Problem Set 8, Winter 2021

Adam Ten Hoeve

```
# Load any packages, if any, that you use as part of your answers here
# For example:
library(MASS)
library(glmnet)
```

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

```
## Loaded glmnet 4.1
```

```
library(mlbench)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(survival)
```

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:

Continous 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 - 5 points

```
data(BostonHousing) # loads the BostonHousing dataset into memory from the mlbench package

# Any processing code for changing variable types

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 ...
```

Next, split the sample into a training set (70%) and a test set (30%). For convenience, I've provided a seed to make your split reproducable.

```
set.seed(123456)

# Your code here
# Randomize the order of the rows, so no inheret bias is introduced
rows = sample(nrow(BostonHousing))
BostonHousing = BostonHousing[rows, ]
# Use first 70% of rows for training set
train.size = ceiling(nrow(BostonHousing) * 0.7)
data.train = BostonHousing[1:train.size, ]
# Use remaining rows for test set
data.test = BostonHousing[(train.size+1):nrow(BostonHousing), ]

nrow(data.train)
```

```
## [1] 355
```

```
nrow(data.test)
```

```
## [1] 151
```

```
nrow(BostonHousing)
```

```
## [1] 506
```

Question 2 - 10 points

After completing Question 1, conduct a ridge regression with cross-validation using the training data set. Use medv as the outcome and all of the other variables in the data set as the predictors. Be sure to display your lambda.min, lambda.1se, and the set of coefficients associated with both of these lambdas.

```
set.seed(123456)

# Your code here
X = data.matrix(dplyr::select(data.train, -medv))
Y = data.train$medv
ridge.model = cv.glmnet(x=X, y=Y, alpha=0)

# Don't forget to display lambda.min, lambda.1se, and the coefficients for both of these!
ridge.model$lambda.min
```

```
## [1] 0.6743424
```

```
coef(ridge.model, s = "lambda.min")
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept)  25.112910753
## crim         -0.093855267
## zn            0.031899140
## indus        -0.016572477
## chas          2.359508404
## nox         -13.038065938
## rm            3.966437081
## age           0.002583889
## dis          -1.023483134
## rad           0.147623210
## tax          -0.006291648
## ptratio      -0.816836608
## b             0.008836670
## lstat        -0.467956286
```

```
ridge.model$lambda.1se
```

```
## [1] 4.757352
```

```r
coef(ridge.model, s = "lambda.1se")
```

```
## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) 16.823675741
## crim        -0.072534121
## zn           0.017456156
## indus       -0.057621551
## chas         2.526380275
## nox         -5.725786603
## rm           3.542426739
## age         -0.007715778
## dis         -0.421972573
## rad          0.015910065
## tax         -0.002953055
## ptratio     -0.586157118
## b            0.007344843
## lstat       -0.319825455
```

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.

```r
# Your code here
X.test = data.matrix(dplyr::select(data.test, -medv))
Y.test = data.test$medv

test.preds = predict(ridge.model, X.test, s="lambda.min")
# MSE = sum((y_i - hat(y_i))^2)
test.mse = sum((Y.test - test.preds)^2)
cat("The Mean Squared Prediction Error on the test set:", test.mse)
```

```
## The Mean Squared Prediction Error on the test set: 3248.032
```

CONTEXT - NYC BIKERS

The NYC Open Data Portal contains information about the number of cyclists who cross different bridges in the eastern part of New York City. The data for this question is an edited subset of the data available. To see the full data, see https://data.cityofnewyork.us/Transportation/Bicycle-Counts-for-East-River-Bridges/gua4-p9wg.

Variables of interest for this question (all are continuous):

M_bridge_count: The daily count of cyclists who ride across the Manhattan Bridge temp_hi: The highest temperature recorded that day (in Fahrenheit) precipitation: The amount of precipitation recorded that day (in inches)

Question 4 - 15 points

Please fit three models using this data: a Poisson model, a quasipossion model, and a negative binomial model. The outcome of these analyses should be M_bridge_count, and the predictors should be temp_hi and precipitation.

```r
bike<-read.csv("NYCBikes.csv")

# Make any changes you need to make to the variable types

bike$M_bridge_count = as.numeric(gsub(",", "", bike$M_bridge_count))

str(bike)
```

```
## 'data.frame':    83 obs. of  9 variables:
##  $ ï..date       : chr  "1-Apr" "2-Apr" "3-Apr" "4-Apr" ...
##  $ day           : chr  "Saturday" "Sunday" "Monday" "Tuesday" ...
##  $ temp_hi       : num  46 62.1 63 51.1 63 48.9 55.9 66 73.9 80.1 ...
##  $ temp_low      : num  37 41 50 46 46 41 39.9 45 55 62.1 ...
##  $ precipitation : num  0 0 0.03 1.18 0 0.73 0 0 0 0 ...
##  $ B_bridge_count: chr  "606" "2,021" "2,470" "723" ...
##  $ M_bridge_count: num  1446 3943 4988 1913 5276 ...
##  $ W_bridge_count: chr  "1,915" "4,207" "5,178" "2,279" ...
##  $ Q_bridge_count: chr  "1,430" "2,862" "3,689" "1,666" ...
```

Now, fit the three models and display the results of each:

Poisson

```r
# Your code here for Poisson
poisson.model = glm(M_bridge_count~temp_hi+precipitation, bike, family="poisson")
# Don't forget to display the results using the summary() function!
summary(poisson.model)
```

```
##
## Call:
## glm(formula = M_bridge_count ~ temp_hi + precipitation, family = "poisson",
##     data = bike)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -42.87  -15.15   -0.94   13.35   31.00
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    7.247542   0.010796   671.3   <2e-16 ***
## temp_hi        0.019148   0.000147   130.2   <2e-16 ***
## precipitation -0.667930   0.005921  -112.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 71103  on 82  degrees of freedom
## Residual deviance: 29149  on 80  degrees of freedom
## AIC: 30006
##
## Number of Fisher Scoring iterations: 4
```

Quasipoisson

```
# Your code here for quasipoisson
quasipoisson.model = glm(M_bridge_count~temp_hi+precipitation, bike, family="quasipoisson")
# Don't forget to display the results using the summary() function!
summary(quasipoisson.model)
```

```
##
## Call:
## glm(formula = M_bridge_count ~ temp_hi + precipitation, family = "quasipoisson",
##     data = bike)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -42.87  -15.15   -0.94   13.35   31.00
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    7.247542   0.202585  35.775  < 2e-16 ***
## temp_hi        0.019148   0.002759   6.940 9.24e-10 ***
## precipitation -0.667930   0.111113  -6.011 5.21e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 352.1449)
##
##     Null deviance: 71103  on 82  degrees of freedom
## Residual deviance: 29149  on 80  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 4
```

Negative binomial

```
# Your code here for negative binomial
neg.bin.model = glm.nb(M_bridge_count~temp_hi+precipitation, bike)
# Don't forget to display the results using the summary() function!
summary(neg.bin.model)
```

```
##
## Call:
## glm.nb(formula = M_bridge_count ~ temp_hi + precipitation, data = bike,
##     init.theta = 10.46595391, link = log)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.3169  -0.6711  -0.0413   0.6359   2.0752
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)    6.894667   0.228107  30.226  < 2e-16 ***
## temp_hi        0.023885   0.003201   7.462 8.50e-14 ***
## precipitation -0.527340   0.075537  -6.981 2.93e-12 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for Negative Binomial(10.466) family taken to be 1)
##
##     Null deviance: 191.327  on 82  degrees of freedom
## Residual deviance:  84.447  on 80  degrees of freedom
## AIC: 1448.3
##
## Number of Fisher Scoring iterations: 1
##
##
##               Theta:  10.47
##           Std. Err.:  1.61
##
##  2 x log-likelihood:  -1440.292
```

Once you have fit all three models, pick one of these and indicate which one is best. Please note what parts of the output (e.g., dispersion parameter estimates, residual deviance, theta parameter, changes in standard errors/p-value) you used to make your decision. Be specific!

The Poisson model has signficant coefficients, but the Residual deviance is orders of magnitude greater than the degrees of freedom (29149 »> 80). This implies that there is overdispersion, so the Poisson model doesn't fit the data very well. The Quasipoisson model tries to loosen the restrictions, by allowing the dispersion parameter to be estimated, which in our case is 352.1449. All of the coefficients are still significant, but their standard error is greater than the origional Poisson model. The Residual deviance of the Quasipiosson model is still much greater than the degrees of freedom, so their is still the problem of overdispersion. The negative binomial model has a much smaller Residual deviance of 84.447, which is much closer to the degrees of freedom 80. The overdispersion is being accounted for and we can put more trust into the correctness of our model. Therefor, the negative binomial model would be the best.

Question 5 - 15 points

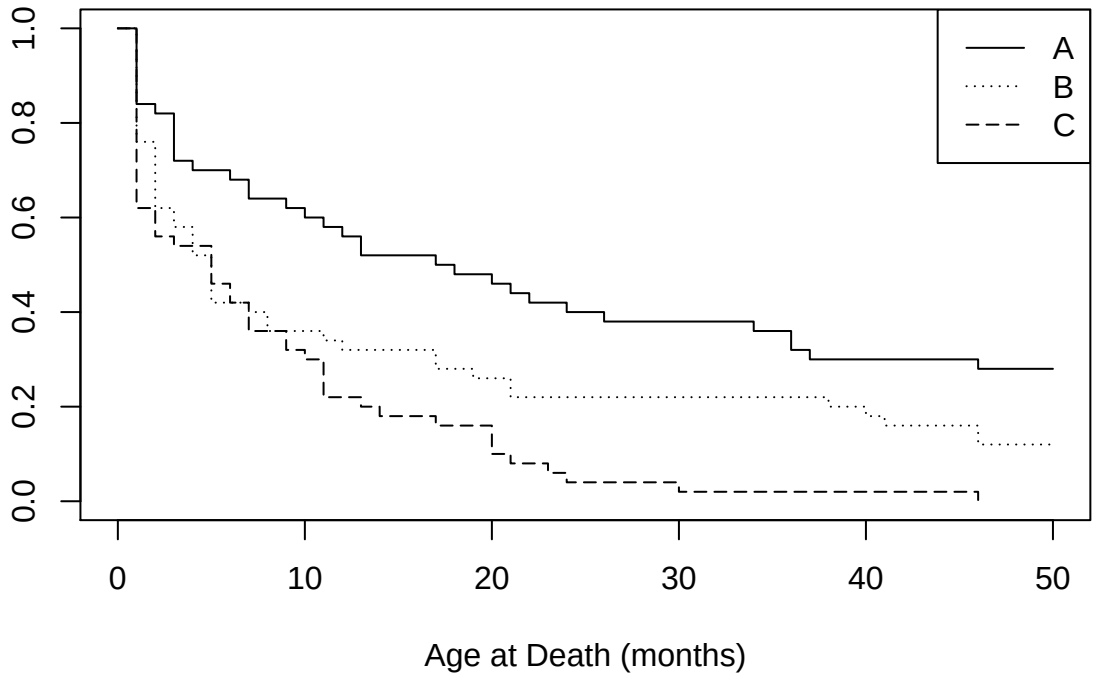
Before beginning this question, please review the material from 9.4.3 in the async material.

The following code is excerpted from the example shown in 9.4.3. Please run the three code chunks and examine their output. Once you've done that, answer the four questions below (5 points per question).

```
# 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))
```




```
# Chunk 2

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

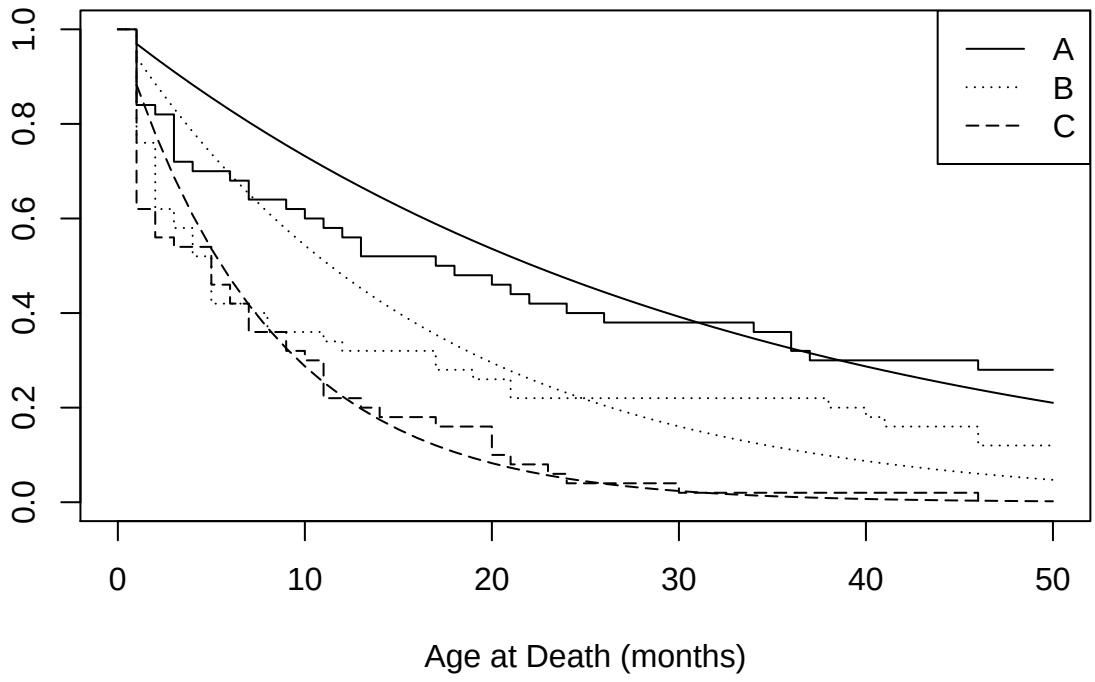
```
##
## Call:
## survreg(formula = Surv(death, status) ~ group, data = sheep,
##     dist = "exponential")
##              Value Std. Error     z       p
## (Intercept)  3.467      0.167 20.80 < 2e-16
## groupB      -0.671      0.225 -2.99  0.0028
## groupC      -1.386      0.219 -6.34 2.3e-10
##
## 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
```

```
# Chunk 3

plot(survfit(Surv(sheep$death,sheep$status)~sheep$group),lty=c(1,3,5),xlab="Age at Death (months)")
```

```r
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)
```



Chunk 1 question: What kind of plot is this? It has a specific name.

Answer here: These are Kaplan Meier Curves.

Chunk 2 question 1: What kind of survival model is being fitted in this code?

Answer here: An Accelerated Failure Time model, fit to an exponential distribution.

Chunk 2 question 2: What does the output of this model suggest about the treatment groups (A, B, and C)?

Answer here: We can tell from the model coefficients which groups will tend to die off faster. Group A is given as the coefficient of the intercept, so 3.467. Because group B has the coefficient -0.671, it's survival time is shorter than group A, on average. And because group C has a coefficient of -1.386, it has an even shorter expected survival time than groups A and B. All of the coefficients have significant p-values, so the differences between groups is also significant.

Chunk 3 question: 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?

Answer here: The smooth curves are the predicted survival function for each group. I.e. the curves use the coefficients to create the predicted survival rate at that value for each group at each age. They are all exponential curves because that is the distribution that the model was fit with.