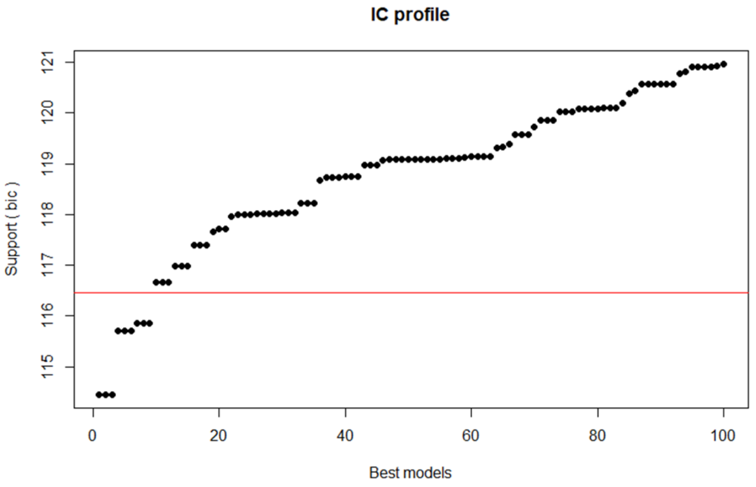
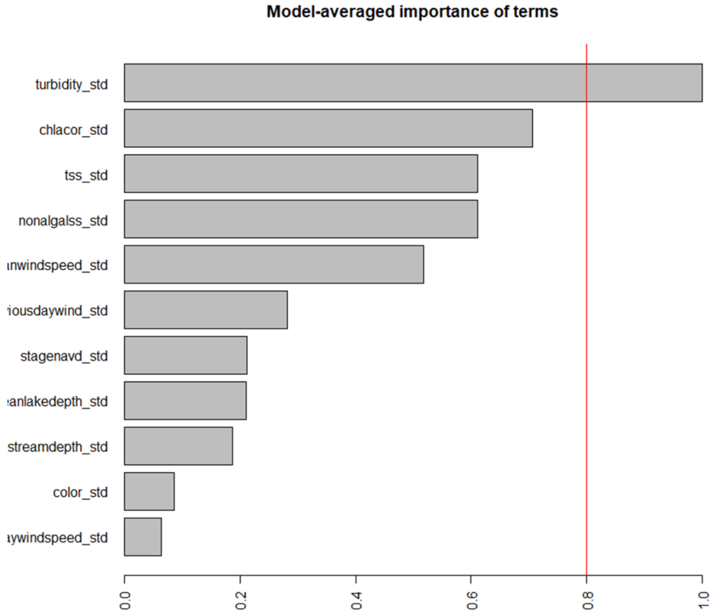
Advanced R: Statistical Machine Learning

Dan Schmutz, MS
Chief Environmental Scientist

Zoom Workshop for SJRWMD
September 24, 2020

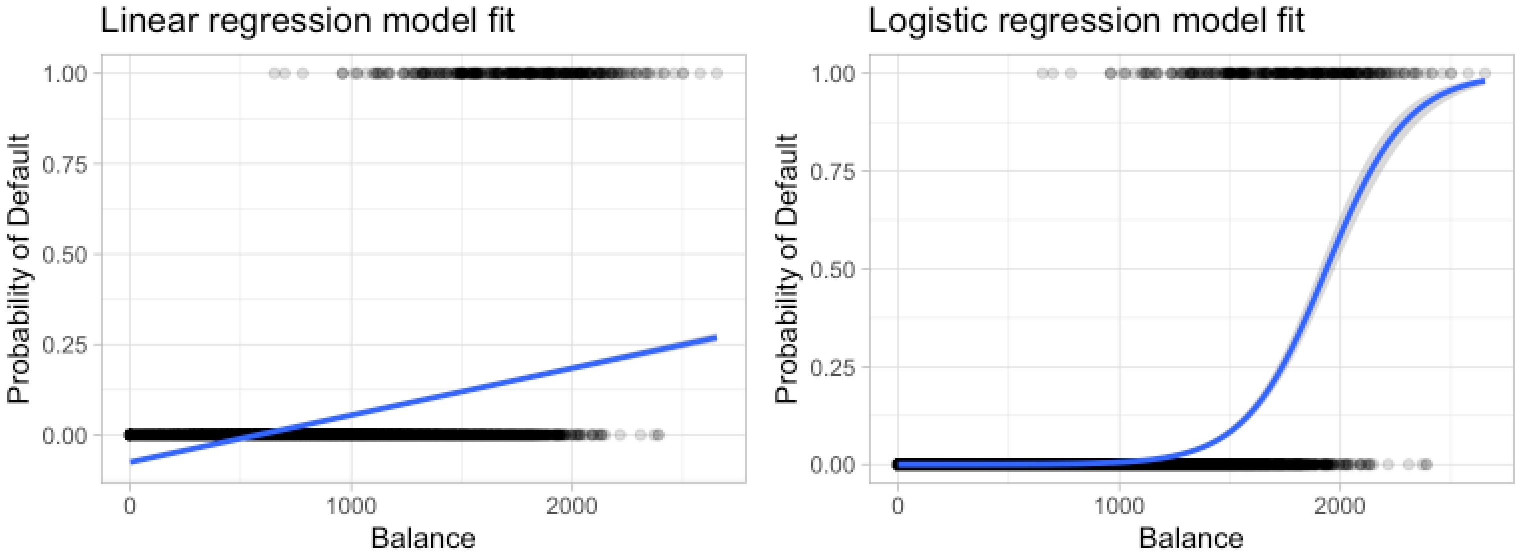




Logistic Regression

Logistic regression for predicting probabilities

- Linear models do poorly at predicting probabilities, but can be greatly improved with the use of a link function that constrains output to be between 0 (never happens) and 1 (always happens)



$$g(X) = \ln\left[\frac{p(X)}{1 - p(X)}\right] = \beta_0 + \beta_1 X$$

GPI

Titanic Dataset

- Predicting probability of survival
- library(titanic)
- library(tidyverse)


RMS Titanic departing Southampton on 10 April 1912

titanic_train (n=891)

- 5 character variables, may make factors
- “Cabin” missing in 687 cases
- 7 numeric (including target “Survived)
- Age missing in 177 cases

```
> skim(titanic_train)
-- Data Summary ------------------------
                           Values
Name                       titanic_train
Number of rows             891
Number of columns          12
_______________________
Column type frequency:
  character                5
  numeric                  7
________________________
Group variables            None

-- Variable type: character -----------------------------------------------------
# A tibble: 5 x 8
  skim_variable n_missing complete_rate   min   max empty n_unique whitespace
* <chr>             <int>         <dbl> <int> <int> <int>    <int>      <int>
1 Name                  0             1    12    82     0      891          0
2 Sex                   0             1     4     6     0        2          0
3 Ticket                0             1     3    18     0      681          0
4 Cabin                 0             1     0    15   687      148          0
5 Embarked              0             1     0     1     2        4          0

-- Variable type: numeric -------------------------------------------------------
# A tibble: 7 x 11
  skim_variable n_missing complete_rate    mean     sd    p0   p25   p50   p75
* <chr>             <int>         <dbl>   <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
1 PassengerId           0         1     446     257.    1    224.  446   668.
2 Survived              0         1       0.384   0.487 0      0     0     1
3 Pclass                0         1       2.31    0.836 1      2     3     3
4 Age                 177         0.801  29.7    14.5   0.42  20.1  28    38
5 SibSp                 0         1       0.523   1.10  0      0     0     1
6 Parch                 0         1       0.382   0.806 0      0     0     0
7 Fare                  0         1      32.2    49.7   0      7.91 14.5  31
   p100 hist
* <dbl> <chr>
1  891  ▇▇▇▇▇▇▇▇
2    1  ▇▁▁▁▁▁▁▅
3    3  ▃▁▁▃▁▁▁▇
4   80  ▁▅▇▅▂▁▁▁
5    8  ▇▁▁▁▁▁▁▁
6    6  ▇▁▁▁▁▁▁▁
7  512. ▇▁▁▁▁▁▁▁
>
```

titanic_test is similar (n=418)

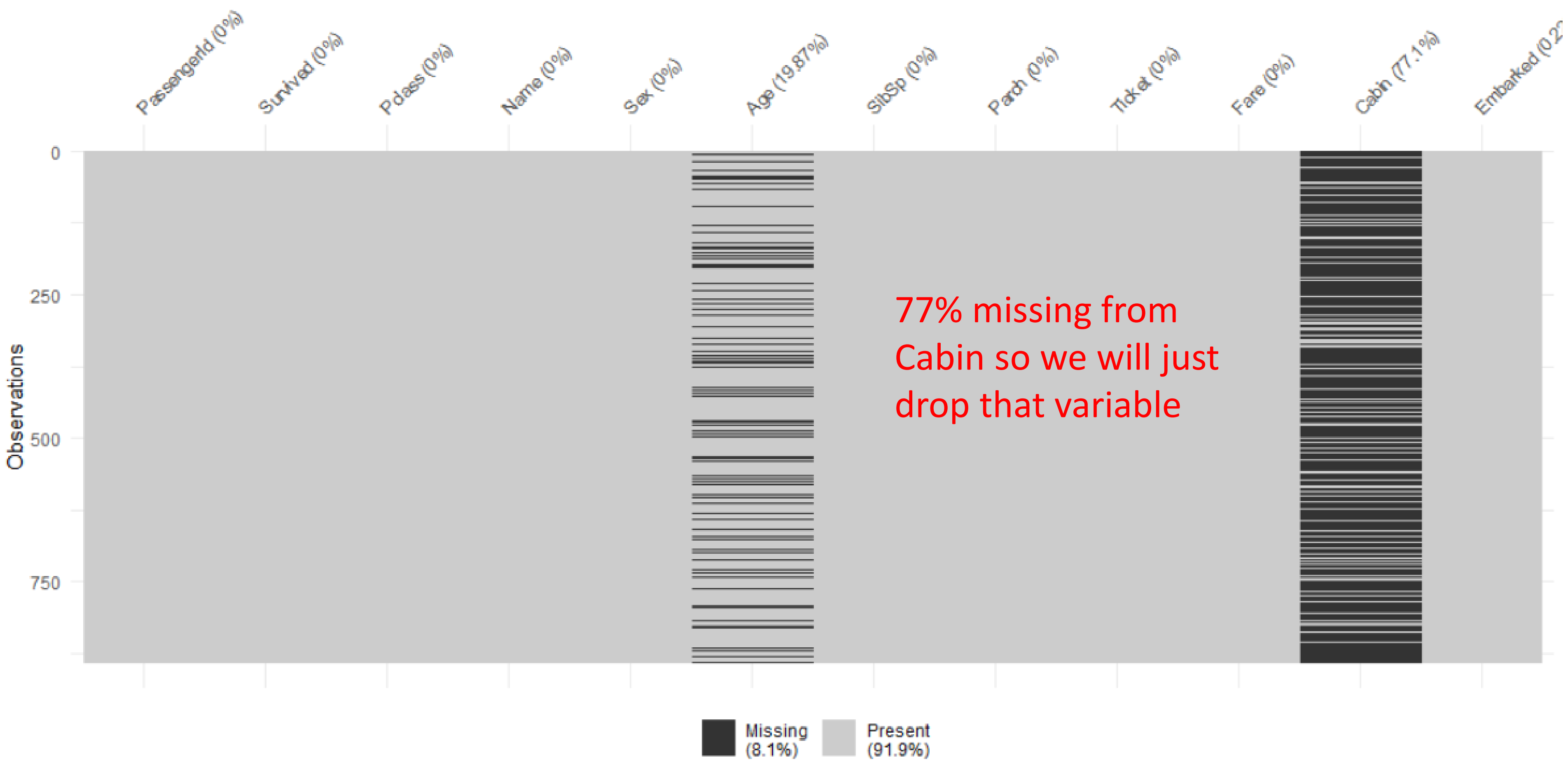
```
> skim(titanic_test)
-- Data Summary ------------------------
                           Values
Name                       titanic_test
Number of rows             418
Number of columns          11
_______________________
Column type frequency:
  character                5
  numeric                  6
________________________
Group variables            None

-- Variable type: character ------------------------------------------------------------------------------
# A tibble: 5 x 8
  skim_variable n_missing complete_rate   min   max empty n_unique whitespace
* <chr>             <int>         <dbl> <int> <int> <int>    <int>      <int>
1 Name                  0             1    13    63     0      418          0
2 Sex                   0             1     4     6     0        2          0
3 Ticket                0             1     3    18     0      363          0
4 Cabin                 0             1     0    15   327       77          0
5 Embarked              0             1     1     1     0        3          0

-- Variable type: numeric ----------------------------------------------------------------------------------
# A tibble: 6 x 11
  skim_variable n_missing complete_rate    mean     sd    p0    p25    p50    p75  p100 hist
* <chr>             <int>         <dbl>   <dbl>  <dbl> <dbl>  <dbl>  <dbl>  <dbl> <dbl> <chr>
1 PassengerId           0         1     1100.  121.    892    996.  1100.  1205.  1309  ▇▇▇▇▇
2 Pclass                0         1        2.27  0.842   1      1      3      3      3  ▃▁▃▁▇
3 Age                  86         0.794   30.3  14.2     0.17  21     27     39     76  ▂▇▃▂▁
4 SibSp                 0         1        0.447 0.897   0      0      0      1      8  ▇▁▁▁▁
5 Parch                 0         1        0.392 0.981   0      0      0      0      9  ▇▁▁▁▁
6 Fare                  1         0.998   35.6  55.9     0      7.90  14.5   31.5  512. ▇▁▁▁▁
> |
```

Visualizing missing values

```
library(visdat)
vis_miss(titanic_train)
```

Visualizing missing values

```
ttrain<-titanic_train
ttrain[ttrain==""] <- NA
vis_miss(train)
```



Data preparation

- Whatever we do to train we will eventually do to test, but avoid leakage of information (e.g., using mean of entire train+test to impute values, which lots of people do)!
- Dropping Cabin and PassengerID as not likely to help determine survival
  - ttrain2<- ttrain %>% dplyr::select(-Cabin, -PassengerId, -Ticket)
- Converting Survived, Sex, Pclass, and Embarked to factors from character

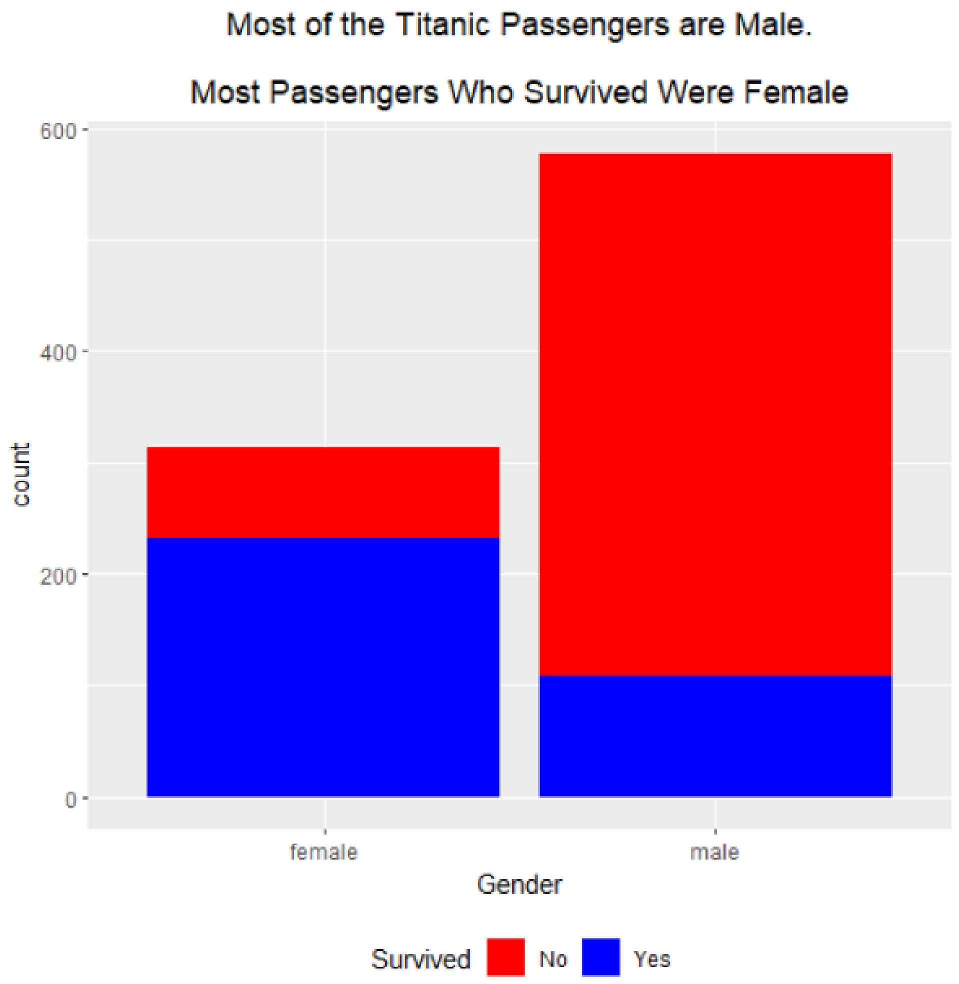
GPI

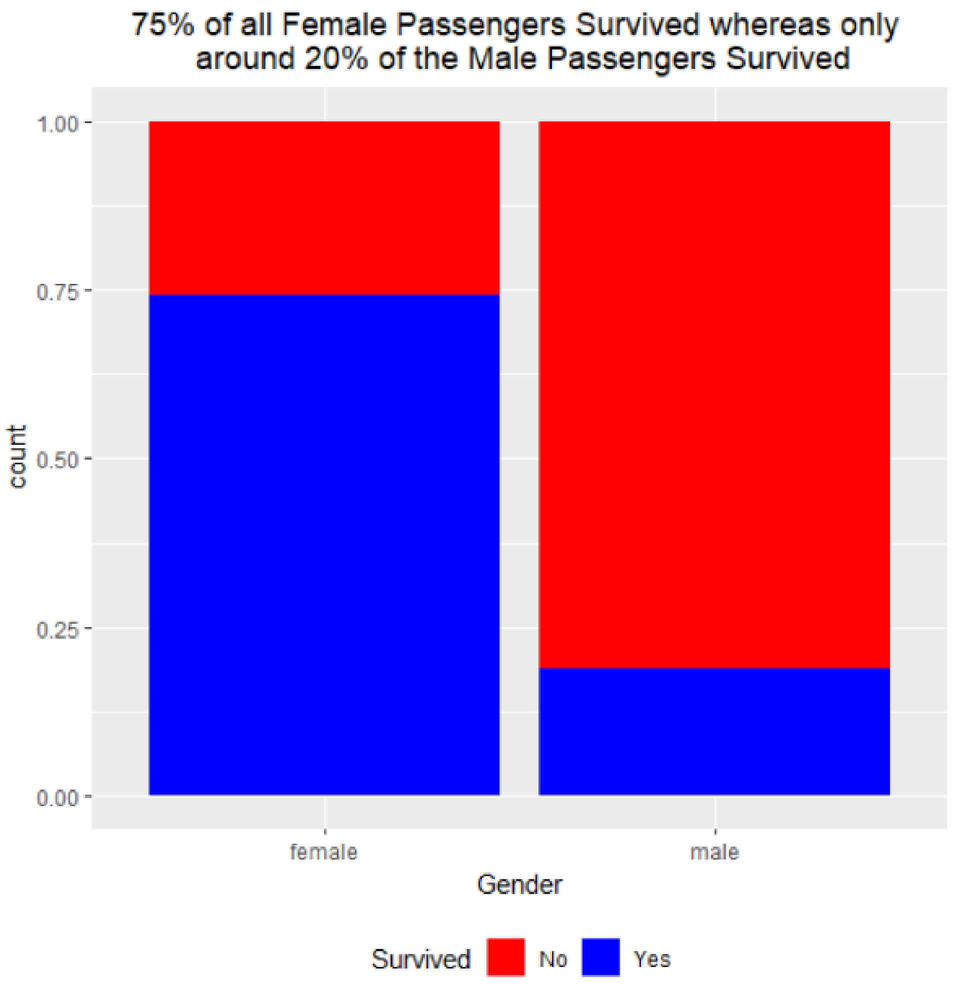
Characters to Factors

- # Character to Factor
  - ttrain2$Gender<-factor(ttrain2$Sex) # safely making new variable as factoring sometimes results in unexpected changes
- # Character to Ordered Factor
  - ttrain2$Pclass2 <- factor(ttrain2$Pclass, order=TRUE, levels = c(3, 2, 1))
- Dropping older versions of Survived, Pclass, and Sex
  - ttrain3<- ttrain2 %>% dplyr::select(-Survived, -Sex, -Pclass)

```
$ Gender  : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
$ Pclass2 : Ord.factor w/ 3 levels "3"<"2"<"1": 1 3 1 3 1 1 3 1 1 2 ...
$ Pclass3 : Factor w/ 3 levels "1","2","3": 3 1 3 1 3 3 1 3 3 2 ...
```

Nice exploratory plot methods available for this dataset online

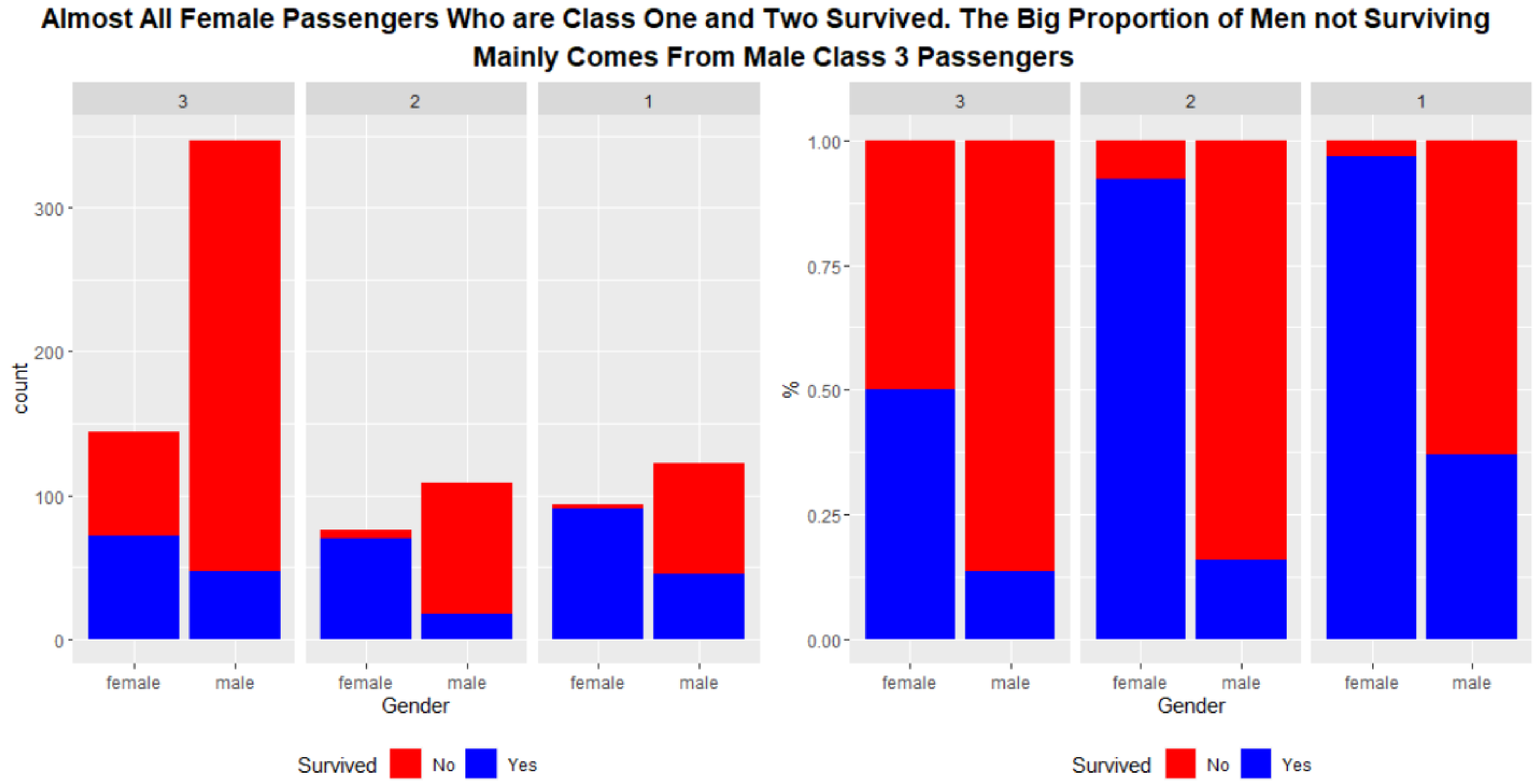




Adapted from: http://thatdatatho.com/2018/09/18/titanic-data-set-increased-prediction-scores-82/

GPI

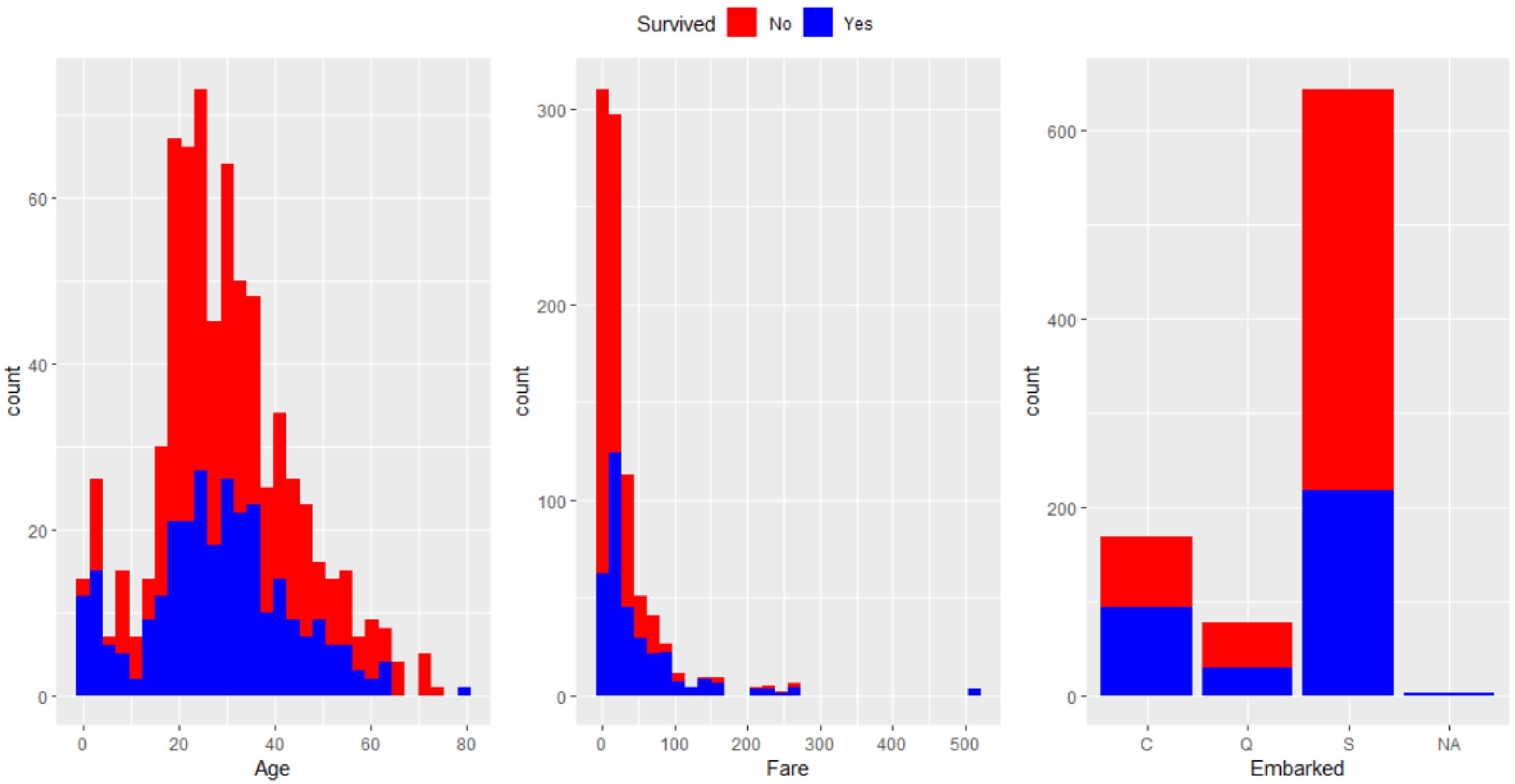
Higher class women had the best chance at surviving



Adapted from: http://thatdatatho.com/2018/09/18/titanic-data-set-increased-prediction-scores-82/

GPI

Very young children did o.k. and those who paid more for their tickets



Adapted from: http://thatdatatho.com/2018/09/18/titanic-data-set-increased-prediction-scores-82/

Imputing Age

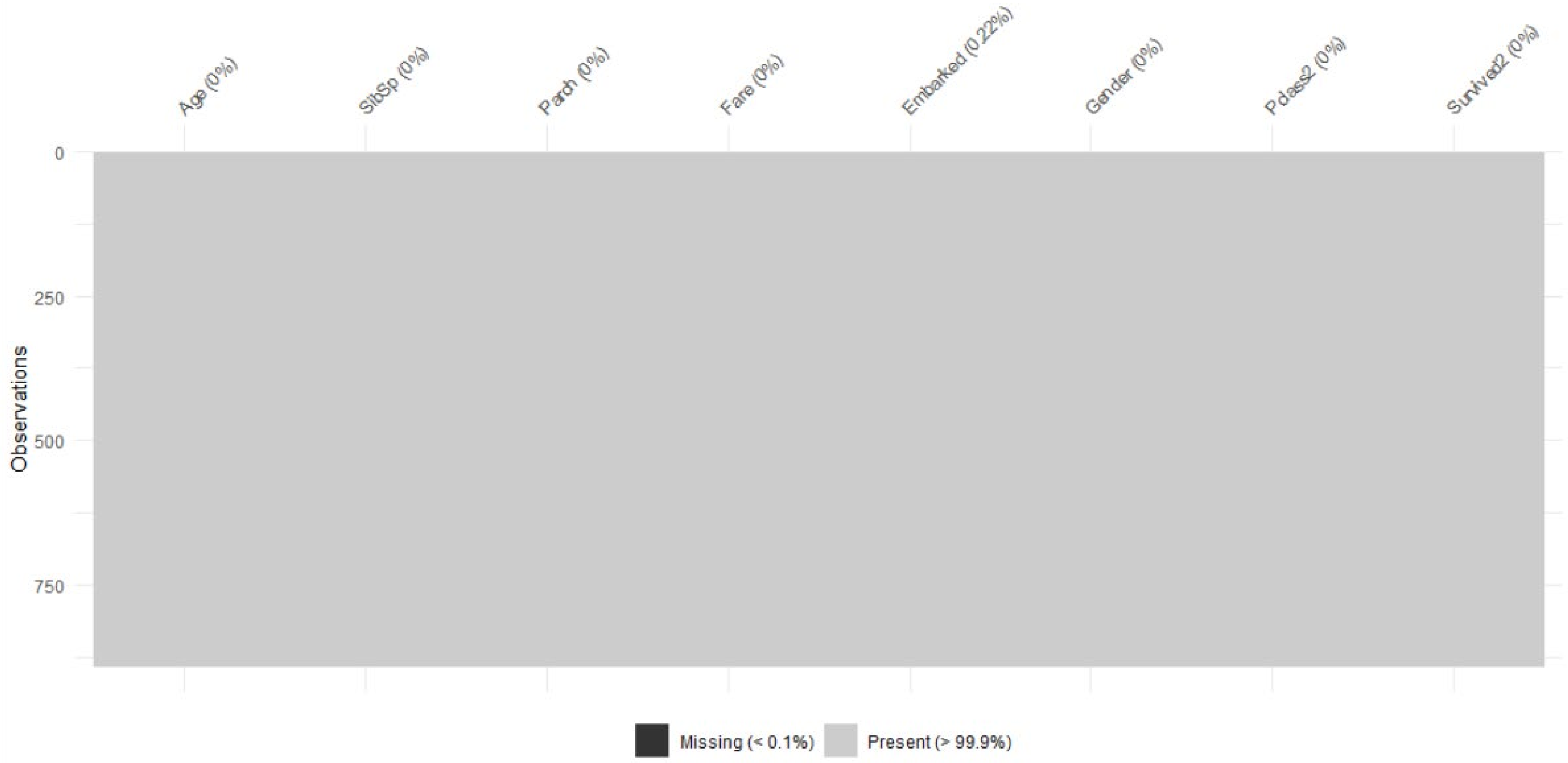
- About 20% of Age variable missing
- Some have investigating extracting titles from the Names—this is what is called “feature engineering” in the machine learning world and can provide a valuable boost to predictive ability.
- Keeping it simpler
  - Dropping Name
    - ttrain4<- ttrain3 %>% dplyr::select(-Name)
  - Imputing age as the median for the train dataset
    - median(ttrain3$Age, na.rm=T)
      - [1] 28
    - ttrain4$Age[is.na(ttrain4$Age)]<-28

Note if we used entire dataset for imputing median Age we risk leaking information from test set to the training set.

GPI

Visualizing missing values

```
vis_miss(ttrain4)
sum(is.na(ttrain4))
[1] 2 # something still missing!
```



Finishing filling missing values

- Embarked has 2 missing
- Filling with the most frequent category
- table(ttrain4$Embarked)
-     C   Q   S

  168 77 644
- Southampton port most popular, so filling missing with that
  - ttrain4$Embarked[is.na(ttrain4$Embarked)]<-"S"

```
-- Variable type: character ------------------------------------------
# A tibble: 1 x 8
  skim_variable n_missing complete_rate   min   max empty n_unique
* <chr>             <int>         <dbl> <int> <int> <int>    <int>
1 Embarked              2         0.998     1     1     0        3
  whitespace
*      <int>
1          0
```

Visualizing missing values

vis_miss(ttrain4) # done

Applying same steps to train dataset

- Realized that test data from Kaggle missing the Survived label!
- Tracked down the actual Survived:
  - http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic3.csv
- titanic3 <- read_csv("titanic3.csv")
- titanicjoinfile<- titanic3 %>% select(name, survived)
- ttest2b<-left_join(ttest2,titanicjoinfile,by=c('Name'='name'))

Additional processing of test

- ttest2b$Survived<-ttest2b$survived
- ttest2b$Survived2 <- factor(ttest2b$Survived)
- ttest3<- ttest2b %>% dplyr::select(-Survived, -survived, -Sex, -Pclass)
- ttest4<- ttest3 %>% dplyr::select(-Name)
- ttest4$Age[is.na(ttest4$Age)]<-28
- ttest4$Embarked[is.na(ttest4$Embarked)]<-"S"
- vis_miss(ttest4)

GPI

Visualizing missing values

We have small amount of missing Fare and Survived2
Dropping rows with any missing values
ttest4<- ttest4 %>% drop_na()

Visualizing missing values

We have small amount of missing Fare and Survived2

Dropping rows with any missing values

ttest4<- ttest4 %>% drop_na()

Confirming train and test ready for analysis

- # saving copies of the final processed files
  - write.csv(ttrain4,file='ttrain4.csv',row.names=F)
  - write.csv(ttest4,file='ttest4.csv',row.names=F)

```
> str(ttrain4)
'data.frame':   891 obs. of  8 variables:
 $ Age      : num  22 38 26 35 35 28 54 2 27 14 ...
 $ SibSp    : int  1 1 0 1 0 0 0 3 0 1 ...
 $ Parch    : int  0 0 0 0 0 0 0 1 2 0 ...
 $ Fare     : num  7.25 71.28 7.92 53.1 8.05 ...
 $ Embarked : chr  "S" "C" "S" "S" ...
 $ Gender   : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
 $ Pclass2  : Ord.factor w/ 3 levels "3"<"2"<"1": 1 3 1 3 1 1 3 1 1 2 ...
 $ Survived2: Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 1 2 2 ...
> str(ttest4)
'data.frame':   395 obs. of  8 variables:
 $ Age      : num  34.5 34.5 47 62 27 22 14 30 30 26 ...
 $ SibSp    : int  0 0 1 0 0 1 0 0 0 1 ...
 $ Parch    : int  0 0 0 0 0 1 0 0 0 1 ...
 $ Fare     : num  7.83 7.83 7 9.69 8.66 ...
 $ Embarked : chr  "Q" "Q" "S" "Q" ...
 $ Gender   : Factor w/ 2 levels "female","male": 2 2 1 2 2 1 2 1 1 2 ...
 $ Pclass2  : Ord.factor w/ 3 levels "3"<"2"<"1": 1 1 1 2 1 1 1 1 1 2 ...
 $ Survived2: Factor w/ 2 levels "0","1": 1 1 2 1 1 2 2 2 1 2 ...
```

GPI

Fitting logistic regression using all the variables

- logis1<-glm(Survived2 ~ ., family = binomial(link='logit'), data = ttrain4)

Summary(logis1)

```
> summary(logis1)

Call:
glm(formula = Survived2 ~ ., family = binomial(link = "logit"), 
    data = ttrain4)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-2.6199  -0.6089  -0.4176   0.6187   2.4514  

Coefficients:
             Estimate Std. Error z value Pr(>|z|)    
(Intercept)  3.040987   0.379697   8.009 1.16e-15 ***
Age         -0.038517   0.007855  -4.903 9.43e-07 ***
SibSp       -0.321794   0.109193  -2.947  0.00321 ** 
Parch       -0.093329   0.118856  -0.785  0.43232    
Fare         0.002339   0.002469   0.947  0.34346    
EmbarkedQ   -0.056267   0.381471  -0.148  0.88274    
EmbarkedS   -0.434226   0.239530  -1.813  0.06986 .  
Gendermale  -2.719444   0.200977 -13.531  < 2e-16 ***
Pclass2.L    1.520313   0.210520   7.222 5.13e-13 ***
Pclass2.Q   -0.127011   0.182384  -0.696  0.48618    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1186.66  on 890  degrees of freedom
Residual deviance:  785.04  on 881  degrees of freedom
AIC: 805.04

Number of Fisher Scoring iterations: 5
```

Evaluating performance on train

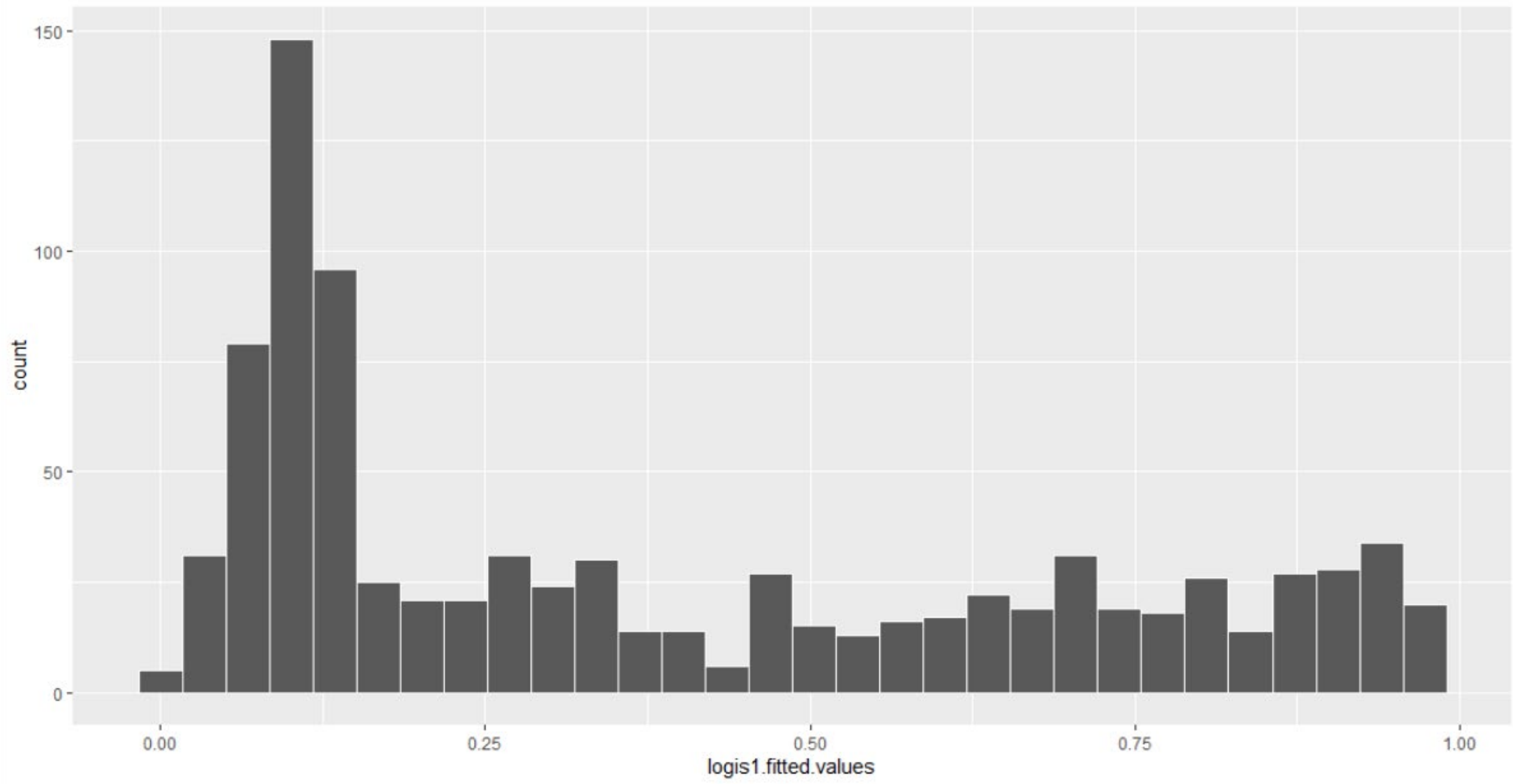
- str(logis1)
- Let's look at the fitted.values

```
> str(logis1)
List of 30
 $ coefficients      : Named num [1:10] 3.04099 -0.03852 -0.32179 -0.09333 0.
00234 ...
  ..- attr(*, "names")= chr [1:10] "(Intercept)" "Age" "SibSp" "Parch" ...
 $ residuals         : Named num [1:891] -1.09 1.09 1.61 1.12 -1.08 ...
  ..- attr(*, "names")= chr [1:891] "1" "2" "3" "4" ...
 $ fitted.values     : Named num [1:891] 0.0838 0.9202 0.6217 0.8894 0.0712
...
  ..- attr(*, "names")= chr [1:891] "1" "2" "3" "4" ...
 $ effects           : Named num [1:891] 4.541 1.368 -0.253 2.045 4.422 ...
  ..- attr(*, "names")= chr [1:891] "(Intercept)" "Age" "SibSp" "Parch" ...
 $ R                 : num [1:10, 1:10] -11.2 0 0 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:10] "(Intercept)" "Age" "SibSp" "Parch" ...
  .. ..$ : chr [1:10] "(Intercept)" "Age" "SibSp" "Parch" ...
 $ rank              : int 10
 $ qr                :List of 5
  ..$ qr   : num [1:891, 1:10] -11.1823 0.0242 0.0434 0.0281 0.023 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:891] "1" "2" "3" "4" ...
  .. .. ..$ : chr [1:10] "(Intercept)" "Age" "SibSp" "Parch" ...
  ..$ rank : int 10
  ..$ qraux: num [1:10] 1.02 1.02 1.03 1.02 1.01 ...
  ..$ pivot: int [1:10] 1 2 3 4 5 6 7 8 9 10
  ..$ tol  : num 1e-11
  ..- attr(*, "class")= chr "qr"
 $ family            :List of 12
  ..$ family    : chr "binomial"
  ..$ link      : chr "logit"
  ..$ linkfun   :function (mu)
```

GPI

Logistic regression output is a continuous probability, not survived: yes or no

- ggplot(data.frame(logis1$fitted.values),aes(x=logis1.fitted.values))+geom_histogram(col='white')



GPI

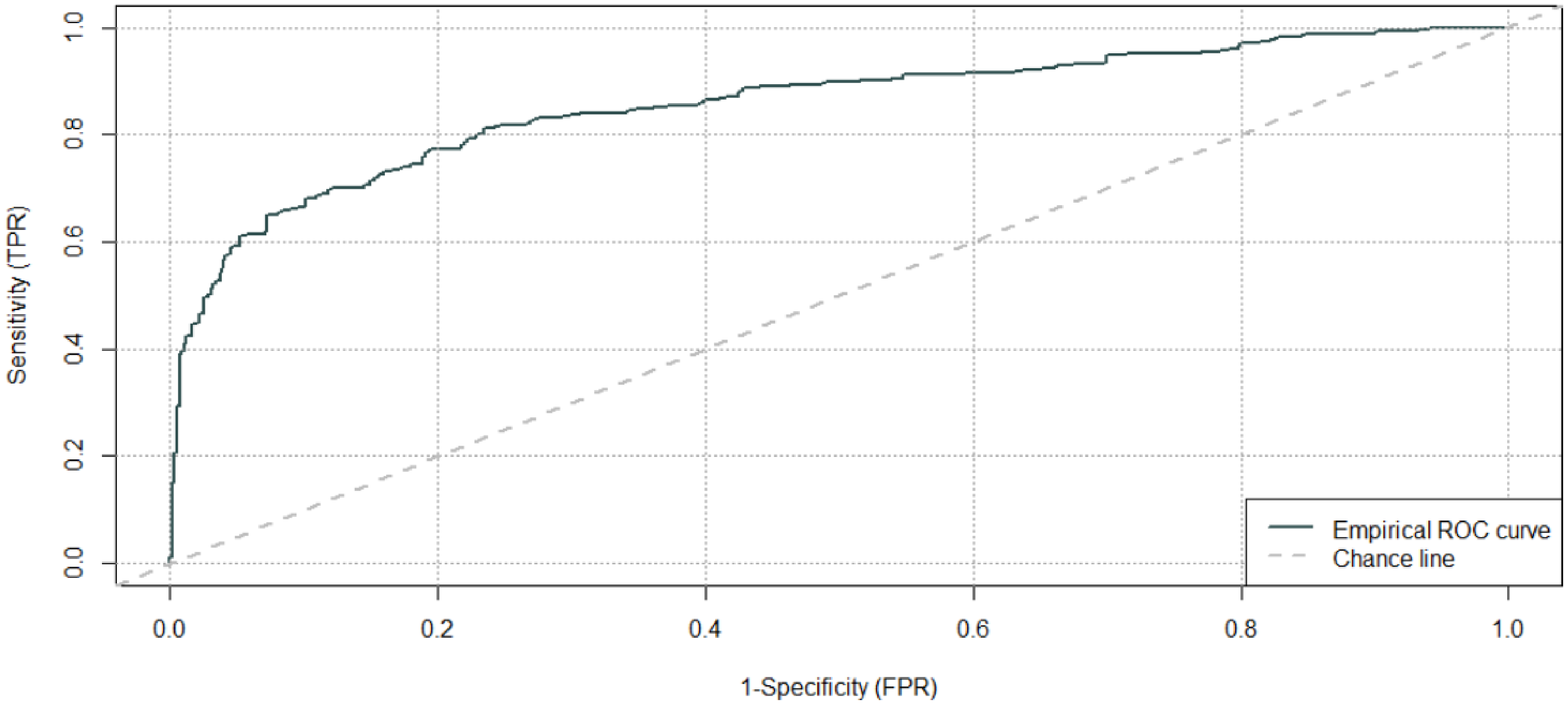
ROC curve evaluation

- AUC 85% represents a good model

```
> ROCit_obj <- rocit(score=logis1$fitted.values,class=ttrain4$Survived2)
> plot(ROCit_obj)
> plot(ROCit_obj, YIndex=F, values=T)
> ciAUC(ROCit_obj)

   estimated AUC : 0.856413574920909
   AUC estimation method : empirical

   CI of AUC
   confidence level = 95%
   lower = 0.829095548110175     upper = 0.883731601731642
> summary(ROCit_obj)

 Method used: empirical
 Number of positive(s): 342
 Number of negative(s): 549
 Area under curve: 0.8564
>
```

GPI

ROC curve for logis1



- Seeking to find cutoff that balances true positive and true negative rate

```
> predg1 <- prediction(logis1$fitted.values,ttrain4$Survived2)
> roc.perf1 = performance(predg1, measure = "tpr", x.measure = "fpr")
> plot(roc.perf1)
> abline(a=0,b=1)
> opt.cut = function(perf, pred){
+   cut.ind = mapply(FUN=function(x, y, p){
+     d = (x - 0)^2 + (y-1)^2
+     ind = which(d == min(d))
+     c(sensitivity = y[[ind]], specificity = 1-x[[ind]],
+       cutoff = p[[ind]])
+   }, perf@x.values, perf@y.values, pred@cutoffs)
+ }
> print(opt.cut(roc.perf1, predg1))
                 [,1]
sensitivity 0.7719298
specificity 0.8051002
cutoff      0.3785297
> 
```

Misclassification or Confusion Matrix

	Death (0)	Survived (1)	Totals
Predicted Death (0)	a	b	a+b
Predicted Survived (1)	c	d	c+d
Totals	a+c	b+d	

a, b, c, d are counts of individuals

- Sensitivity (a/a+c): Probability that a person who died will be modeled as dying.
- Specificity (d/b+d): Probability that a person who survived will be modeled as surviving.
- Accuracy = (a+d /a+b+c+d): Overall probability of a correct outcome.
- Positive Predictive Value (a/a+b): Probability that a model prediction of death is accurate.
- Negative Predictive Value (d/c+d): Probability that a model prediction of survival is accurate.

Tech tip: PPV and NPV change with predicted prevalence of stress types in the population

GPI

confusionMatrix for 0.5 cutoff train

- Accuracy = 80%
- Guessing not survived would achieve 62% accuracy

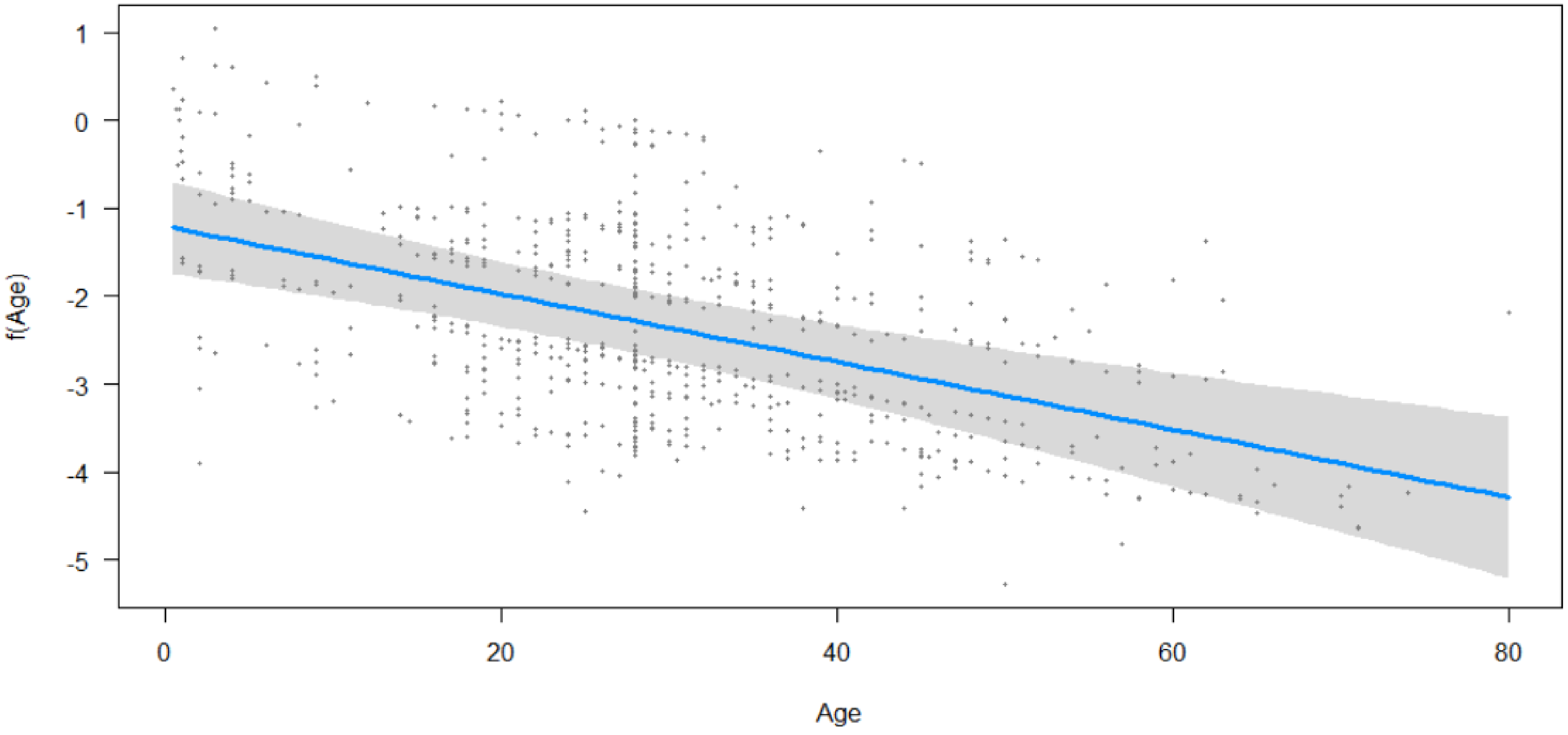
```
> confusionMatrix(factor(pred_ttrain4_df2$cutpt5),pred_ttrain4_df2$Survived2)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 477 102
         1  72 240

               Accuracy : 0.8047
                 95% CI : (0.7771, 0.8303)
    No Information Rate : 0.6162
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.5802

 Mcnemar's Test P-Value : 0.02791

            Sensitivity : 0.8689
            Specificity : 0.7018
         Pos Pred Value : 0.8238
         Neg Pred Value : 0.7692
             Prevalence : 0.6162
         Detection Rate : 0.5354
   Detection Prevalence : 0.6498
      Balanced Accuracy : 0.7853

       'Positive' Class : 0
```

confusionMatrix for 0.38 cutoff train

- Accuracy = 79%

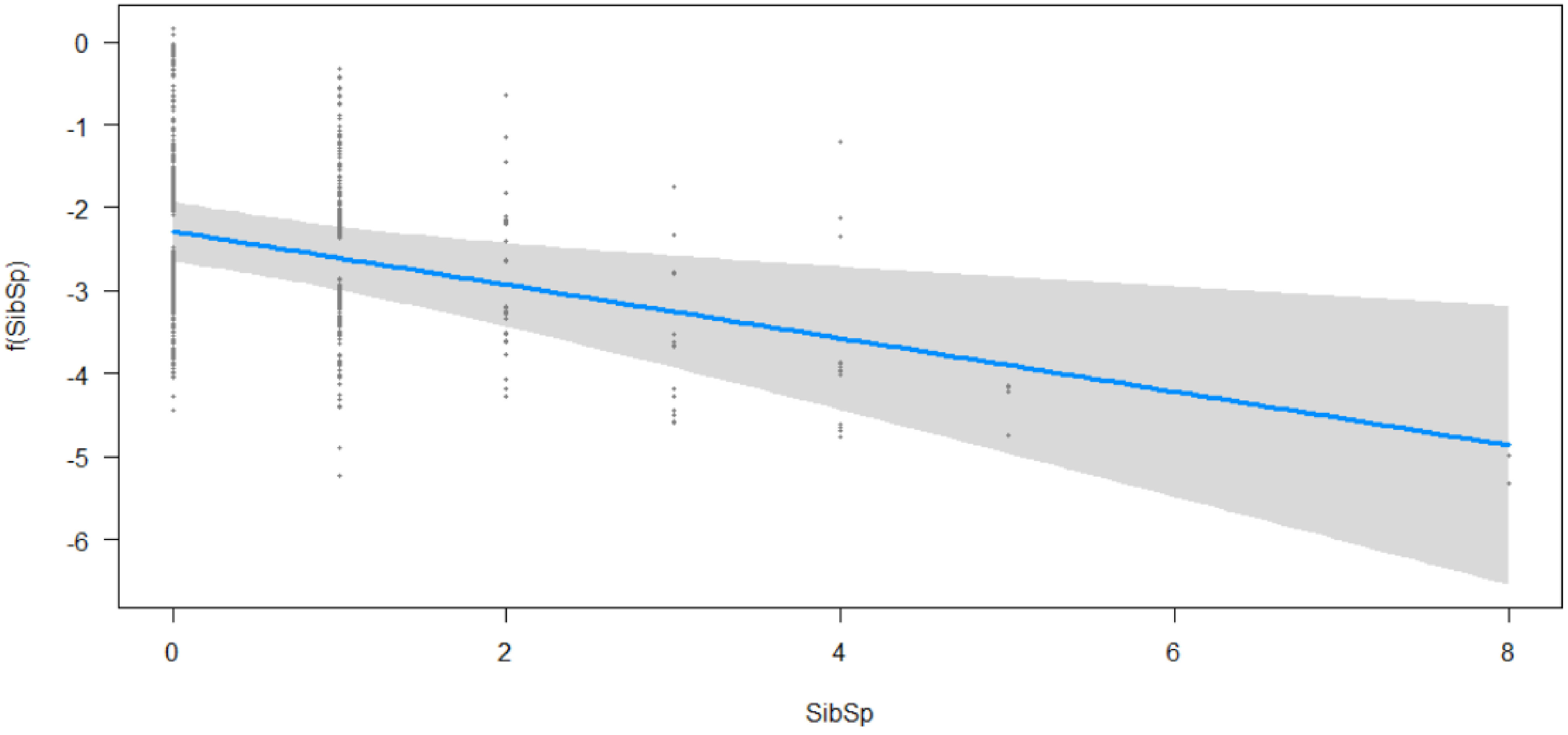
```
> confusionMatrix(factor(pred_ttrain4_df2$cutpt38),pred_ttrain4_df2$Survived2)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 442  79
         1 107 263

               Accuracy : 0.7912
                 95% CI : (0.7631, 0.8175)
    No Information Rate : 0.6162
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.5654

 Mcnemar's Test P-Value : 0.04773

            Sensitivity : 0.8051
            Specificity : 0.7690
         Pos Pred Value : 0.8484
         Neg Pred Value : 0.7108
             Prevalence : 0.6162
         Detection Rate : 0.4961
   Detection Prevalence : 0.5847
      Balanced Accuracy : 0.7871

       'Positive' Class : 0
```

Visreg(logis1)

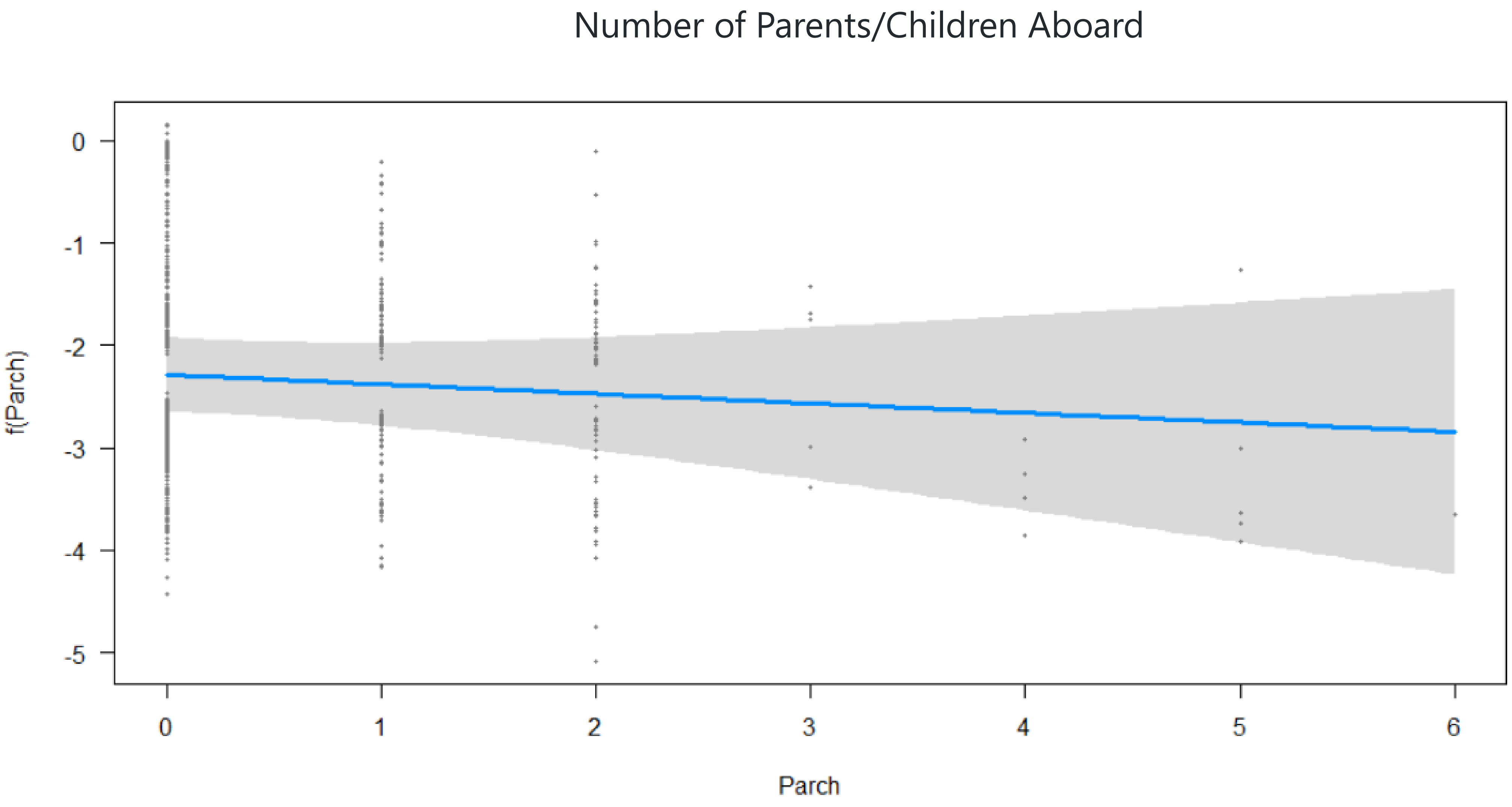


Visreg(logis1)

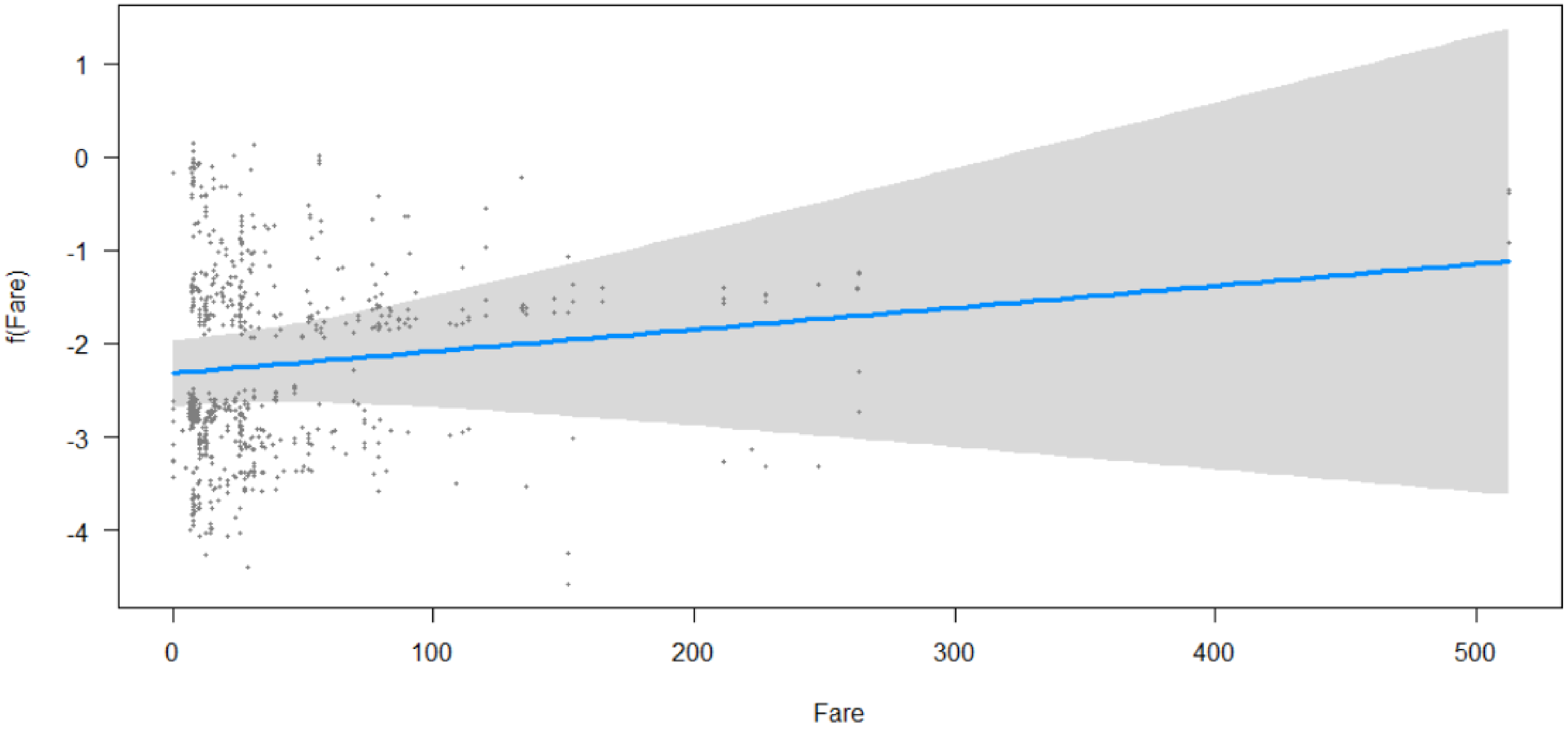
Number of Siblings/Spouses Aboard



Visreg(logis1)

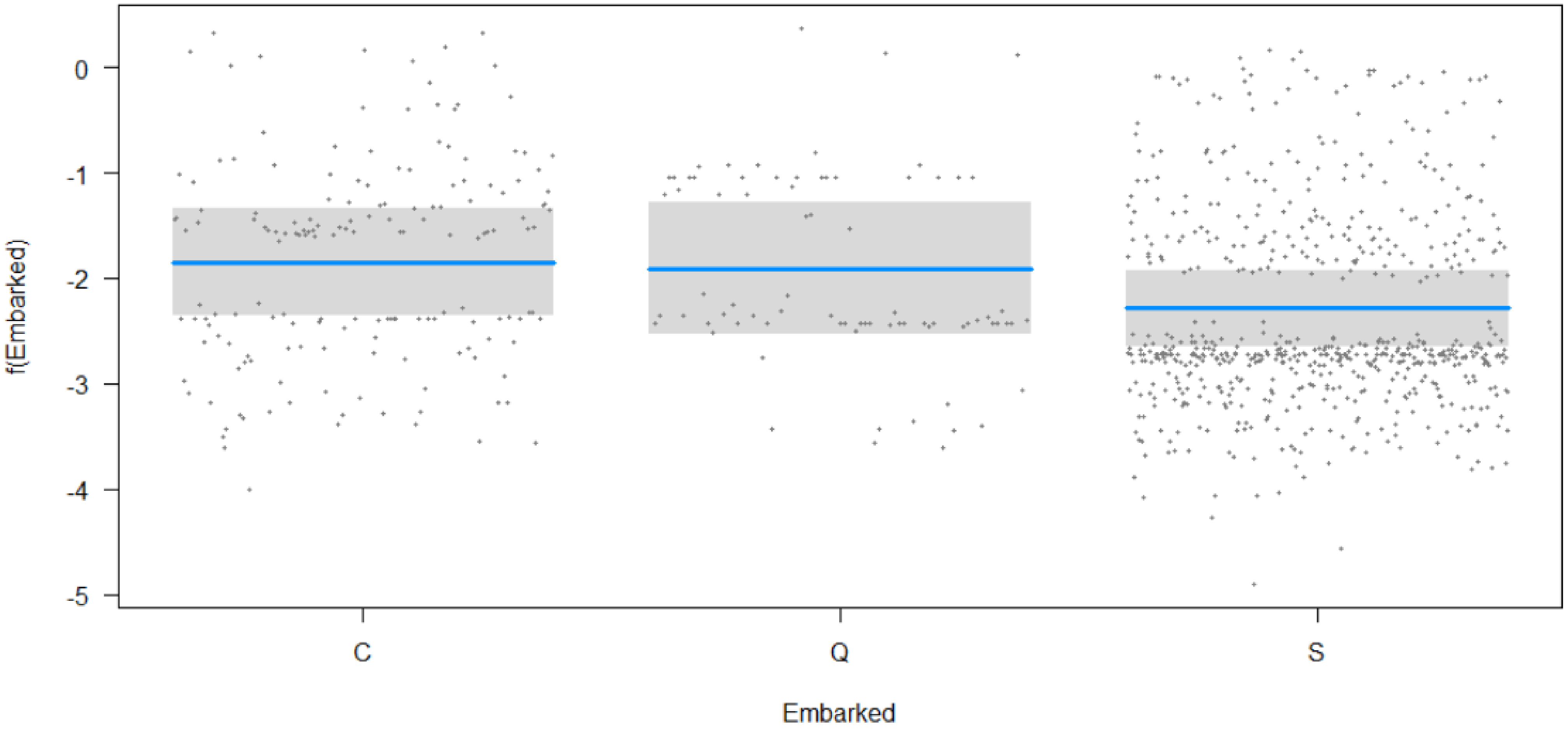


Visreg(logis1)

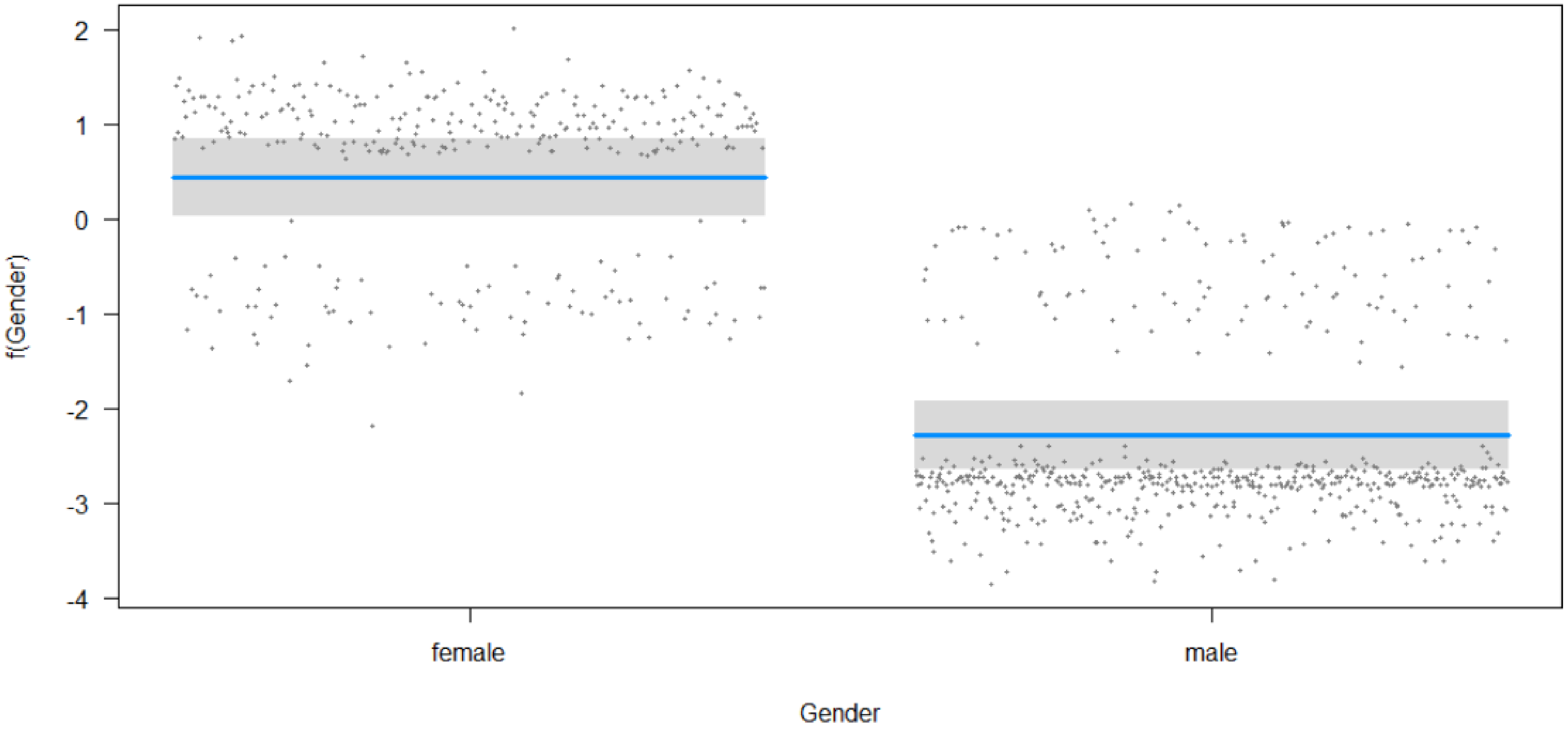


Visreg(logis1)

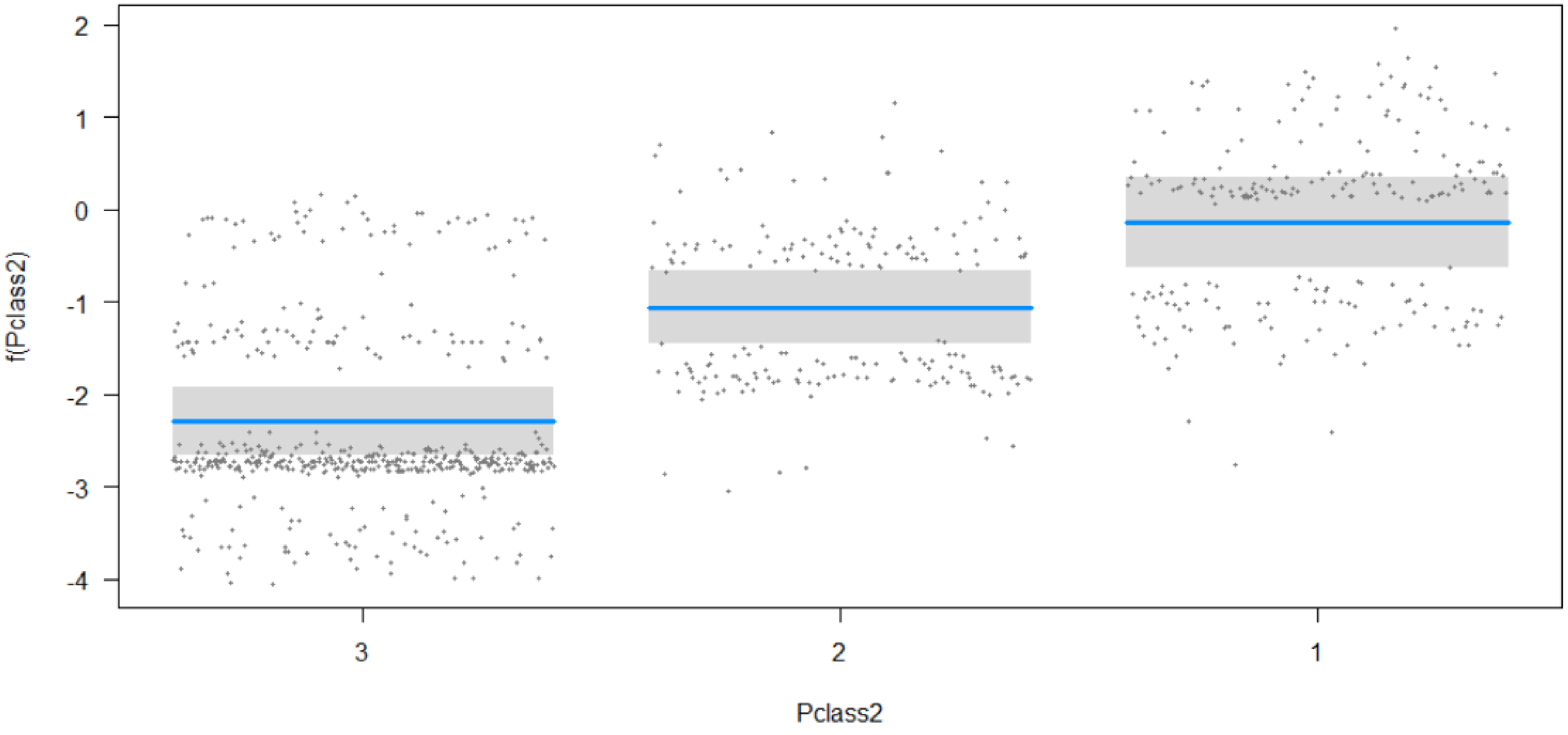
Port of Embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)



Visreg(logis1)



Visreg(logis1)



confusionMatrix for 0.5 cuttoff test

```
> confusionMatrix(factor(pred_ttest4_df2$cutpt5),pred_ttest4_df2$Survived2)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 205  48
         1  45  97

               Accuracy : 0.7646
                 95% CI : (0.7196, 0.8055)
    No Information Rate : 0.6329
    P-Value [Acc > NIR] : 1.408e-08

                  Kappa : 0.4911

 Mcnemar's Test P-Value : 0.8357

            Sensitivity : 0.8200
            Specificity : 0.6690
         Pos Pred Value : 0.8103
         Neg Pred Value : 0.6831
             Prevalence : 0.6329
         Detection Rate : 0.5190
   Detection Prevalence : 0.6405
      Balanced Accuracy : 0.7445

       'Positive' Class : 0
```

GPI

confusionMatrix for 0.38 cutoff test

```
> confusionMatrix(factor(pred_ttest4_df2$cutpt38),pred_ttest4_df2$Survived2)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 187  40
         1  63 105

               Accuracy : 0.7392
                 95% CI : (0.693, 0.7819)
    No Information Rate : 0.6329
    P-Value [Acc > NIR] : 4.546e-06

                  Kappa : 0.4569

 Mcnemar's Test P-Value : 0.03018

            Sensitivity : 0.7480
            Specificity : 0.7241
         Pos Pred Value : 0.8238
         Neg Pred Value : 0.6250
             Prevalence : 0.6329
         Detection Rate : 0.4734
   Detection Prevalence : 0.5747
      Balanced Accuracy : 0.7361

       'Positive' Class : 0
```

GPI