Problem Set 7, Winter 2024

Michael Ghattas

START:

```r
# Load necessary packages
library(glmnet)
```

```
## Loading required package: Matrix

## Loaded glmnet 4.1-8
```

```r
library(mlbench)
library(haven)
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
```

CONTEXT - HOUSE VALUES IN BOSTON, CIRCA 1970 This dataset was obtained through the mlbench package, which contains a subset of data sets available through the UCI Machine Learning Repository. From the help file: Housing data for 506 census tracts of Boston from the 1970 census. The dataframe BostonHousing contains the original data by Harrison and Rubinfeld (1979). The original data are 506 observations on 14 variables, medv being the target variable:

Continuous variables: crim per capita crime rate by town zn proportion of residential land zoned for lots over 25,000 sq.ft
indus proportion of non-retail business acres per town nox nitric oxides concentration (parts per 10 million) rm average number of rooms per dwelling age proportion of owner-occupied units built prior to 1940 dis weighted distances to five Boston employment centres rad index of accessibility to radial highways tax full-value property-tax rate per USD 10,000 ptratio pupil-teacher ratio by town b 1000(B - 0.63)^2 where B is the proportion of blacks by town lstat percentage of lower status of the population medv median value of owner-occupied homes in USD 1000's

Categorical variables: chas Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

Question 1 - 10 points

First, load the data into memory. The variable types are already stored in this data set.

```
# Load BostonHousing data
data(BostonHousing)
str(BostonHousing)
```

```
## 'data.frame':    506 obs. of  14 variables:
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : num  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ b      : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

Before you begin your analysis, you will split the data into a 70% training set and a 30% test set. First, save the number of rows in the data set for use in the splitting code.

```
n <- nrow(BostonHousing)
```

When splitting data into training/test data sets, it's good practice to set a random seed to create a split that's reproducible. For this question, use the following seed.

```
# Set seed and split data into 70% train, 30% test
set.seed(202211)
tv.split <- sample(rep(0:1, c(round(n * 0.3), n - round(n * 0.3))), n)
table(tv.split)
```

```
## tv.split
##   0   1
## 152 354
```

```
dat.train <- BostonHousing[tv.split == 1,]
dat.test <- BostonHousing[tv.split == 0,]
```

In Problem Set 6, you were shown some code from the async to create a train-validate-test split tvt2 <- sample(rep(0:2,c(round(n.*2),round(n*.2),n-2*round(n*.2))),n)

In this problem, however, you are splitting your data into just training and test sets (i.e., just two groups). You can make some changes to the rep() function contained in this line code to create a split for just train-test. To help you make these adaptations, the following code chunk contains the isolated version of what's contained in the tvt2 rep() function. Run it to see what it produces and then make alterations that will instead produce a set of 0's (test set, 30%) and 1's (training set, 70%) for splitting purposes.

```r
# Set the modified rep() function for a 70/30 split
tvt2.rep <- rep(0:1, c(round(n * 0.3), n - round(n * 0.3)))
# Check the structure of the split
tvt2.rep
```

```
##   [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [38] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [112] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [149] 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [260] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [297] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [334] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [371] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [408] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [445] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [482] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```r
table(tvt2.rep)
```

```
## tvt2.rep
##   0   1
## 152 354
```

Once you've found something that works, insert it into the blank space in tv.split to obtain a 70% training/30% test split. Display a table of the split to verify that approximately 70% of tv.split is equal to 1 and approximately 30% is equal to zero

```r
set.seed(202211)
# Load the data and set up initial variables
data(BostonHousing)
n <- nrow(BostonHousing)
# Create a 70/30 split vector using rep()
tvt2.rep <- rep(0:1, c(round(n * 0.3), n - round(n * 0.3)))
# Randomly shuffle and assign to train/test using sample()
tv.split <- sample(tvt2.rep, n)
# Verify the split proportions
table(tv.split)
```

```
## tv.split
##   0   1
## 152 354
```

```r
# Create the training and test datasets based on tv.split
dat.train <- BostonHousing[tv.split == 1, ]
dat.test <- BostonHousing[tv.split == 0, ]
```

Question 2 - 10 points

After completing Question 1, conduct a cross-validated ridge regression using the training data set. Use medv as the outcome and all of the other variables in the data set as the predictors.

```
set.seed(202211)
# Prepare the predictor matrix (X) and the outcome variable (y) for the training data
X.train <- model.matrix(medv ~ ., data = dat.train)[, -1] # Remove the intercept column
y.train <- dat.train$medv
# Fit a cross-validated ridge regression model (alpha = 0 for ridge)
cvfit.house.ridge <- cv.glmnet(X.train, y.train, alpha = 0, family = "gaussian")
# Display lambda.min, the lambda value that minimizes the cross-validation error
cvfit.house.ridge$lambda.min
```

```
## [1] 0.6558898
```

```
# Display the coefficients associated with lambda.min
coef(cvfit.house.ridge, s = "lambda.min")
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  30.031482758
## crim         -0.105482442
## zn            0.038892030
## indus        -0.020951069
## chas1         2.933854416
## nox         -10.205423687
## rm            3.665364705
## age          -0.012514745
## dis          -1.225165597
## rad           0.152002327
## tax          -0.003662155
## ptratio      -0.833546577
## b             0.008284729
## lstat        -0.529835645
```

For this question, the only lambda of interest is lambda.min. Make sure that lambda.min and the coefficients associated with it are visible in your knitted document.

```
# Display lambda.min, the lambda value that minimizes the cross-validation error
cvfit.house.ridge$lambda.min
```

```
## [1] 0.6558898
```

```
# Display the coefficients associated with lambda.min
coef(cvfit.house.ridge, s = "lambda.min")
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  30.031482758
## crim         -0.105482442
## zn            0.038892030
## indus        -0.020951069
## chas1         2.933854416
## nox         -10.205423687
## rm            3.665364705
## age          -0.012514745
## dis          -1.225165597
## rad           0.152002327
## tax          -0.003662155
## ptratio      -0.833546577
## b             0.008284729
## lstat        -0.529835645
```

Question 3 - 5 points

Using the results from Question 2, compute the mean squared prediction error for the lambda.min model when applied to the *test* data set. Be sure to show how you computed it and to display the result; once you've done that, answer the question below.

```
# Prepare the predictor matrix (X) for the test data
X.test <- model.matrix(medv ~ ., data = dat.test)[, -1] # Remove the intercept column
y.test <- dat.test$medv
# Use the fitted model to predict the median house values (medv) for the test set using lambda.min
predictions <- predict(cvfit.house.ridge, X.test, s = "lambda.min")
# Calculate the Mean Squared Prediction Error (MSPE)
mean((y.test - predictions)^2)
```

```
## [1] 21.12796
```

A) What is the mean squared prediction error you computed (your answer here): 21.13%

CONTEXT - DRINKING AND SENSATION-SEEKING A study is described in Coxe, West, and Aiken (2009) where 400 college students were asked how many alcoholic beverages they had consumed during the previous Saturday. They also completed an eight-item subscale on the Revised NEO Personality Inventory that measures sensation-seeking.

Variables contained in this data set: case: The participant ID y: The number of drinks the participant reported drinking the previous Saturday sensation: The participant's mean score on the eight-item sensation-seeking subscale gender: The participant's gender (0 = female, 1 = male)

Question 4 - 15 points

To date, I've given you data sets that can be read into R using base R functionality. There are numerous statistical software packages that are used out in the world, and it's possible (and, in some industries, likely) that you will encounter data sets that have a proprietary format and can only be directly opened and edited using proprietary softare. One example of this is SAS, which is celebrating it's 50th year since its original launch in 1972. It's native data format is .sas7bdat, which is similar to the .RData files you've used in that it contain metadata along with raw data.

Do you have to have a SAS license just to open a data set a colleague sent you? Not if you know how to open it into R! There are two main packages for reading in non-native data formats into R. The older of the two is the foreign package, and the package that does this in the "tidyverse" constellation of packages is the haven package. For this question, you'll use the haven package to read in the drinking and sensation data set, which is in the .sas7bdat format. The official RStudio page for the haven package can be found here: https://haven.tidyverse.org/. Read it to find which function you will need to use to read in the SAS file.

```
# Load the drinking and sensation-seeking data set from a SAS file
drinking <- read_sas("drinking_coxe_west_aiken.sas7bdat") # Replace "path_to_file" with the actual file
# Display the structure of the data to verify variable types
str(drinking)
```

```
## tibble [400 x 4] (S3: tbl_df/tbl/data.frame)
##  $ gender   : num [1:400] 1 1 1 1 1 1 0 0 1 1 ...
##  $ y        : num [1:400] 4 4 1 6 0 8 6 1 7 8 ...
##  $ sensation: num [1:400] 6.06 5.31 3.94 4.3 5.58 ...
##  $ case     : num [1:400] 1 2 3 4 5 6 7 8 9 10 ...
```

Although I expect it to come in correctly, check that the y and sensation variables are numeric before continuing. We will not be using the case or gender variables in this question. Now you will fit three models using this data: a Poisson model, a quasipossion model, and a negative binomial model. The outcome of these analyses should be *y*, and the predictor should be *sensation*.

Poisson model

```
# Fit the Poisson model
model.poisson <- glm(y ~ sensation, family = poisson(link = "log"), data = drinking)
summary(model.poisson)
```

```
##
## Call:
## glm(formula = y ~ sensation, family = poisson(link = "log"),
##     data = drinking)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.14032    0.21280  -0.659     0.51
## sensation    0.23148    0.03966   5.837 5.33e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 1186.8  on 399  degrees of freedom
## Residual deviance: 1151.7  on 398  degrees of freedom
## AIC: 2079
##
## Number of Fisher Scoring iterations: 5
```

Quasipoisson model

```
# Fit the Quasipoisson model
model.quasipoisson <- glm(y ~ sensation, family = quasipoisson(link = "log"), data = drinking)
summary(model.quasipoisson)
```

```
##
## Call:
## glm(formula = y ~ sensation, family = quasipoisson(link = "log"),
##     data = drinking)
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.14032    0.35907  -0.391 0.696166
## sensation    0.23148    0.06692   3.459 0.000601 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 2.847168)
##
##     Null deviance: 1186.8  on 399  degrees of freedom
## Residual deviance: 1151.7  on 398  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 5
```

Negative binomial model

```
# Fit the Negative Binomial model
model.nb <- glm.nb(y ~ sensation, data = drinking)
summary(model.nb)
```

```
##
## Call:
## glm.nb(formula = y ~ sensation, data = drinking, init.theta = 1.392859438,
##     link = log)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.08261    0.36027  -0.229  0.81863
## sensation    0.22050    0.06822   3.232  0.00123 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(1.3929) family taken to be 1)
##
##     Null deviance: 463.49  on 399  degrees of freedom
## Residual deviance: 452.56  on 398  degrees of freedom
## AIC: 1772.1
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  1.393
##           Std. Err.:  0.163
##
##  2 x log-likelihood:  -1766.091
```

Once you've fit all three models, answer the three questions below.

A) Look at the output for the Poisson model and the quasipoisson model. Which of the two - Poisson or quasipoisson - have *larger* standard errors for the coefficients? Your answer here (Poisson or quasipoisson): Quasipoisson. The standard errors in the Quasipoisson model are larger (e.g., 0.06692 for sensation) compared to those in the Poisson model (e.g., 0.03966 for sensation).

B) Look at the quasipoisson model output. What was the dispersion parameter taken to be in the quasipoisson model? Your answer here: 2.847168. This is shown in the Quasipoisson model output under (Dispersion parameter for quasipoisson family taken to be 2.847168).

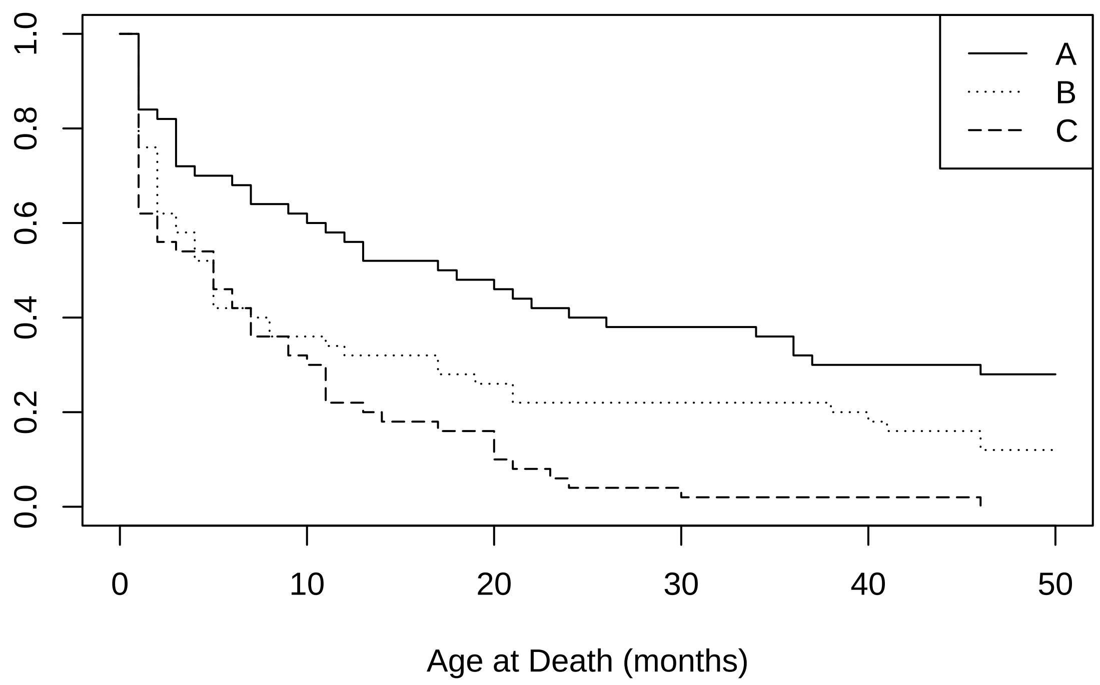
C) Per the guidelines presented in the async and discussed during the live session, which of the three models - Poisson, quasipoisson, and negative binomial - is the best based on the *residual deviance*? Your answer here (Poisson, quasipoisson, or negative binomial): Negative Binomial. The Negative Binomial model has the lowest residual deviance (452.56), indicating a better fit to the data compared to the Poisson (1151.7) and Quasipoisson (1151.7) models.

D) For the model that is the best based on residual deviance, which of the following statements is a correct "gist" interpretation of the coefficient associated with sensation? Statement 1: As the mean score on the sensation-seeking subscale increases, the predicted/expected count of drinks *increases*. Statement 2: As the mean score on the sensation-seeking subscale increases, the predicted/expected count of drinks *decreases*. Your answer here: Statement 1. In the Negative Binomial model, the positive coefficient for sensation (0.22050) indicates that as the mean score on the sensation-seeking subscale increases, the predicted/expected count of drinks also increases.

Question 5 - 15 points

Before beginning this question, please review the material from 9.1.3 in the async material. The following code is excerpted from the example shown in 9.1.3. The outcome of interest is time to death of sheep. Each sheep received some level of anti-parasite treatment; A and B contained actual anti-parasite ingredients and C was a placebo (i.e., no active ingredient in the treatment). Please run the three code chunks and examine their output. Once you've done that, answer the four questions below.

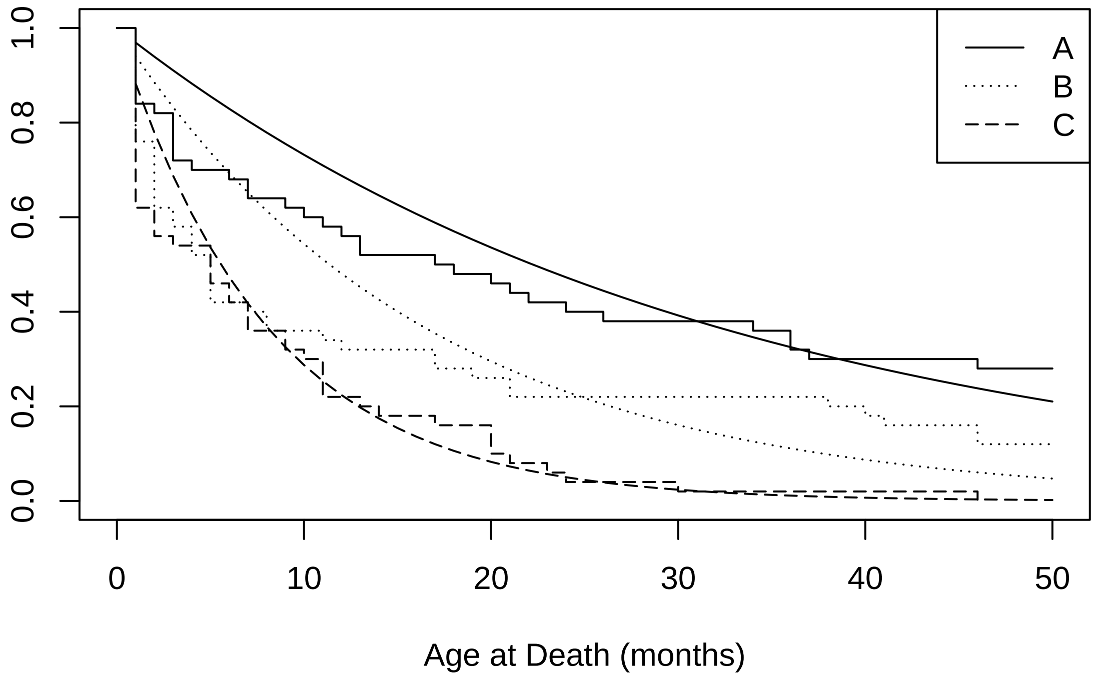
```r
# Chunk 1
sheep<-read.csv("sheep.deaths.csv")
with(sheep,plot(survfit(Surv(death,status)~group),lty=c(1,3,5),xlab="Age at Death (months)"))
legend("topright", c("A", "B","C"), lty = c(1,3,5))
```



```r
# Chunk 2
model<-survreg(Surv(death,status)~group, dist="exponential",data=sheep)
summary(model)
```

```
##
## Call:
## survreg(formula = Surv(death, status) ~ group, data = sheep,
##     dist = "exponential")
##              Value Std. Error     z       p
## (Intercept)  3.467      0.167 20.80 < 2e-16
## groupB      -0.671      0.225 -2.99  0.0028
## groupC      -1.386      0.219 -6.34 2.3e-10
##
## Scale fixed at 1
##
## Exponential distribution
## Loglik(model)= -482   Loglik(intercept only)= -502.1
##  Chisq= 40.35 on 2 degrees of freedom, p= 1.7e-09
## Number of Newton-Raphson Iterations: 5
## n= 150
```

```r
# Chunk 3
plot(survfit(Surv(sheep$death,sheep$status)~sheep$group),lty=c(1,3,5),xlab="Age at Death (months)")
legend("topright", c("A", "B","C"), lty = c(1,3,5))
points(1:50,
       1-pexp(1:50,rate=1/exp(model$coefficients[1])),
       type="l",
      lty=1)
# The survival curve S(t) for group B.
points(1:50,
       1-pexp(1:50,rate=1/exp(sum(model$coefficients[c(1,2)]))),
       type="l",
      lty=3)
# The survival curve S(t) for group C.
points(1:50,
1-pexp(1:50,rate=1/exp(sum(model$coefficients[c(1,3)]))),
       type="l",
      lty=5)
```



# Question about Chunk 1 A) What kind of plot is this? It has a specific name. Your answer here: his is a Kaplan-Meier survival curve plot, which shows the estimated survival function for each group.

B) Which group had the most number of sheep whose outcomes were censored? Your answer here: Group A seems to have a higher survival rate towards the end of the observed period, indicating a greater likelihood of censored outcomes in this group. Groups B and C reach lower survival probabilities earlier, suggesting fewer censored outcomes in these groups.To determine this precisely, we would need a count of censored observations per group from the data. Typically, we can tell this from the plot if the lines plateau higher on the y-axis, but exact numbers require data inspection.

C) In the context of this data, what does it mean if a sheep's outcome was censored? Your answer here: Censoring in this context means that the sheep was still alive at the last recorded time point or was lost to follow-up, meaning the exact time of death is unknown.

Questions about Chunk 2

D) What kind of survival model is being fitted in this code? Be specific. Your answer here: This is an exponential survival model fitted using a parametric approach with the survreg function.

E) Looking at the p-values, is Group A significantly different from Group B? Your answer here: Yes, Group A is significantly different from Group B (p = 0.0028).

F) Looking at the p-values, is Group A significantly different from Group C? Your answer here: Yes, Group A is significantly different from Group C (p = Close to Zero)

G) Looking at the coefficient estimates, which group - B or C - had the lowest predicted survival time? Your answer here: Group C has the lowest predicted survival time, as indicated by the more negative coefficient (-1.386) compared to Group B.

Question about Chunk 3

H) The jagged lines on this plot are the same as those from the plot shown in Chunk 1. What is being visualized by the the *smooth, curved lines* in this plot? Again, be specific. Your answer here: The smooth, curved lines represent the fitted exponential survival curves based on the parametric model from Chunk 2 for each group (A, B, and C). These curves show the theoretical survival probabilities over time according to the exponential model's estimates for each group.

END.