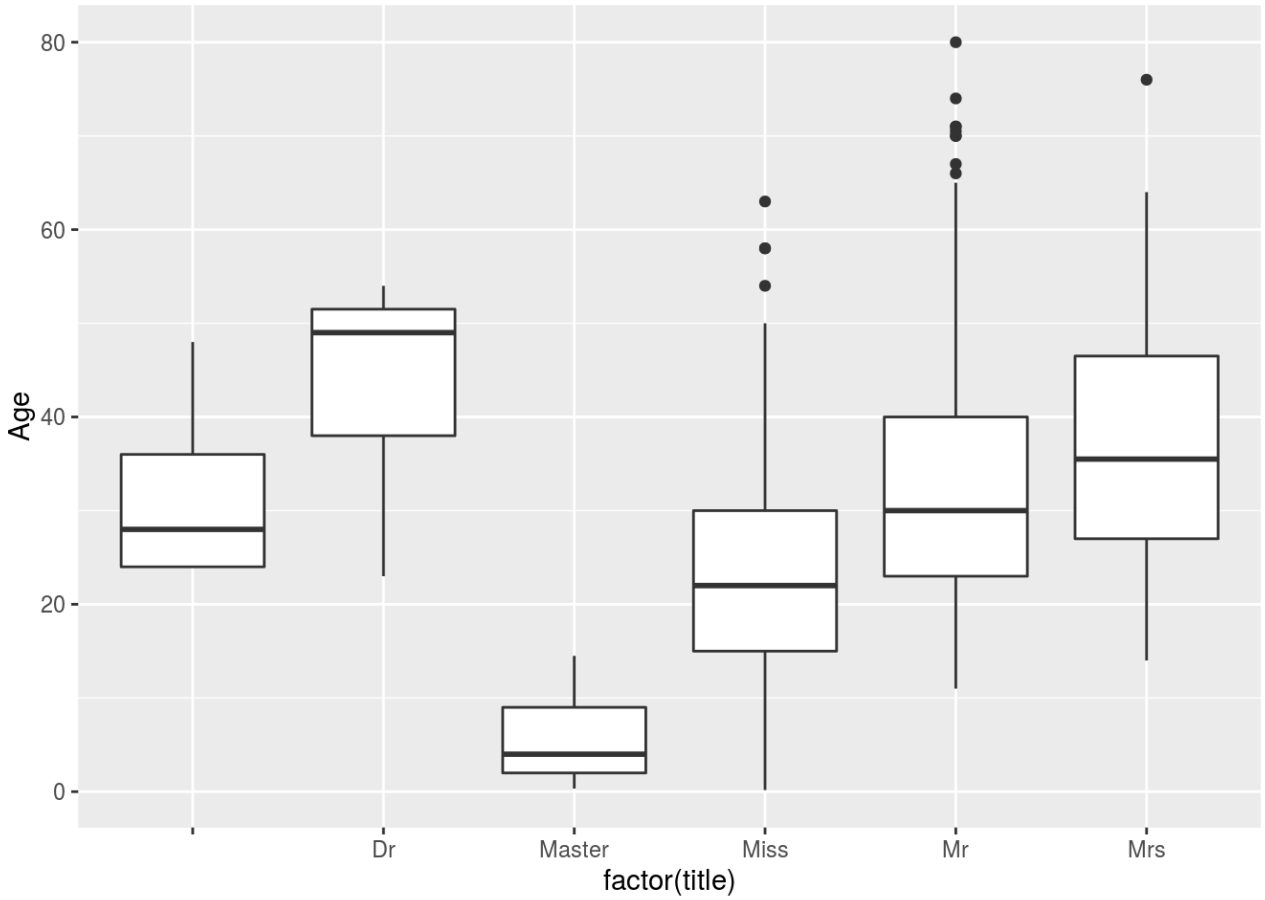
```
dall<-pretreatment(dall)
dall$GrpTicket = as.factor(dall$GrpTicket)
lm_Ager <-  lm_Age(dall)
```

```
##
## Call:
## lm(formula = I(log(Age + 1e-04)) ~ title * isSingle + isSingle *
##     SibSp + Parch * I(SibSp > 1) + Pclass * nFare + hasCabin *
##     hasNoChildren, data = ds_m)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5754 -0.2286  0.0163  0.2583  1.4469
##
## Coefficients: (2 not defined because of singularities)
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    3.401903   0.209015  16.276  < 2e-16 ***
## titleDr                        0.484928   0.523685   0.926  0.35467
## titleMaster                   -1.905229   0.214962  -8.863  < 2e-16 ***
## titleMiss                     -1.249825   0.209566  -5.964 3.39e-09 ***
## titleMr                        0.221971   0.208219   1.066  0.28666
## titleMrs                       0.322536   0.192400   1.676  0.09397 .
## isSingleTRUE                   0.086370   0.104714   0.825  0.40966
## SibSp                          0.013576   0.044052   0.308  0.75800
## Parch                          0.070146   0.033862   2.072  0.03856 *
## I(SibSp > 1)TRUE              -0.519364   0.220030  -2.360  0.01844 *
## Pclass2                       -0.379821   0.063447  -5.986 2.97e-09 ***
## Pclass3                       -0.527468   0.061712  -8.547  < 2e-16 ***
## nFare                          0.003895   0.023287   0.167  0.86719
## hasCabinTRUE                   0.025534   0.059887   0.426  0.66993
## hasNoChildrenTRUE             -0.423573   0.131288  -3.226  0.00129 **
## titleDr:isSingleTRUE          -0.155426   0.530120  -0.293  0.76944
## titleMaster:isSingleTRUE       1.401654   0.496886   2.821  0.00488 **
## titleMiss:isSingleTRUE         1.325949   0.109442  12.116  < 2e-16 ***
## titleMr:isSingleTRUE           0.064780   0.100858   0.642  0.52083
## titleMrs:isSingleTRUE                NA         NA      NA       NA
## isSingleTRUE:SibSp             0.218126   0.106505   2.048  0.04081 *
## Parch:I(SibSp > 1)TRUE         0.432673   0.124658   3.471  0.00054 ***
## Pclass2:nFare                 -0.069956   0.067181  -1.041  0.29798
## Pclass3:nFare                        NA         NA      NA       NA
## hasCabinTRUE:hasNoChildrenTRUE  0.194022   0.089423   2.170  0.03026 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4831 on 1023 degrees of freedom
## Multiple R-squared:  0.626,  Adjusted R-squared:  0.6179
## F-statistic: 77.82 on 22 and 1023 DF,  p-value: < 2.2e-16
```

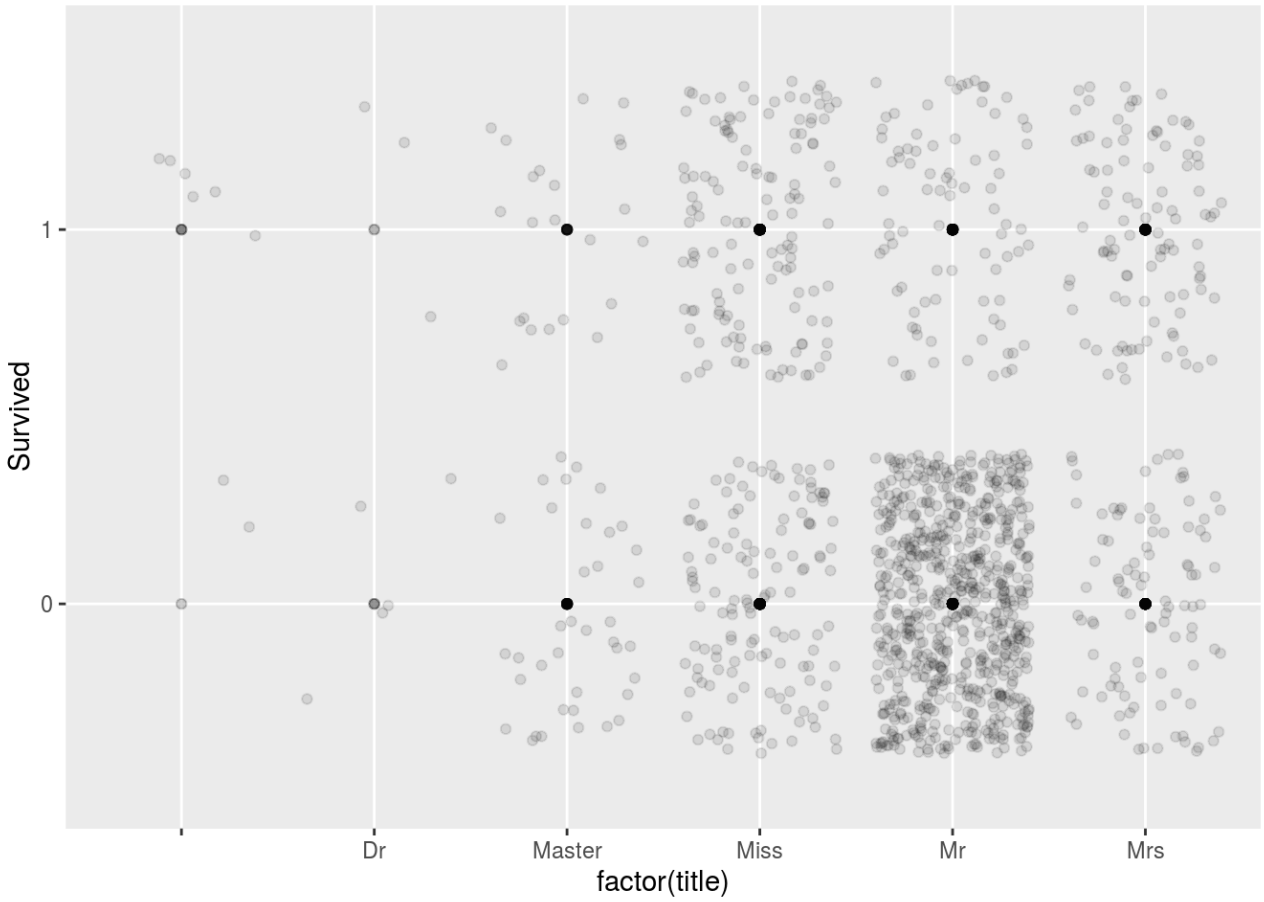
```
dall$pAge_lr <- exp(predict(lm_Ager, dall))
trf_Age <- md_Age(dall)
```

```
##                   Overall
## title           20.640984
## familySz        11.541341
## pAge_lr         33.876516
## SibSp           12.970200
## Pclass          20.398718
## nFare           20.137060
## CabinLet         6.983252
## isSinglenoSibSp  6.484454
## hasNoChildren    8.080118
## r_SibSp_Parch   12.858146
```

```
ggplot(dall, aes(factor(title),Age)) + geom_boxplot()
```




```
ggplot(dall, aes(factor(title),Survived))  + geom_point(alpha=0.1) +geom_jitt
er(alpha=0.1)
```




```
dall<-psttreatment(lm_Ager, trf_Age,dall)
```

```
##
##                (-2e+03,11] (11,17] (17,30] (30,60] (60,200]
##   (-2e+03,11]           87       1       2       0        0
##   (11,17]                9       6      48       1        0
##   (17,30]                5       5     346      99        0
##   (30,60]                1       0     133     270        0
##   (60,200]               0       0       8      25        0
```

```
ds <- dall[dall$src == 0,]
dtest <- dall[dall$src == 1,]
```

Modeling

```
ctrl = trainControl(method="repeatedcv", number=20, repeats=10, selectionFunc
tion = "oneSE")
in_train = createDataPartition(ds$Survived, p=.90, list=FALSE)

dtr <- ds[in_train,]
lookup<-generate_n_Survived_GrpTicket(dtr)
dtr<-add_info_by_name(dtr,"GrpTicket", lookup)
dtr$p_Survived<-sapply(1 : length(dtr$nsample), generate_BAY_p_ds, dtest =dtr
)

# train model using multiple methods

tune_grid <-  expand.grid(interaction.depth = c(1, 3, 9),
                          n.trees = (1:25)*5,
                          shrinkage = 0.1,
                          n.minobsinnode = 2)
tgbm2 =
  train(
    Survived ~  Sex + p_Survived + cnt_ticket + title + Pclass + CabinCnt + C
abinLet + I(CabinLet=="") + AgeDecile + SibSp + Parch + Embarked +
              familySz + nFare +isSinglenoSibSp:r_SibSp_Parch, data=dtr, meth
od="gbm",
    tuneGrid=tune_grid, preProc = c("center", "scale"), metric="Kappa", trCon
trol=ctrl, verbose=FALSE
  )

trf =
  train(
    Survived ~  Sex + p_Survived + cnt_ticket + title + Pclass + CabinCnt + C
abinLet + I(CabinLet=="") + AgeDecile + SibSp + Parch + Embarked +
              familySz + nFare +isSinglenoSibSp:r_SibSp_Parch,
    data=dtr, method="rf", metric="Kappa", trControl=ctrl, verbose=FALSE, ntr
ee=350
  )

#method="svmLinear",
tsvm =
  train(
    Survived ~  Sex + p_Survived + cnt_ticket + title + Pclass + CabinCnt + C
abinLet + I(CabinLet=="") + AgeDecile + SibSp + Parch + Embarked +
              familySz + nFare +isSinglenoSibSp:r_SibSp_Parch,
    data=dtr, method="svmPoly",tuneLength=4, metric="Kappa", trControl=ctrl,
verbose=FALSE, preProc = c("center", "scale")
  )

## Variables importance
```

```
varImp(tgbm2)
```

```
## gbm variable importance
##
##   only 20 most important variables shown (out of 33)
##
##                                    Overall
## titleMr                           100.0000
## p_Survived                         68.9845
## nFare                              17.1572
## Pclass3                            14.2305
## Sexmale                            13.5918
## cnt_ticket                          5.1280
## CabinCnt                            4.1965
## CabinLetE                           3.3770
## EmbarkedQ                           1.9507
## AgeDecile(30,60]                    1.6356
## CabinLetB                           1.2045
## EmbarkedS                           1.1335
## SibSp                               0.8699
## Pclass2                             0.8539
## familySz                            0.6406
## Parch                               0.6350
## titleMrs                            0.6277
## I(CabinLet == "")TRUE               0.5803
## isSinglenoSibSpFALSE:r_SibSp_Parch  0.5379
## CabinLetD                           0.5044
```

```
varImp(trf)
```

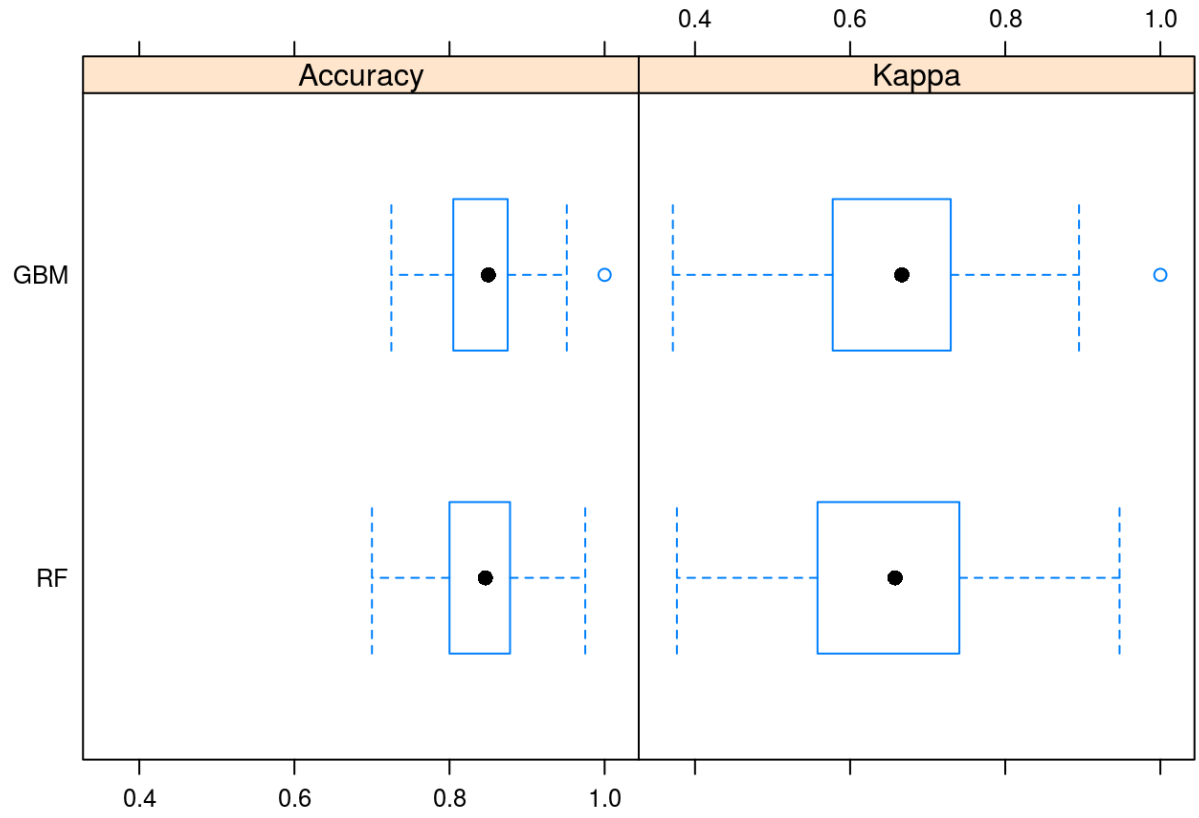
```
## rf variable importance
##
##   only 20 most important variables shown (out of 33)
##
##                                     Overall
## p_Survived                          100.000
## titleMr                              55.766
## Sexmale                              33.021
## nFare                                18.140
## Pclass3                              12.103
## cnt_ticket                           11.218
## familySz                              8.446
## isSinglenoSibSpFALSE:r_SibSp_Parch    4.694
## CabinCnt                              4.568
## I(CabinLet == "")TRUE                 4.258
## AgeDecile(30,60]                      4.005
## SibSp                                 3.714
## EmbarkedS                             3.533
## AgeDecile(17,30]                      3.188
## CabinLetE                             3.050
## EmbarkedC                             2.933
## Parch                                 2.770
## Pclass2                               2.704
## EmbarkedQ                             2.221
## titleMiss                             2.002
```

Resampling

```
resampls = resamples(list(RF = trf,
                          GBM = tgbm2))
difValues = diff(resampls)
summary(difValues)
```

```
##
## Call:
## summary.diff.resamples(object = difValues)
##
## p-value adjustment: bonferroni
## Upper diagonal: estimates of the difference
## Lower diagonal: p-value for H0: difference = 0
##
## Accuracy
##     RF     GBM
## RF         -0.002699
## GBM 0.6343
##
## Kappa
##     RF     GBM
## RF         0.0004165
## GBM 0.9738
```

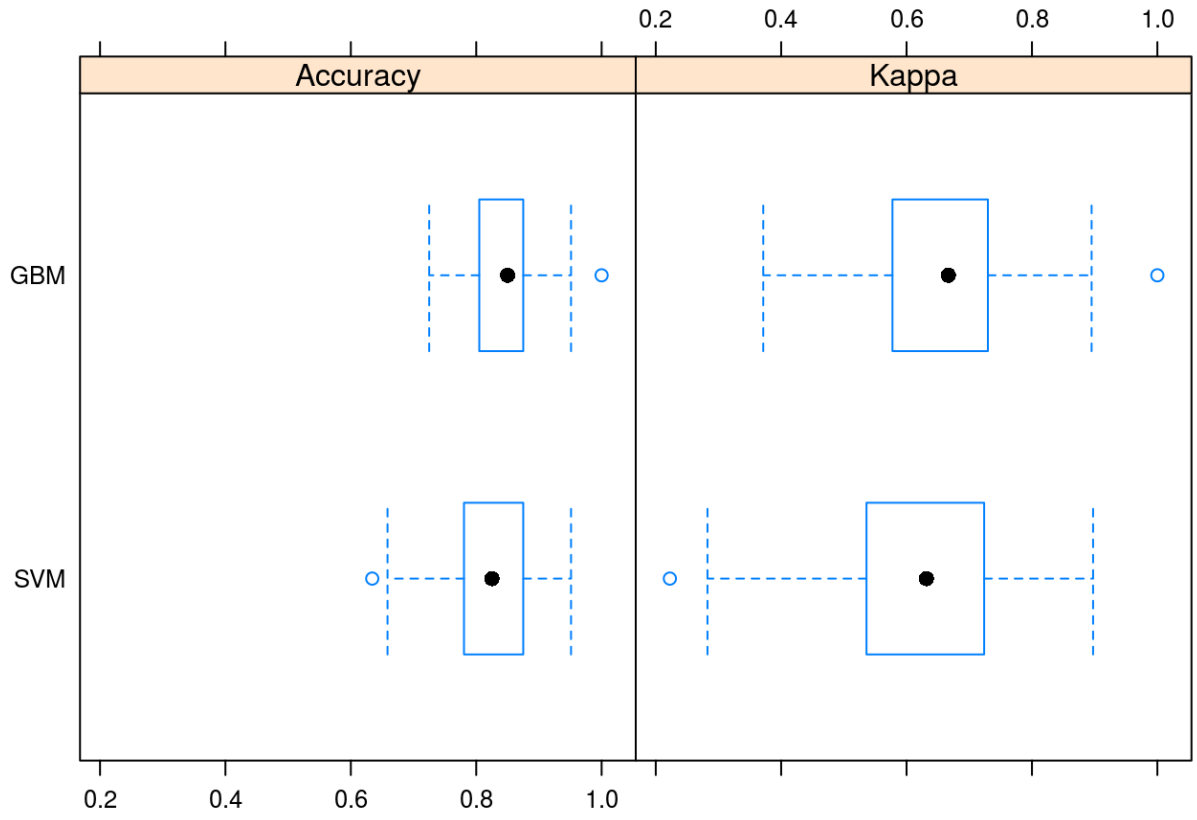
```
bwplot(resampls, layout=c(2,1))
```



```
resampls = resamples(list(SVM = tsvm,
                          GBM = tgbm2))
difValues = diff(resampls)
summary(difValues)
```

```
##
## Call:
## summary.diff.resamples(object = difValues)
##
## p-value adjustment: bonferroni
## Upper diagonal: estimates of the difference
## Lower diagonal: p-value for H0: difference = 0
##
## Accuracy
##     SVM      GBM
## SVM          -0.01811
## GBM 0.001218
##
## Kappa
##     SVM     GBM
## SVM         -0.03119
## GBM 0.01204
```

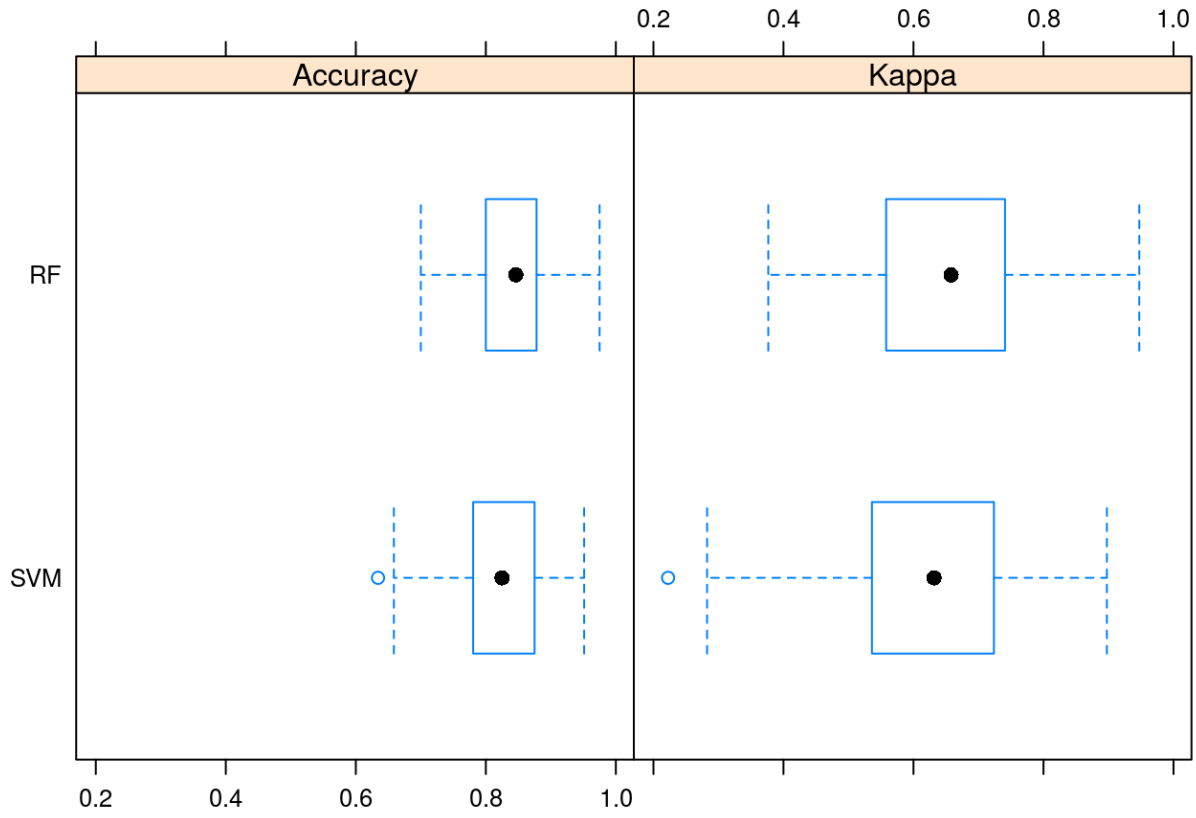
```
bwplot(resampls, layout=c(2,1))
```




```
resampls = resamples(list(RF = trf,
                          SVM = tsvm))
difValues = diff(resampls)
summary(difValues)
```

```
##
## Call:
## summary.diff.resamples(object = difValues)
##
## p-value adjustment: bonferroni
## Upper diagonal: estimates of the difference
## Lower diagonal: p-value for H0: difference = 0
##
## Accuracy
##     RF      SVM
## RF          0.01541
## SVM 0.00726
##
## Kappa
##     RF      SVM
## RF          0.03161
## SVM 0.01141
```

```
bwplot(resampls, layout=c(2,1))
```



Verification Using Portion of Training Set

```
test = ds[-in_train,]

test<-add_info_by_name(test,"GrpTicket", lookup)
#backgroud survived rate = Beta(1,1)
test$SurvivedTotal[is.na(test$nsample)]=0
test$nsample[is.na(test$nsample)]=0
test$p_Survived<-sapply(1 : length(test$nsample), generate_BAY_p_ds, dtest =t
est)

test<-run_prediction(tgbm2, trf, tsvm, test)

confusionMatrix(test$pred.tgbm2, test$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 48  7
##          1  6 27
##
##                Accuracy : 0.8523
##                  95% CI : (0.7606, 0.9189)
##     No Information Rate : 0.6136
##     P-Value [Acc > NIR] : 8.639e-07
##
##                   Kappa : 0.6867
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8889
##             Specificity : 0.7941
##          Pos Pred Value : 0.8727
##          Neg Pred Value : 0.8182
##              Prevalence : 0.6136
##          Detection Rate : 0.5455
##    Detection Prevalence : 0.6250
##       Balanced Accuracy : 0.8415
##
##        'Positive' Class : 0
##
```

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

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 48  7
##          1  6 27
##
##                Accuracy : 0.8523
##                  95% CI : (0.7606, 0.9189)
##     No Information Rate : 0.6136
##     P-Value [Acc > NIR] : 8.639e-07
##
##                   Kappa : 0.6867
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8889
##             Specificity : 0.7941
##          Pos Pred Value : 0.8727
##          Neg Pred Value : 0.8182
##              Prevalence : 0.6136
##          Detection Rate : 0.5455
##    Detection Prevalence : 0.6250
##       Balanced Accuracy : 0.8415
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(test$pred.svm, test$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 47  7
##          1  7 27
##
##                Accuracy : 0.8409
##                  95% CI : (0.7475, 0.9102)
##     No Information Rate : 0.6136
##     P-Value [Acc > NIR] : 3.01e-06
##
##                   Kappa : 0.6645
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8704
##             Specificity : 0.7941
##          Pos Pred Value : 0.8704
##          Neg Pred Value : 0.7941
##              Prevalence : 0.6136
##          Detection Rate : 0.5341
##    Detection Prevalence : 0.6136
##       Balanced Accuracy : 0.8322
##
##        'Positive' Class : 0
##
```

```
confusionMatrix(test$pred.vote, test$Survived)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 47  7
##          1  7 27
##
##                Accuracy : 0.8409
##                  95% CI : (0.7475, 0.9102)
##     No Information Rate : 0.6136
##     P-Value [Acc > NIR] : 3.01e-06
##
##                   Kappa : 0.6645
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8704
##             Specificity : 0.7941
##          Pos Pred Value : 0.8704
##          Neg Pred Value : 0.7941
##              Prevalence : 0.6136
##          Detection Rate : 0.5341
##    Detection Prevalence : 0.6136
##       Balanced Accuracy : 0.8322
##
##        'Positive' Class : 0
##
```

Generate Summit File for Competition

```
# Regenerate lookup using all training info
#lookup<-generate_n_Survived_GrpTicket(ds)
# Create variable for prob to Survive within the same cabin
dtest<-add_info_by_name(dtest,"GrpTicket", lookup)

#backgroud survived rate = Beta(1,1)
dtest$SurvivedTotal[is.na(dtest$nsample)]=0
dtest$nsample[is.na(dtest$nsample)]=0
dtest$p_Survived<-sapply(1 : length(dtest$nsample), generate_BAY_p_ds, dtest
=dtest)

## Run the data through the model
dtest<-run_prediction(tgbm2, trf, tsvm, dtest)
## Format the summit file and save to ./data/pred.csv
dSummit<-as.data.frame(cbind(dtest$PassengerId,as.integer(dtest$pred.vote)-1)
)
names(dSummit)=c("PassengerId","Survived")
write.table(dSummit,"./data/pred.csv", row.names=F, col.names = T, sep=",", q
uote=FALSE)
```