Lab Worksheet 7

In 1898, Hermon Bumpus, an American biologist working at Brown University, collected data on one of the first examples of natural selection directly observed in nature. Immediately following a bad winter storm, he collected 136 English house sparrows, *Passer domesticus*, and brought them indoors. Of these birds, 64 had died during the storm, but 72 recovered and survived. By comparing measurements of physical traits, Bumpus demonstrated physical differences between the dead and living birds. He interpreted this finding as evidence for natural selection as a result of this storm:

```
bumpus <- read.csv("http://wilkelab.org/classes/SDS348/data_sets/bumpus_full.cs
v")
head(bumpus)
```

```
##    Sex   Age Survival Length Wingspread Weight Skull_Length Humerus_Length
## 1 Male Adult    Alive    154        241   24.5         31.2           17.4
## 2 Male Adult    Alive    160        252   26.9         30.8           18.7
## 3 Male Adult    Alive    155        243   26.9         30.6           18.6
## 4 Male Adult    Alive    154        245   24.3         31.7           18.8
## 5 Male Adult    Alive    156        247   24.1         31.5           18.2
## 6 Male Adult    Alive    161        253   26.5         31.8           19.8
##   Femur_Length Tarsus_Length Sternum_Length Skull_Width
## 1         17.0          26.0           21.1        14.9
## 2         18.0          30.0           21.4        15.3
## 3         17.9          29.2           21.5        15.3
## 4         17.5          29.1           21.3        14.8
## 5         17.9          28.7           20.9        14.6
## 6         18.9          29.1           22.7        15.4
```

The data set has three categorical variables (`Sex` , with levels `Male` and `Female` , `Age` , with levels `Adult` and `Young` , and `Survival` , with levels `Alive` and `Dead`) and nine numerical variables that hold various aspects of the birds' anatomy, such as wingspread, weight, etc.

Split the `bumpus` data set into a random training and test set. Use 70% of the data as a training set.

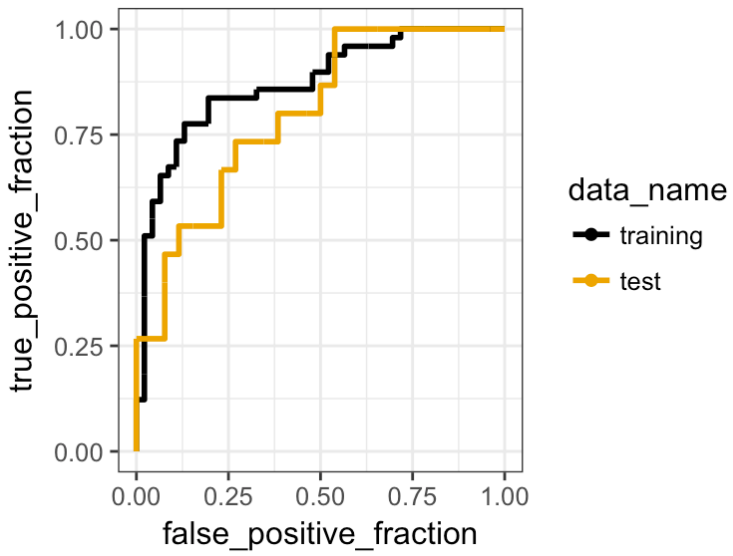
```
train_fraction <- 0.7 # fraction of data for training purposes
set.seed(123)  # set the seed to make your partition reproductible
train_size <- floor(train_fraction * nrow(bumpus)) # number of observations in
training set
train_indices <- sample(1:nrow(bumpus), size = train_size)

train_data <- bumpus[train_indices, ] # get training data
test_data <- bumpus[-train_indices, ] # get test data
```

Fit a logistic regression model on the training data set, then predict the survival on the test data set, and

plot the resulting ROC curves.

```
# model to use:
# Survival ~ Sex + Length + Weight + Humerus_Length + Sternum_Length

glm.out.train <- glm(Survival ~ Sex + Length + Weight + Humerus_Length + Sternu
m_Length,
                data=train_data,
                family=binomial)

# results data frame for training data
df_train <- data.frame(predictor = predict(glm.out.train, train_data),
                       known_truth = train_data$Survival,
                       data_name = "training")

# results data frame for test data
df_test <- data.frame(predictor = predict(glm.out.train, test_data),
                       known_truth = test_data$Survival,
                       data_name = "test")

df_combined <- rbind(df_train, df_test)

ggplot(df_combined, aes(d = known_truth, m = predictor, color = data_name)) +
  geom_roc(n.cuts = 0) + scale_color_colorblind()
```

```
## Warning in verify_d(data$d): D not labeled 0/1, assuming Alive = 0 and Dead
## = 1!
```



2. Area under the ROC curves

Calculate the area under the training and test curve for this following model.

```
# model to use:
# Survival ~ Weight + Humerus_Length

train_fraction <- 0.6 # fraction of data for training purposes
set.seed(101)  # set the seed to make your partition reproductible
n_obs <- nrow(bumpus) # number of observations in bumpus data set
train_size <- floor(train_fraction * nrow(bumpus)) # number of observations in
training set
train_indices <- sample(1:n_obs, size = train_size)

train_data <- bumpus[train_indices, ] # get training data
test_data <- bumpus[-train_indices, ] # get test data

glm.out.train <- glm(Survival ~ Weight + Humerus_Length,
                data=train_data,
                family=binomial)

# results data frame for training data
df_train <- data.frame(predictor = predict(glm.out.train, train_data),
                       known_truth = train_data$Survival,
                       data_name = "training")

# results data frame for test data
df_test <- data.frame(predictor = predict(glm.out.train, test_data),
                       known_truth = test_data$Survival,
                       data_name = "test")

df_combined <- rbind(df_train, df_test)

p <- ggplot(df_combined, aes(d = known_truth, m = predictor, color = data_name)
) +
  geom_roc(n.cuts = 0) + scale_color_colorblind()

calc_auc(p)
```

```
## Warning in verify_d(data$d): D not labeled 0/1, assuming Alive = 0 and Dead
## = 1!
```

```
##   PANEL group       AUC
## 1     1     1 0.6966463
## 2     1     2 0.7661290
```

```
model <- unique(df_combined$data_name)
model_info <- data.frame(model,
                         group = order(model))

left_join(model_info, calc_auc(p)) %>%
  select(-group, -PANEL) %>%
  arrange(desc(AUC))
```

```
## Warning in verify_d(data$d): D not labeled 0/1, assuming Alive = 0 and Dead
## = 1!
```

```
## Joining, by = "group"
```

```
##      model       AUC
## 1     test 0.7661290
## 2 training 0.6966463
```

3. If this was easy

Use the worksheet from the previous class (class 13) to write code that generates an arbitrary number of random subdivisions of the data into training and test sets, fits a given model, calculates the area under the curve for each test data set, and then calculates the average and standard deviation of these values.

```
# function that does the heavy lifting
generate_AUC_values <- function(data, formula, train_fraction)
{
  n_obs <- nrow(data) # number of observations in data set
  train_size <- floor(train_fraction * nrow(data)) # number of observations in
training set
  train_indices <- sample(1:n_obs, size = train_size)

  train_data <- data[train_indices, ] # get training data
  test_data <- data[-train_indices, ] # get test data
  glm.out <- glm(formula, data=train_data, family=binomial)

  df_train <- data.frame(predictor = predict(glm.out, train_data),
                         known_truth = train_data$Survival,
                         data_name = "AUC_train")

  df_test <- data.frame(predictor = predict(glm.out, test_data),
                         known_truth = test_data$Survival,
                         data_name = "AUC_test")

  df_combined <- rbind(df_train, df_test)
  p <- ggplot(df_combined, aes(d = known_truth, m = predictor, color = data_nam
e)) +
    geom_roc(n.cuts = 0)

  data_name <- unique(df_combined$data_name)
  data_info <- data.frame(data_name,
                          group = order(data_name))
  left_join(data_info, calc_auc(p)) %>%
    select(-group, -PANEL) %>%
    spread(data_name, AUC)
}

# example use
generate_AUC_values(bumpus, Survival ~ Wingspread, 0.2)
```

```
## Joining, by = "group"
```

```
##   AUC_train AUC_test
## 1 0.4914773 0.469845
```

```
# function that does repeated random subsampling validation
subsample_validate <- function(data, formula, train_fraction, replicates)
{
  reps <- data.frame(rep=1:replicates) # dummy data frame to iterate over
  reps %>% group_by(rep) %>% # iterate over all replicates
    do(generate_AUC_values(data, formula, train_fraction)) %>% # run calc_AUC f
or each replicate
    ungroup() %>%     # ungroup so we can summarize
    summarize(mean_AUC_train = mean(AUC_train),        # summarize
              sd_AUC_train = sd(AUC_train),
              mean_AUC_test = mean(AUC_test),
              sd_AUC_test = sd(AUC_test)) %>%
    mutate(train_fraction=train_fraction, replicates=replicates) # add columns
containing meta data
}
```

Now that we have these two functions, we can use them to complete the exercise. (We set message = FALSE and warning = FALSE for this R chunk so that we don't get repeated messages and warnings in the knitted html.)

```
train_fraction <- 0.2 # fraction of data for training purposes
replicates <- 20 # how many times do we want to randomly sample
set.seed(116) # random seed
model <- Survival ~ Length + Humerus_Length # the model we want to fit
subsample_validate(bumpus, model, train_fraction, replicates)
```

```
## # A tibble: 1 x 6
##   mean_AUC_train sd_AUC_train mean_AUC_test sd_AUC_test train_fraction
##            <dbl>        <dbl>         <dbl>       <dbl>          <dbl>
## 1          0.746       0.0748         0.712      0.0376          0.200
## # ... with 1 more variable: replicates <dbl>
```

```
# redo with a different model
model2 <- Survival ~ Sex + Length + Weight + Humerus_Length + Sternum_Length
subsample_validate(bumpus, model2, train_fraction, replicates)
```

```
## # A tibble: 1 x 6
##   mean_AUC_train sd_AUC_train mean_AUC_test sd_AUC_test train_fraction
##            <dbl>        <dbl>         <dbl>       <dbl>          <dbl>
## 1          0.880       0.0740         0.751      0.0662          0.200
## # ... with 1 more variable: replicates <dbl>
```