COMP 4442 Final Exam, Spring 2024

Michael Ghattas

```
knitr::opts_chunk$set(echo = TRUE)

library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-8
```

```
library(MASS)
library(survival)
library(survminer)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: ggpubr
```

```
##
## Attaching package: 'survminer'
```

```
## The following object is masked from 'package:survival':
##
##     myeloma
```

```
library(car)
```

```
## Loading required package: carData
```

There are four questions on this final, all of which have multiple parts. Please be sure to provide answers to all parts of each question. Each question has an associated .csv file, which you will load into memory at the beginning of the question. All of the included data sets are simulated, so any results should not be taken as evidence for or against the existence of anything in the real world. The data were simulated to minimize the ambiguity and messiness that typifies real data. If you feel that something is ambiguous in a way that impedes your ability to answer the questions, please let me know.

Almost there - you can do it!

Question 1 - Train/validate/test (30 points total)

For this question, a 400 observation data set was split such that 50% of the data was assigned to a training set, 25% was assigned to a validation set, and the remaining 25% was assigned to a test set. The outcome variable is y, which is contained in all three data sets, and is continuous. The predictor variables, also contained in all three data sets, are all continuous.

Run the code chunk below to load the data into memory before beginning your work on this question.

```
# Load data
q1.train <- read.csv("q1_train.csv", header=TRUE, sep=",")
q1.valid <- read.csv("q1_valid.csv", header=TRUE, sep=",")
q1.test <- read.csv("q1_test.csv", header=TRUE, sep=",")

# Check structures
str(q1.train)
```

```
## 'data.frame':    200 obs. of  11 variables:
##  $ X : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ y : num  -4.59 11.39 8.94 -3.16 -6.79 ...
##  $ x1: num  0.734 -1.121 0.804 0.455 0.19 ...
##  $ x2: num  -0.436 0.715 -0.55 0.442 -0.539 ...
##  $ x3: num  -0.4641 -1.3285 0.0387 2.3444 -1.5627 ...
##  $ x4: num  0.953 0.763 0.832 -1.153 0.272 ...
##  $ x5: num  1.3568 -0.0246 -1.0233 1.815 -1.9222 ...
##  $ x6: num  0.386 -0.514 0.867 -0.974 -0.57 ...
##  $ x7: num  -2.524 1.779 0.183 -0.884 -0.422 ...
##  $ x8: num  1.2597 0.8249 0.6353 -0.7715 0.0262 ...
##  $ x9: num  0.853 -0.5235 0.224 0.0744 1.169 ...
```

```
str(q1.valid)
```

```
## 'data.frame':    100 obs. of  11 variables:
##  $ X : int  201 202 203 204 205 206 207 208 209 210 ...
##  $ y : num  -5.27 1.03 5.71 5.96 5.87 ...
##  $ x1: num  0.484 -0.154 1.398 0.134 -0.498 ...
##  $ x2: num  -0.7664 1.1196 -0.2013 -0.6587 0.0931 ...
##  $ x3: num  0.659 0.431 -0.112 0.583 0.595 ...
##  $ x4: num  0.211 -1.014 0.176 -0.635 -0.34 ...
##  $ x5: num  1.456 -1.187 -0.808 0.597 0.538 ...
##  $ x6: num  0.774 -2.073 -1.691 0.204 1.343 ...
##  $ x7: num  -1.2134 -0.7085 -0.0487 0.8162 0.3041 ...
##  $ x8: num  -0.379 -0.68 0.175 -0.512 -0.374 ...
##  $ x9: num  0.498 0.638 0.566 -1.977 -0.487 ...
```

```
str(q1.test)
```

```
## 'data.frame':    100 obs. of  11 variables:
##  $ X : int  301 302 303 304 305 306 307 308 309 310 ...
##  $ y : num  -1.5 10.92 7.95 21.58 14.16 ...
##  $ x1: num  0.2025 -0.3654 0.2806 -0.026 0.0843 ...
##  $ x2: num  0.644 -0.674 -0.443 0.416 0.184 ...
```

```
##  $ x3: num  -1.035 -1.253 -0.27 0.251 0.248 ...
##  $ x4: num  0.4752 -1.4235 -0.0929 -0.413 -0.329 ...
##  $ x5: num  -0.163 1.855 1.497 -0.149 -0.317 ...
##  $ x6: num  1.5316 0.6652 1.1081 -0.5957 0.0307 ...
##  $ x7: num  -1.45 1.726 -0.157 1.232 0.31 ...
##  $ x8: num  -1.053 0.825 -1.296 -2.1 0.781 ...
##  $ x9: num  -2.227 0.928 -0.21 0.444 -0.328 ...
```

Q1, Part 1 - Generating candidate models on the training set (15 points)

You will generate four candidate models: a model obtained through forward selection, a model obtained through backward selection, a cross-validated ridge regression model, and a cross-validated lasso regression model. For both cross-validated models, use the model associated with lambda.1se. You will fit all of these models to the training data (q1.train), using y as the outcome and x1-x9 as the predictors. Please do not include interactions or squares in your pool of potential predictors. Once you have done this, answer the four questions below.

Forward selection:

```
# Set up for forward selection
model.null <- lm(y ~ 1, data = q1.train)
model.full <- lm(y ~ ., data = q1.train)
model.forward <- step(model.null, scope = formula(model.full), direction = "forward", trace = 1)
```

```
## Start:  AIC=746.58
## y ~ 1
##
##        Df Sum of Sq    RSS    AIC
## + x7    1   2194.78 6082.2 686.96
## + x2    1    767.58 7509.4 729.12
## + x9    1    505.60 7771.4 735.98
## + x8    1    453.85 7823.1 737.30
## + x6    1    354.83 7922.1 739.82
## + x1    1    230.70 8046.2 742.93
## + x3    1    109.30 8167.6 745.92
## <none>              8276.9 746.58
## + x5    1     61.31 8215.6 747.10
## + x4    1     12.80 8264.1 748.27
## + X     1      9.64 8267.3 748.35
##
## Step:  AIC=686.96
## y ~ x7
##
##        Df Sum of Sq    RSS    AIC
## + x2    1    720.55 5361.6 663.74
## + x6    1    355.92 5726.2 676.90
## + x1    1    178.87 5903.3 682.99
## + x3    1    123.97 5958.2 684.84
## + x5    1    119.88 5962.3 684.98
## <none>              6082.2 686.96
## + x8    1     56.98 6025.2 687.08
```

```
## + x9    1     43.03 6039.1 687.54
## + X     1     27.96 6054.2 688.04
## + x4    1      0.02 6082.1 688.96
##
## Step:  AIC=663.74
## y ~ x7 + x2
##
##        Df Sum of Sq    RSS    AIC
## + x6    1   239.607 5122.0 656.60
## + x8    1   110.530 5251.1 661.57
## + x9    1    90.275 5271.3 662.34
## + x5    1    79.371 5282.2 662.76
## <none>              5361.6 663.74
## + X     1    26.723 5334.9 664.74
## + x1    1     5.145 5356.5 665.55
## + x4    1     1.646 5360.0 665.68
## + x3    1     0.045 5361.6 665.74
##
## Step:  AIC=656.6
## y ~ x7 + x2 + x6
##
##        Df Sum of Sq    RSS    AIC
## + x9    1   105.466 5016.5 654.44
## + x8    1    83.531 5038.5 655.31
## <none>              5122.0 656.60
## + x5    1    33.507 5088.5 657.28
## + X     1    25.850 5096.2 657.58
## + x1    1     5.456 5116.6 658.38
## + x4    1     5.089 5116.9 658.40
## + x3    1     1.228 5120.8 658.55
##
## Step:  AIC=654.44
## y ~ x7 + x2 + x6 + x9
##
##        Df Sum of Sq    RSS    AIC
## <none>              5016.5 654.44
## + x8    1    44.607 4971.9 654.65
## + x5    1    34.381 4982.2 655.06
## + X     1    30.606 4985.9 655.21
## + x4    1    10.724 5005.8 656.01
## + x1    1     6.442 5010.1 656.18
## + x3    1     1.598 5014.9 656.37
```

```
summary(model.forward)
```

```
##
## Call:
## lm(formula = y ~ x7 + x2 + x6 + x9, data = q1.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.7632  -3.1881   0.5546   3.2557  16.1706
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.3638     0.3637  11.998  < 2e-16 ***
## x7            2.9305     0.3735   7.846 2.76e-13 ***
## x2            2.0546     0.4089   5.025 1.13e-06 ***
## x6            1.1549     0.3670   3.147  0.00191 **
## x9            0.8273     0.4086   2.025  0.04426 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.072 on 195 degrees of freedom
## Multiple R-squared:  0.3939, Adjusted R-squared:  0.3815
## F-statistic: 31.68 on 4 and 195 DF,  p-value: < 2.2e-16
```

Backward selection:

```
# Set up for backward selection
model.backward <- step(model.full, direction = "backward", trace = 1)
```

```
## Start:  AIC=661.68
## y ~ X + x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
##
##        Df Sum of Sq    RSS    AIC
## - x1    1      0.94 4899.6 659.72
## - x3    1      5.41 4904.0 659.90
## - x4    1     15.85 4914.5 660.33
## - X     1     18.39 4917.0 660.43
## - x5    1     31.57 4930.2 660.96
## - x8    1     35.91 4934.5 661.14
## <none>              4898.6 661.68
## - x9    1     79.16 4977.8 662.89
## - x6    1    199.03 5097.7 667.64
## - x2    1    471.15 5369.8 678.04
## - x7    1   1371.37 6270.0 709.04
##
## Step:  AIC=659.72
## y ~ X + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9
##
##        Df Sum of Sq    RSS    AIC
## - x3    1      5.62 4905.2 657.95
## - x4    1     16.83 4916.4 658.40
## - X     1     19.65 4919.2 658.52
## - x5    1     30.97 4930.5 658.98
## - x8    1     38.47 4938.0 659.28
## <none>              4899.6 659.72
## - x9    1     78.56 4978.1 660.90
## - x6    1    199.15 5098.7 665.69
## - x2    1    611.28 5510.8 681.23
## - x7    1   1373.06 6272.6 707.13
##
## Step:  AIC=657.95
## y ~ X + x2 + x4 + x5 + x6 + x7 + x8 + x9
##
##        Df Sum of Sq    RSS    AIC
```

```
## - x4    1     17.50 4922.7 656.66
## - X     1     19.64 4924.8 656.75
## - x5    1     26.90 4932.1 657.04
## - x8    1     38.74 4943.9 657.52
## <none>              4905.2 657.95
## - x9    1     77.92 4983.1 659.10
## - x6    1    196.80 5102.0 663.81
## - x2    1    668.75 5573.9 681.51
## - x7    1   1374.79 6280.0 705.36
##
## Step:  AIC=656.66
## y ~ X + x2 + x5 + x6 + x7 + x8 + x9
##
##        Df Sum of Sq    RSS    AIC
## - X     1     19.14 4941.8 655.43
## - x5    1     26.90 4949.6 655.75
## - x8    1     33.26 4955.9 656.01
## <none>              4922.7 656.66
## - x9    1     73.57 4996.3 657.63
## - x6    1    191.11 5113.8 662.28
## - x2    1    657.40 5580.1 679.73
## - x7    1   1441.35 6364.0 706.02
##
## Step:  AIC=655.43
## y ~ x2 + x5 + x6 + x7 + x8 + x9
##
##        Df Sum of Sq    RSS    AIC
## - x5    1     30.11 4971.9 654.65
## - x8    1     40.34 4982.2 655.06
## <none>              4941.8 655.43
## - x9    1     68.51 5010.3 656.19
## - x6    1    188.39 5130.2 660.92
## - x2    1    659.23 5601.0 678.48
## - x7    1   1428.37 6370.2 704.21
##
## Step:  AIC=654.65
## y ~ x2 + x6 + x7 + x8 + x9
##
##        Df Sum of Sq    RSS    AIC
## - x8    1     44.61 5016.5 654.44
## <none>              4971.9 654.65
## - x9    1     66.54 5038.5 655.31
## - x6    1    229.59 5201.5 661.68
## - x2    1    676.68 5648.6 678.17
## - x7    1   1405.86 6377.8 702.45
##
## Step:  AIC=654.44
## y ~ x2 + x6 + x7 + x9
##
##        Df Sum of Sq    RSS    AIC
## <none>              5016.5 654.44
## - x9    1    105.47 5122.0 656.60
## - x6    1    254.80 5271.3 662.34
## - x2    1    649.66 5666.2 676.79
```

```
## - x7    1   1583.84 6600.4 707.31
```

```
summary(model.backward)
```

```
##
## Call:
## lm(formula = y ~ x2 + x6 + x7 + x9, data = q1.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.7632  -3.1881   0.5546   3.2557  16.1706
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.3638     0.3637  11.998  < 2e-16 ***
## x2            2.0546     0.4089   5.025 1.13e-06 ***
## x6            1.1549     0.3670   3.147  0.00191 **
## x7            2.9305     0.3735   7.846 2.76e-13 ***
## x9            0.8273     0.4086   2.025  0.04426 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.072 on 195 degrees of freedom
## Multiple R-squared:  0.3939, Adjusted R-squared:  0.3815
## F-statistic: 31.68 on 4 and 195 DF,  p-value: < 2.2e-16
```

Cross-validated ridge regression (lambda.1se); please use the provided random seed

```
# Ridge regression
x.train <- model.matrix(y ~ ., data = q1.train)[, -1]
y.train <- q1.train$y

set.seed(202211)
cv.ridge <- cv.glmnet(x.train, y.train, alpha = 0)
lambda.ridge <- cv.ridge$lambda.1se
coef(cv.ridge, s = lambda.ridge)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  4.125391346
## X            0.002312099
## x1           0.277432804
## x2           0.929027394
## x3           0.130129959
## x4          -0.162272536
## x5           0.207005189
## x6           0.552815084
## x7           1.374023463
## x8           0.459935686
## x9           0.545744357
```

Cross-validated lasso regression (lambda.1se); please use the provided random seed

```r
# Lasso regression
set.seed(202211)
cv.lasso <- cv.glmnet(x.train, y.train, alpha = 1)
lambda.lasso <- cv.lasso$lambda.1se
coef(cv.lasso, s = lambda.lasso)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept) 4.4004154
## X           .
## x1          .
## x2          1.0097719
## x3          .
## x4          .
## x5          .
## x6          0.2302298
## x7          2.2487866
## x8          .
## x9          .
```

A) What predictors are included in the model selected by forward selection?

Your answer here: x7, x2, x6, x9

B) What predictors are included in the model selected by backward selection?

Your answer here: x2, x6, x7, x9

C) What predictors are included in the cross-validated ridge model (lambda.1se)?

Your answer here: X, x1, x2, x3, x4, x5, x6, x7, x8, x9 (i.e. All)

D) What predictors are included in the cross-validated lasso model (lambda.1se)?

Your answer here: x2, x6, x7

Q1, Part 2 - Fitting candidate models on the validation set and choosing a model (10 points)

Now, use the validation data (q1.valid) to generate new predictions for the four candidate models obtained in the previous part. Use these to compute the mean squared prediction error to evaluate how well each candidate model (which was estimated using training data) predicts the y's in the validation data set. Once you have done this, answer the three questions below.

Forward model:

```r
# Forward model
pred.forward <- predict(model.forward, newdata = q1.valid)
mspe.forward.valid <- mean((q1.valid$y - pred.forward)^2)
mspe.forward.valid
```

```
## [1] 24.01926
```

Backward model:

```
# Backward model
pred.backward <- predict(model.backward, newdata = q1.valid)
mspe.backward.valid <- mean((q1.valid$y - pred.backward)^2)
mspe.backward.valid
```

```
## [1] 24.01926
```

Cross-validated ridge model (lambda.1se):

```
# Ridge regression
pred.ridge <- predict(cv.ridge, newx = model.matrix(y ~ ., data = q1.valid)[, -1], s = lambda.ridge)
mspe.ridge.valid <- mean((q1.valid$y - pred.ridge)^2)
mspe.ridge.valid
```

```
## [1] 25.36912
```

Cross-validated lasso model (lambda.1se):

```
# Lasso regression
pred.lasso <- predict(cv.lasso, newx = model.matrix(y ~ ., data = q1.valid)[, -1], s = lambda.lasso)
mspe.lasso.valid <- mean((q1.valid$y - pred.lasso)^2)
mspe.lasso.valid
```

```
## [1] 26.96377
```

A) List the mean squared prediction error for each of your candidate models here:

Forward model mean squared prediction error: 24.019 Backward model mean squared prediction error: 24.019 Ridge model mean squared prediction error: 25.369 Lasso model mean squared prediction error: 26.964

B) Based on the mean squared prediction error, which model do you choose?

Your answer here: The forward and backward models both have the lowest MSPE of 24.01926. Since these models yield the same performance on the validation set, either can be chosen.

C) Display the coefficient values for the model you chose here: Proceeding with forward model to avoid duplication of work.

```
# Display coefficients for the chosen forward model
summary(model.forward)$coefficients
```

```
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 4.3637562  0.3637005 11.998212 3.377924e-25
## x7          2.9305430  0.3734886  7.846405 2.760709e-13
## x2          2.0546180  0.4088599  5.025238 1.133810e-06
## x6          1.1548705  0.3669608  3.147122 1.907571e-03
## x9          0.8273288  0.4086073  2.024753 4.425641e-02
```

Q1, Part 3 - Testing the final model on the test set (5 points)

Now that you've chosen a model, you will test this model's generalizability by generating new predictions for the final model chosen in the previous part. Use these new predictions to compute the mean squared prediction error to evaluate how well the final model (which was estimated using training data) predicts the y's in the test data set. Once you've done this, answer the question below:

```
# Test the final model
pred.test <- predict(model.forward, newdata = q1.test)  # Replace with the chosen model
mspe.test <- mean((q1.test$y - pred.test)^2)
mspe.test
```

```
## [1] 30.40355
```

A) What is the mean squared prediction error for the final model making predictions on the test set?

Your answer here: 30.40355

Question 2 - Evaluating the generalizability of a "shrunk" logistic regression model (30 points total)

For this question, a different 400-observation data set was split such that 70% of it was assigned to a training set and the remaining 30% was assigned to a test set. The outcome variable is y, which is binary. The predictor variables, V1 - V10, are all continuous.

Run the code chunk below to load the data into memory before beginning your work on this question.

```
# Load data
log.train <- read.csv("logistic_train.csv", header=TRUE, sep=",")
log.test <- read.csv("logistic_test.csv", header=TRUE, sep=",")

# Note: y is an int type; don't change this for this question
str(log.train)
```

```
## 'data.frame':    300 obs. of  11 variables:
##  $ y  : int  1 0 0 0 0 0 0 0 1 0 ...
##  $ V1 : num  0.8063 -0.0893 1.0258 -2.1566 0.0688 ...
##  $ V2 : num  0.19978 0.4983 0.39872 -1.39152 0.00962 ...
##  $ V3 : num  0.787 0.143 1.129 -2.023 -0.213 ...
##  $ V4 : num  0.1045 0.0849 1.2643 -1.4822 -0.4206 ...
##  $ V5 : num  0.637 0.212 0.968 -2.209 0.216 ...
##  $ V6 : num  -0.19688 -0.48276 0.00274 -0.91685 -0.01067 ...
##  $ V7 : num  0.02089 0.29542 0.71478 -0.4424 -0.00614 ...
##  $ V8 : num  0.0153 0.1858 0.3519 -2.1991 -0.6784 ...
##  $ V9 : num  0.0874 -0.0845 1.0442 -1.1866 0.5171 ...
##  $ V10: num  -0.00807 0.32335 -0.78685 -2.53471 -0.61956 ...
```

```
str(log.test)
```

```
## 'data.frame':    100 obs. of  11 variables:
##  $ y  : int  0 0 1 1 0 1 0 0 0 0 ...
```

```
##  $ V1 : num  0.778 0.151 2.052 0.587 0.824 ...
##  $ V2 : num  0.172 -1.104 1.255 -0.425 0.804 ...
##  $ V3 : num  0.511 -0.478 2.042 0.318 0.853 ...
##  $ V4 : num  0.224 -0.665 1.927 -0.507 1.073 ...
##  $ V5 : num  0.1185 -0.0299 1.7304 0.5088 0.965 ...
##  $ V6 : num  1.381 0.505 1.046 0.752 -0.566 ...
##  $ V7 : num  -1.081 -1.53 0.775 -0.132 0.041 ...
##  $ V8 : num  0.41 -0.411 1.349 0.613 0.255 ...
##  $ V9 : num  0.4726 0.135 1.5779 0.0577 0.6249 ...
##  $ V10: num  0.503 0.321 0.5 1.271 0.384 ...
```

Q2, Part 1 - Fitting the "shrunk" logistic model (5 points)

I fitted a penalized logistic regression model to the training data; that is, I applied a method of coefficient shrinkage to a logistic regression model fitted to the training. Run the code chunk below to fit this model and answer the two questions below.

```
x.train.mat <- model.matrix(y ~ ., data = log.train)[, -1]
y.train.vec <- log.train$y

set.seed(12345)
cvfit.log <- cv.glmnet(x = x.train.mat, y = y.train.vec, family = "binomial", alpha = 1)
coef(cvfit.log, s = "lambda.min")
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) 0.11202810
## V1          0.15054756
## V2          .
## V3          .
## V4          0.52991688
## V5          .
## V6          .
## V7          0.12957868
## V8          0.03803014
## V9          .
## V10         0.09262162
```

A) What type of penalized/"shrunk" logistic regression model did I fit to the training data? *Be specific.*

Your answer here: The model fitted is a Lasso (Least Absolute Shrinkage and Selection Operator) logistic regression model. This type of penalized regression uses the alpha = 1 parameter in cv.glmnet to apply L1 regularization, which enforces sparsity in the model by setting some coefficients to exactly zero.

B) Which predictors (V1 - V10) are included the lambda.min penalized/"shrunk" logistic regression model? Put differently, which predictor coefficients were NOT set to zero in the lambda.min model?

Your answer here (list the included predictors): V1, V4, V7, V8, V10

Q2, Part 2 - Obtaining new predictions from test data (5 points)

In this section, you will generate new predictions of Y based on the values of the predictors in the test set. That is, you will use the predictor values from log.test to predict new y values.

```
x.test.mat <- model.matrix(y ~ ., data = log.test)[, -1]
y.test.pred <- predict(cvfit.log, newx = x.test.mat, s = "lambda.min", type = "class")
```

Next, display the counts of both the predicted y values and the actual y values in log.test. Remember that these are binary variables. Hint: this will help you check your work in the next part.

```
# Counts for predicted y's & Counts for actual y's
table(y.test.pred)
```

```
## y.test.pred
##  0  1
## 39 61
```

```
table(log.test$y)
```

```
##
##  0  1
## 47 53
```

Q2, Part 3 - Evaluating the generalizability on the test set (20 points)

For this part, you will create a confusion matrix and compute the four model indices we discussed in class: accuracy, precision, recall, and F1 score. Use the table() function to create the confusion matrix and be sure to display it in the knitted document. Write your code for this below and answer the two questions below.

```
## Code for creating and displaying the confusion matrix
confusion.matrix <- table(Predicted = y.test.pred, Actual = log.test$y)
confusion.matrix
```

```
##           Actual
## Predicted  0  1
##         0 27 12
##         1 20 41
```

```
## Code for computing and displaying the model indices
accuracy <- sum(diag(confusion.matrix)) / sum(confusion.matrix)
precision <- confusion.matrix[2, 2] / sum(confusion.matrix[2, ])
recall <- confusion.matrix[2, 2] / sum(confusion.matrix[, 2])
F1 <- 2 * (precision * recall) / (precision + recall)

accuracy
```

```
## [1] 0.68
```

```
precision
```

```
## [1] 0.6721311
```

```
recall
```

```
## [1] 0.7735849
```

```
F1
```

```
## [1] 0.7192982
```

A) Fill in the values of your confusion matrix. Note that you should also have the confusion matrix displayed as part of the output from your code.

True positives (your answer here): 41 False positives (your answer here): 20 True negatives (your answer here): 27 False negatives (your answer here): 12

B) Fill in the values of your model indices. Again, be sure that you also have these displayed as part of the output from your code.

Accuracy: 0.68 Precision: 0.672 Recall: 0.773 F1 score: 0.719

Question 3 - Modeling count data (20 points total)

Alyssa analyzed workplace injury incidents obtained from a shipping warehouse. She sampled 45 employee records at random and recorded the number of workplace injury reports in the employee's file in the past five years ("reports") and how many years since the employee underwent safety training certification ("years").

Run the code chunk below to load the data into memory before beginning your work on this question.

```
# Load data
injury <- read.csv("injury.csv", header=TRUE, sep=",")
str(injury)
```

```
## 'data.frame':    300 obs. of  3 variables:
##  $ ID     : int  3 4 14 16 38 50 52 54 57 71 ...
##  $ reports: int  1 0 6 5 1 3 3 4 1 0 ...
##  $ years  : int  0 5 6 5 4 4 9 4 6 4 ...
```

Q3, Part 1 - Describing the data (10 points)

Before starting the analysis, Alyssa conducted some descriptive analyses to present alongside her modeling work. Please recreate the descriptive analyses she conducted by answering the following three questions. Any code you use to answer the questions should go into the chunk below:

```
# Zero or one injury reports
percent_zero_one <- sum(injury$reports <= 1, na.rm = TRUE) / nrow(injury) * 100

# Median years for 6 or more injury reports
median_years_6plus <- median(injury$years[injury$reports >= 6], na.rm = TRUE)
```

```r
# Median reports for safety certification < 5 years
median_reports_5years <- median(injury$reports[injury$years < 5], na.rm = TRUE)

percent_zero_one
```

```
## [1] 7.666667
```

```r
median_years_6plus
```

```
## [1] 8.5
```

```r
median_reports_5years
```

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

A) What percentage of the sampled employees had zero or one injury report in their files?

Your answer here: 7.67%

B) Among workers who had 6 or more injury reports, what was the median number of years since their most recent safety certification?

Your answer here: 8.5 years.

C) Among workers who had a safety certification that was less than 5 years old, what was the median number of injury reports?

Your answer here: 1

Q3, Part 2 - Fitting a quasipoisson model (10 points)

After some exploratory modeling, Alyssa determined that a *quasipoisson* model was the most appropriate model for these data.

First, please re-create her analysis by fitting a quasipoisson model to the data, using reports as the outcome and years as the predictor.

```r
model.quasipoisson <- glm(reports ~ years, family = quasipoisson(), data = injury)
summary(model.quasipoisson)
```

```
##
## Call:
## glm(formula = reports ~ years, family = quasipoisson(), data = injury)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.30079    0.30661  -0.981    0.332
## years        0.23313    0.04511   5.168 5.83e-06 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 1.828019)
##
##     Null deviance: 142.323  on 44  degrees of freedom
## Residual deviance:  91.294  on 43  degrees of freedom
##   (255 observations deleted due to missingness)
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

Next, answer the following three questions about the analysis:

A) Was the number of years since safety training a significant predictor of the number of injury reports?

Your answer here (yes/no): Yes, the number of years since safety training is a significant predictor of the number of injury reports. The p-value for the years variable is much smaller than the standard significance level of 0.05.

B) Based on the results, does the predicted count of workplace injury reports *increase* or *decrease* as the number of years since an employees most recent safety certification increases? How do you know?

Your answer here: The predicted count of workplace injury reports increases as the number of years since the most recent safety certification increases. This is evident from the positive coefficient for the years variable (0.23313), indicating that for each additional year since certification, the expected count of injury reports increases on average.

C) Based on your answer to the previous question, do you think Alyssa recommended greater frequency of safety certification or a lower frequency of safety certification?

Your answer here (higher or lower): Higher. Alyssa likely recommended greater frequency of safety certification. Since the model shows that injury reports increase with more years since the last certification, reducing the time between certifications could help mitigate the risk of workplace injuries.

Question 4 - Modeling time-to-event data (20 points total)

As part of receving services at a community mental health clinic, a prospective client must first submit a completed intake form; upon approval, the client may then schedule a first visit with a provider. Ahmed designed an intervention study to determine if different follow-up schedules ("group") decrease the amount of time in between the client's approval and the client's first appointment ("first_visit", in decimal days). Clients in Group A, which was the current standard for client follow-up at the time, received a reminder via text message to schedule their first appointment 14 days after approval. Clients in Group B received reminders 14 and 21 days after approval, and clients in Group C received reminders 7 and 14 days after approval. If the client saw a provider within 30 days of approval, "status" is equal to 1; otherwise, it's equal to zero.

Run the code chunk below to load the data into memory before beginning your work on this question.

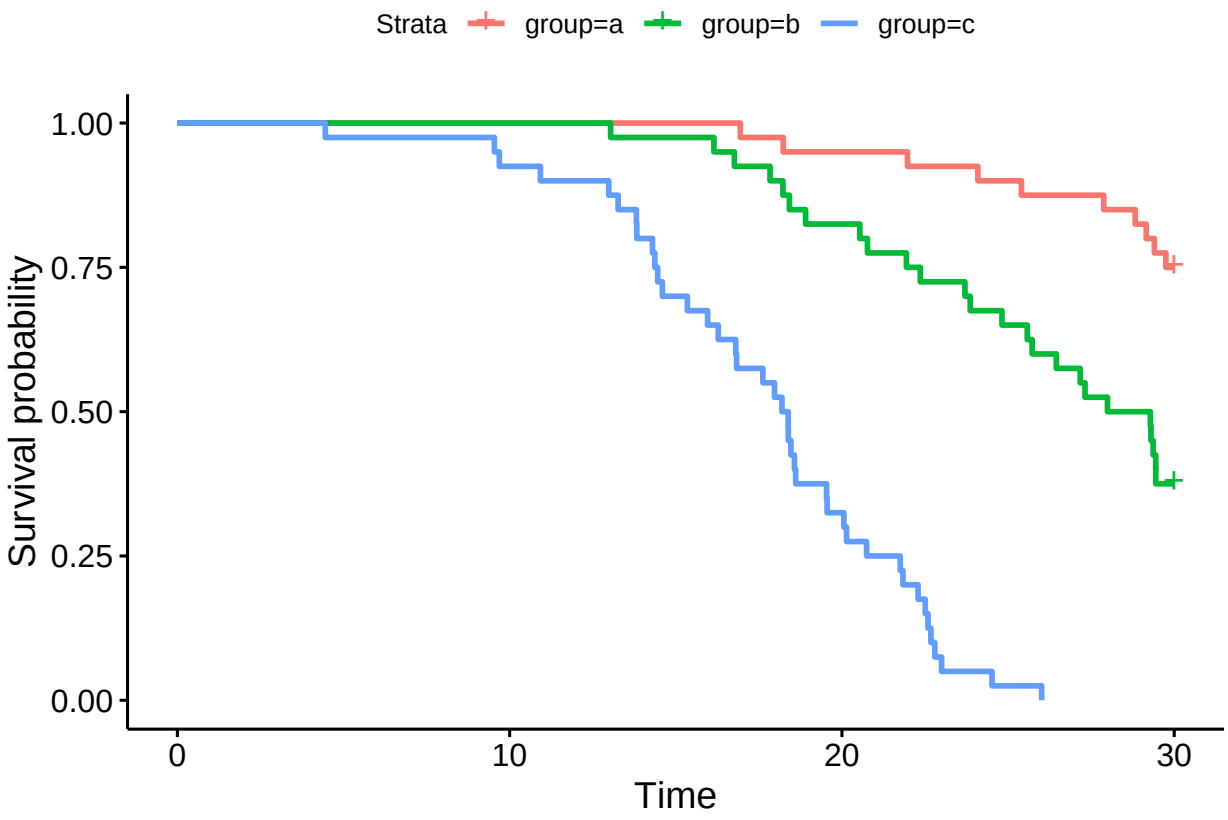
```
# Load data
intake <- read.csv("enrollment.csv", header=TRUE, sep=",")
str(intake)
```

```
## 'data.frame':    120 obs. of  3 variables:
##  $ status     : int  0 0 1 1 0 0 1 0 0 0 ...
##  $ group      : chr  "a" "a" "a" "a" ...
##  $ first_visit: num  30 30 28.8 29.4 30 ...
```

Q4, Part 1 - Kaplan-Meier curves and censoring (10 points)

Ahmed started his analysis by fitting Kaplan-Meier curves for each group, which can be seen by running the code chunk below. Please examine this visualization and answer the three questions below.

```
# Kaplan-Meier Curves
fit <- survfit(Surv(first_visit, status) ~ group, data = intake)
ggsurvplot(fit)
```



A) Based on what you see in the Kaplan-Meier curves, which group had the most censored observations?

Your answer here: Group A appears to have the most censored observations because its survival probability line flattens out earlier compared to groups B and C, indicating that fewer events (first visits) were observed within the 30-day window.

B) The line for group C ends somewhere between 20 and 30 days. What does this mean for members of group C in terms of their time-to-first visit?

Your answer here: The Kaplan-Meier curve for group C ending between 20 and 30 days indicates that most participants in group C had their first visits within that timeframe. After the curve ends, no further events (first visits) were observed, implying the remaining participants were censored by day 30.

C) During the design phase, Ahmed took steps to prevent left censoring from occurring. If this had happened and a client's status was left-censored, what would that have meant with regard to their time-to-first visit?

Your answer here: Left censoring would mean that the exact time of the first visit occurred before the study began (i.e. before Ahmed began observing clients). Preventing left censoring ensures that the time-to-first visit starts at the point of approval and is fully observable within the study period.

Q4, Part 2: Fitting parametric survival models (10 points)

Next, Ahmed fitted two accelerated failure time (AFT) models: an exponential AFT model and a Weibull AFT model. Run the code chunk below and review the results of the two analyses.

```
# Parametric Survival Models
model.weib <- survreg(Surv(first_visit, status) ~ group, dist = "weibull", data = intake)
model.exp <- survreg(Surv(first_visit, status) ~ group, dist = "exponential", data = intake)
summary(model.weib)
```

```
##
## Call:
## survreg(formula = Surv(first_visit, status) ~ group, data = intake,
##     dist = "weibull")
##               Value Std. Error      z      p
## (Intercept)  3.6725     0.0732  50.14 <2e-16
## groupb      -0.2670     0.0832  -3.21 0.0013
## groupc      -0.7144     0.0839  -8.52 <2e-16
## Log(scale)  -1.5413     0.0983 -15.68 <2e-16
##
## Scale= 0.214
##
## Weibull distribution
## Loglik(model)= -261.7   Loglik(intercept only)= -310.6
##  Chisq= 97.9 on 2 degrees of freedom, p= 5.5e-22
## Number of Newton-Raphson Iterations: 6
## n= 120
```

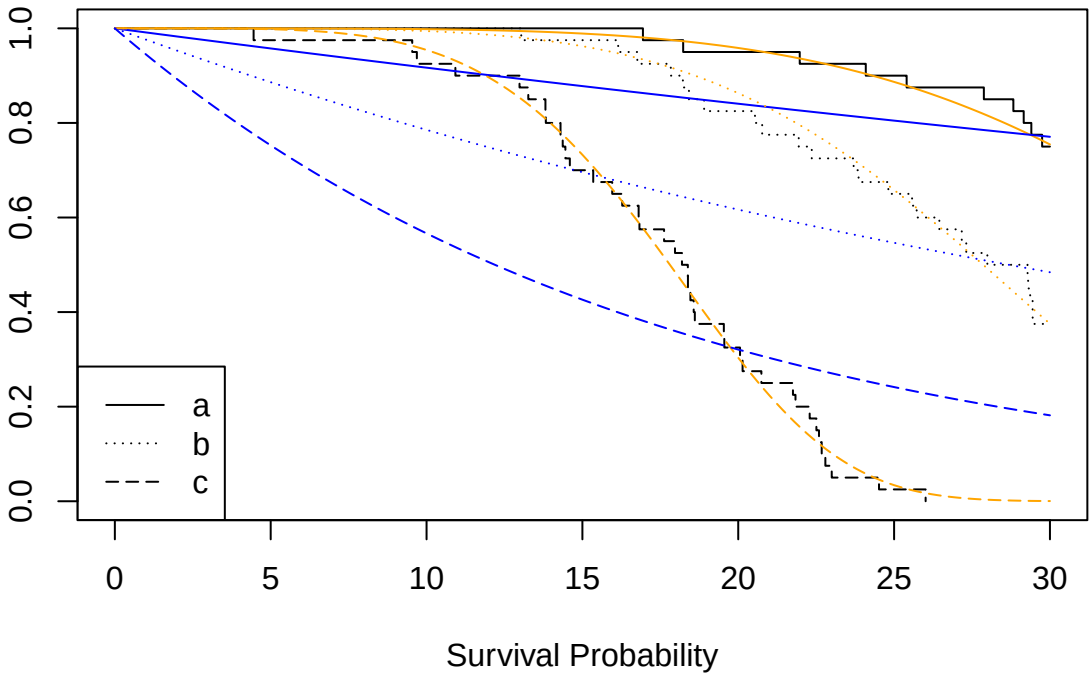
```
summary(model.exp)
```

```
##
## Call:
## survreg(formula = Surv(first_visit, status) ~ group, data = intake,
##     dist = "exponential")
##               Value Std. Error     z       p
## (Intercept)  4.746      0.316 15.01 < 2e-16
## groupb      -1.024      0.374 -2.74  0.0062
## groupc      -1.879      0.354 -5.31 1.1e-07
##
```

```
## Scale fixed at 1
##
## Exponential distribution
## Loglik(model)= -330.2   Loglik(intercept only)= -348.9
##  Chisq= 37.26 on 2 degrees of freedom, p= 8.1e-09
## Number of Newton-Raphson Iterations: 4
## n= 120
```

He then created a Kaplan-Meier plot with overlaid cumulative distribution curves for both models. The orange cumulative distribution curves are derived from the fitted Weibull AFT model, and the blue curves are from the fitted exponential AFT model. Run the code chunk below to view these and answer the four questions below.

```
# Overlay Kaplan-Meier with AFT models
plot(survfit(Surv(first_visit,status) ~ group,
               data = intake),lty=c(1,3,5),xlab="Survival Probability")
legend("bottomleft", c("a", "b","c"), lty = c(1,3,5))
points(seq(0,30,by=.2),
       1-psurvreg(seq(0,30,by=.2),mean=model.weib$coefficients[1],
                  scale=model.weib$scale),type="l",lty=1,col="orange")
points(seq(0,30,by=.2),
       1-pexp(seq(0,30,by=.2),rate=1/exp(model.exp$coefficients[1])),
       type="l", lty=1,col="blue")
points(seq(0,30,by=.2),
       1-psurvreg(seq(0,30,by=.2),mean=sum(model.weib$coefficients[c(1,2)]),
                  scale=model.weib$scale),type="l",lty=3,col="orange")
points(seq(0,30,by=.2),
       1-psurvreg(seq(0,30,by=.2),mean=sum(model.exp$coefficients[c(1,2)]),
                  distribution = "exponential"),type="l",lty=3,col="blue")
points(seq(0,30,by=.2),
      1-psurvreg(seq(0,30,by=.2),mean=sum(model.weib$coefficients[c(1,3)]),scale=model.weib$scale),
       type="l",lty=5,col="orange")
points(seq(0,30,by=.2),
       1-psurvreg(seq(0,30,by=.2),mean=sum(model.exp$coefficients[c(1,3)]),
                  distribution = "exponential"),type="l",lty=5,col="blue")
```



D) Based on the visualization, which of the two models - exponential AFT (blue) or Weibull AFT (orange) - appear to best model the time-to-first visit for the different intervention groups?

Your answer here (exponential or Weibull): The Weibull AFT model (orange) appears to better model the time-to-first visit for the different intervention groups. The orange cumulative distribution curves from the Weibull model align more closely with the Kaplan-Meier survival curves, especially in capturing the differences in survival probabilities across time for all three groups. The exponential model (blue) oversimplifies the survival trends, resulting in poorer fit.

E) Look at the model output for the model you chose in the previous question. Is there evidence of a difference between Group A and Group B?

Your answer here (yes/no): Yes, there is evidence of a difference between Group A and Group B based on the Weibull AFT model output. The coefficient for groupb is statistically significant. This indicates that the follow-up schedule for Group B (Days 14 and 21) significantly impacts the time-to-first visit compared to Group A (Day 14 only).

F) From this same model output, is there evidence of a difference between Group A and Group C?

Your answer here (yes/no): Yes, there is evidence of a difference between Group A and Group C based on the Weibull AFT model output. The coefficient for groupc is statistically significant. This indicates that the follow-up schedule for Group C (Days 7 and 14) significantly impacts the time-to-first visit compared to Group A (Day 14 only).

G) Considering the results shown in both parts 1 and 2 of this question, which follow-up schedule - Day 14 only, Days 7 and 14, or Days 14 and 21 - do you think Ahmed recommended to the clinic? Please note that stating the group letter alone here won't earn full credit; you must explicitly reference the specific intervention the group experienced.

Your answer here: Ahmed likely recommended the Days 7 and 14 follow-up schedule (Group C). Based on both the Kaplan-Meier survival curves and the Weibull AFT model output, Group C exhibited the shortest time-to-first visit, with a significant reduction compared to Groups A and B. This intervention (reminders at Days 7 and 14) effectively increases the likelihood of clients scheduling their first visit sooner, thus achieving the goal of reducing time-to-event.

End.