Biweekly Challenge #2 - Melanoma Survival

Due Friday, October 20th at 7PM

Please show all code and output, but no warnings or messages. Submit the assignment as a PDF file into your Google Drive folder.

The goal of this challenge is to learn about the `dplyr` package and practice logistic regression using a dataset involving melanoma patients and their survival status. `dplyr` is a package part of the tidyverse (which also includes `ggplot2`, `tidyr`, and more) used for data wrangling - the process of transforming raw, unstructured data into a more usable form for modeling or analysis. We would suggest looking through this guide on using `dplyr` before starting the assignment, as well as referencing it while completing it. This document is a more compact representation of `dplyr`'s functions and may be useful to print as a reference.

The package `MASS` contains the dataset we will be using, so install it before starting the assignment if you have not yet done so for the course.

```
#install.packages("MASS")
library(MASS)
library(dplyr)
```

First, let's load in the data and look at the first 5 observations. Type `help(Melanoma)` into your RStudio console to get information about the variables themselves.

```
head(Melanoma)
```

```
##   time status sex age year thickness ulcer
## 1   10      3   1  76 1972      6.76     1
## 2   30      3   1  56 1968      0.65     0
## 3   35      2   1  41 1977      1.34     0
## 4   99      3   0  71 1968      2.90     0
## 5  185      1   1  52 1965     12.08     1
## 6  204      1   1  28 1971      4.84     1
```

We can also view each of the variables in the dataset, along with their types and the first few observations of each.

```
glimpse(Melanoma)  #similar to str() in base R
```

```
## Observations: 205
## Variables: 7
## $ time      <int> 10, 30, 35, 99, 185, 204, 210, 232, 232, 279, 295, 3...
## $ status    <int> 3, 3, 2, 3, 1, 1, 1, 3, 1, 1, 1, 3, 1, 1, 1, 3, 1, 1...
## $ sex       <int> 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1...
## $ age       <int> 76, 56, 41, 71, 52, 28, 77, 60, 49, 68, 53, 64, 68, ...
## $ year      <int> 1972, 1968, 1977, 1968, 1965, 1971, 1972, 1974, 1968...
## $ thickness <dbl> 6.76, 0.65, 1.34, 2.90, 12.08, 4.84, 5.16, 3.22, 12....
## $ ulcer     <int> 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
```

Exercise 1. Notice that the variables `sex` and `ulcer` are both categorical, yet are coded in the dataset as integers. Using `dplyr`, change them both to factors, with the labels of sex being "Female" for 0 and "Male" for 1. (hint: the `factor` function from base R will be useful here, as well as `mutate` from `dplyr`. To view more information about a function in RStudio, type `?functionName`.) *5 pts*

Exercise 2. Create a new categorical response variable called `died` that indicates whether the given patient died from melanoma or not, which corresponds to having a 1 in the `status` column. This is called an indicator variable, so if the value of `died` is 1, it indicates that `died` is true. 0 means the person did not die. (Hint: the `mutate` and `ifelse` functions will come in handy here. Be sure to use a logical operator in the `ifelse` function.) *5 pts*

Now we will introduce another very important part of using the `dplyr` package, the pipe function. The pipe operator (`%>%`, or shift+ctrl+m) can be used to simplify commands which require multiple function calls in succession. It does so by taking the output of one function and passing it as input to the next one (read it like saying "then" in between function calls.). As an example, consider the following command, which calls `summary` on the data for only women:

```
summary(filter(Melanoma, sex == "Female"))
```

```
##       time          status          sex           age       
##  Min.   :  99   Min.   :1.000   Female:126   Min.   : 4.00  
##  1st Qu.:1636   1st Qu.:2.000   Male  :  0   1st Qu.:42.00  
##  Median :2059   Median :2.000                Median :54.00  
##  Mean   :2283   Mean   :1.833                Mean   :51.56  
##  3rd Qu.:3131   3rd Qu.:2.000                3rd Qu.:64.75  
##  Max.   :5565   Max.   :3.000                Max.   :89.00  
##       year        thickness      ulcer       died       
##  Min.   :1962   Min.   : 0.100   0:79   Min.   :0.0000  
##  1st Qu.:1968   1st Qu.: 0.970   1:47   1st Qu.:0.0000  
##  Median :1971   Median : 1.620          Median :0.0000  
##  Mean   :1970   Mean   : 2.486          Mean   :0.2222  
##  3rd Qu.:1972   3rd Qu.: 3.060          3rd Qu.:0.0000  
##  Max.   :1974   Max.   :17.420          Max.   :1.0000
```

It can equivalently be written using the pipe operator like so:

```
Melanoma %>%
  filter(sex == "Female") %>%
  summary()
```

```
##       time          status          sex           age       
##  Min.   :  99   Min.   :1.000   Female:126   Min.   : 4.00  
##  1st Qu.:1636   1st Qu.:2.000   Male  :  0   1st Qu.:42.00  
##  Median :2059   Median :2.000                Median :54.00  
##  Mean   :2283   Mean   :1.833                Mean   :51.56  
##  3rd Qu.:3131   3rd Qu.:2.000                3rd Qu.:64.75  
##  Max.   :5565   Max.   :3.000                Max.   :89.00  
##       year        thickness      ulcer       died       
##  Min.   :1962   Min.   : 0.100   0:79   Min.   :0.0000  
##  1st Qu.:1968   1st Qu.: 0.970   1:47   1st Qu.:0.0000  
##  Median :1971   Median : 1.620          Median :0.0000  
##  Mean   :1970   Mean   : 2.486          Mean   :0.2222  
##  3rd Qu.:1972   3rd Qu.: 3.060          3rd Qu.:0.0000  
##  Max.   :1974   Max.   :17.420          Max.   :1.0000
```

We pipe the output of one function to the input of another one - so instead of writing `filter(Melanoma, sex == "Female")`, `dplyr` is passing the argument 'Melanoma' into the filter function. The data is then filtered based on the specifed logical operator, and that subset of the data is passed into the summary command. Piping is especially useful when there is a long chain of function calls in one step.

`dplyr` makes it easy to chain many of its verbs with the pipe operator to output or operate on only specific subsets of the data, which in many cases can be incredibly useful. Consider the following table:

```
Melanoma %>%
  group_by(sex) %>%
  summarise (n = n(), age = mean(age), thickness = mean(thickness), died = mean(died)) %>%
  arrange(desc(n))
```

```
## # A tibble: 2 x 5
##      sex     n      age thickness      died
##   <fctr> <int>    <dbl>     <dbl>     <dbl>
## 1 Female   126 51.56349  2.486429 0.2222222
## 2   Male    79 53.89873  3.611139 0.3670886
```

First, we take the dataset of Melanoma and use `group_by` to arrange the data into categorical groupings, meaning that all future operations will be applied to both groups at the same time. Using `summarise`, we can collapse the whole dataset into a table of specified summary statistics, in this case the mean age, mean tumor thickness, and percentage that died for each gender. The `n()` function (usable only inside `summarise`, `mutate`, and `filter`) counts the number of observations in the current group or groups. Finally, we sort the table in descending order based on the number of observations using the `arrange` function.

Exercise 3. Create a table like the one above, but instead group by `ulcer`, and include only those above the age of 65. Comment briefly on what the results of the table implies about the survival of melanoma patients. *5 pts*

Exercise 4. Using `dplyr`, create a subset of the data called Melanoma2 that only includes `sex`, `age`, `thickness`, `ulcer`, and `died`. Fit a GLM to this new set of data using all of the variables to predict `died`. In a few sentences, interpret the coefficients of this model. Are there certain variables that could be removed from the model without much effect? *5 pts*