- *Coding:* The Rmd template has code for fitting a lasso model to the training set, determining the value of $\lambda$ that minimizes the cross-validation error, and calculating the test RMSE for the best model. The only change required is to add the interaction variable when setting up the design matrix in the training set as well as in the test set. To repeat the whole process for ridge regression, simply set `alpha` to `0` instead of `1`. Unlike Task 5, there is no need to change the family argument from `gaussian` to `Gamma`, which is not allowed by the `glmnet()` function. The fact that `glmnet()` has restrictions on the model form is a downside to regularization.

**Task 10 – Consider a decision tree** ***(5 points)*** In this task you are asked to describe the pros and cons of using a regression decision tree, which is an alternative model for predicting `Crash_Score`. No fitting is needed. You will find the advantages and disadvantages discussed towards the end of Subsection 5.1.1 useful. Before describing the pros and cons, it is also a good idea to give a precise and concise description of decision trees as in the model solution:

> "A regression decision tree is an alternative method of linking predictors to a target variable. A tree divides the feature space into a finite, non-overlapping set of buckets. All observations in a given bucket have the same predicted value."

Note that because almost all of the variables in this exam are factors, advantages and disadvantages that are specific to continuous variables score lower.

It is expected that future exams will have one task like this to assess your conceptual understanding of an alternative predictive model not tested in the project. An exam project may, for example, test GLMs and base decision trees in the main body of the project and ask you to consider the merits and demerits of using ensemble trees like random forests and boosting. At a minimum, you should list and briefly describe *at least two to three* advantages and disadvantages. The more, the better, so long as your points are well justified and not conflicting with one another.

**Task 11 – Executive summary** ***(20 points)*** At long last, you are asked to summarize your analysis using an executive summary. Please refer to the "Structure of Executive Summary" section earlier on how to write a good executive summary. You can see that the model solution indeed follows the suggested 4-part structure very closely. Also notice that the solution makes good use of the information given on the first page of the project statement, e.g., "If this investigation looks promising, they will provide statewide data for further analysis" is rephrased in the concluding paragraph as a possible next step.

Note that the executive summary should only cover your work in Tasks 1-8, but not Task 9. In other words, there is no need to (and we shouldn't) discuss what we did using the ridge regression and lasso models.

## Hospital Readmissions Sample Project

This sample project was released in May 2019, one month before the June 2019 PA exam (yes, the SOA announced the change of exam format just one month before the June exam!!). Similar to the June exam, this sample project has a total of 11 miscellaneous tasks covering data exploration, data cleaning, model construction, feature selection, and model interpretation. Here we are hired by a group of hospitals to construct a GLM for identifying patients who are most likely to be readmitted based on the given patient level data. The target variable is `Readmission.Status`, a binary variable that reflects whether a patient is readmitted or not. The GLM constructed is required to outperform the LACE index, an existing tool for predicting readmission (details of LACE are not required for this project), with respect to the AUC.

Your virtual "assistant" has supplied code chunks for doing binarization, the training/test split, and combining factor levels. You will have to decide in which tasks these code chunks have to be used.

## Task 1 – Perform univariate exploration of the four non-factor variables *(6 points)*

- *Content:* This warm-up task bears a striking resemblance to Task 1 of the June 2019 PA exam and asks that you explore the distribution of the four non-factor variables, `ER`, `LOS`, `Age`, and `HCC.Riskscore`, with the aid of *both* graphical displays and summary statistics. It is a simple test of the univariate data exploration skills you learned in Subsection 2.2.1. The summary statistics can be obtained from the chunk loading the data, but you have to decide what kind of graphical displays to use for the four variables. Although they are all non-factors, `ER` is a discrete variable with a narrow range (0 to 9) and so a frequency table or a bar chart is better than a histogram (which is for a continuous variable) for visualizing its distribution.

  Because you are asked to consider only the four non-factor variables, you are expected to describe in some detail the characteristics of their distribution. For each variable, you can comment on the following items:

  ▷ What type of variable is it? Is it a count variable or a continuous variable?

  ▷ Describe the key summary statistics, e.g., the range of the variable, how its mean and median compare, and any interesting features you notice (e.g., right skew, presence of outliers, multi-mode).

  For both `LOS` and `HCC.Riskscore`, their distribution has a pronounced right skew which can be reduced by a log transformation. If you are interested, you can look at the histogram of the two log-transformed variables. You will find that their distribution is much closer to a bell shape after the log transformation. As suggested by the task statement, the log-transformed variables will be used throughout the rest of the project.

  You may notice that the model solution creates an additional binary variable called `Under65` indicating whether a patient is below age 65 or above, based on the description of the `age` variable in the data dictionary. The creation of this variable, not requested by the task statement, is not absolutely necessary. In fact, this new variable will be dropped when we perform stepwise selection later.

  One possibly important thing that is not addressed in the model solution but may be worth mentioning (as suggested by the June 2019 exam) is the distribution of the target variable `Readmission.Status`. You may take the initiative to point out that its mean is 0.1259, meaning that only 12.59% of the 66,782 patients were readmitted. This speaks to the imbalanced nature of the target variable, something not dealt with by this project (note the sentence "For this assignment, do not perform undersampling or oversampling" in the project statement).

- *Coding:* You are given code to create a histogram for `ER` using the `ggplot2` package. As mentioned above, a frequency table or a bar chart, which can be created respectively by the `table()` function and the `geom_bar()` function (recall what you learned in Section 2.2), will work better for `ER`. You can simply replace `ER` by `LOS`, `Age`, and `HCC.Riskscore` in the given code to create a histogram for each of the three variables. Towards the end of this task, you will create the two log-transformed variables, called `logLOS` and `logRiskscore` in the model solution. Just remember to drop the original variables.

## Task 2 – Examine relationships between `DRG.Class` and `DRG.Complication` *(5 points)*

- *Content:* This task, also on data exploration, requests that you look at the two factor variables `DRG.Class` and `DRG.Complication`, and suggest how they can be combined.

  Placing the two variables in a two-way classification table shows that although there are $3 \times 5 = 15$ possible combinations in theory, there are only eight combinations with one or more entries. Moreover, the combination with `DRG.Class = SURG` and `DRG.Complication = MedicalMCC.CC` is not possible as surgical patients should be classified to `SurgMCC.CC`, `SurgNoC`, or `Other` (here, you have to look at what the two variables really mean according to the data dictionary). Those six records should be due to a reporting error and be removed. Following the removal, there are only seven distinct combinations, which are carried by a newly created factor variable, called `DRG` in the model solution.

- *Coding:* Code is provided to produce a two-way classification table for `DRG.Class` and `DRG.Complication`. You will have to delete the six inconsistent observations by logical subsetting (what we learned in Section 1.3 is useful!). A series of `ifelse` statements are then used to define the new `DRG` variable according to the seven combinations (if you are more comfortable working with Excel, feel free to do the manipulations there, then load the CSV file back into RStudio). Do remember to delete `DRG.Class` and `DRG.Complication` at the end. The model solution also relevels the `DRG` variable, but this makes no difference in this case.

## Task 3 – Use observations from cluster analysis to consider a new feature *(9 points)*

- *Content:* This task is the cluster analysis version of Task 3 of the June 2019 exam. You are specifically asked to run a $k$-means cluster analysis on `LOS` and `Age` (or their log-transformed version), explain the importance of the `nstart` argument of the `kmeans()` function, and create a new factor variable that can replace `LOS` and `Age`. All of these items are covered in Subsection 6.2.3 of this manual.

  You can begin your write-up with a one- or two-sentence description of what a $k$-means cluster analysis does, e.g., The model solution says

  > "A cluster analysis attempts to partition the observations into $k$ subsets."

  Then proceed to explain the role played by the `nstart` argument in mitigating the effect of the random initial cluster centers on the final solution. As we learned in Section 6.2, it is advisable to set `nstart` to a larger integer (e.g., 20 to 50). The model solution uses 20 to save time. The elbow plot does not show an unequivocal elbow, but any value from 3 to 6 could be used with justification. The model solution chooses 5 clusters and generates the corresponding cluster labels.

- *Coding:* You are given all of the necessary code to run a $k$-means cluster analysis from $k = 1$ to $k = 12$, to generate an elbow plot, and to create the cluster labels variable (based on 8 clusters for the purposes of illustration). You merely have to change `LOS` to `logLOS`, change `nstart` to a larger integer such as 20 and change `as.factor(km8$cluster)` to `as.factor(km5$cluster)`,

| Predictor Combination | Numeric Target Variable | Categorical Target Variable |
|---|---|---|
| Continuous × Categorical | Scatterplot for the target variable and the continuous predictor color distinguished by the categorical predictor (e.g., Figure 2.2.7) | Box plot for the continuous variable split by the target variable and faceted by the other categorical predictor (e.g., Figure 2.2.9) |
| Categorical × Categorical | Box plot for the target variable split by one categorical predictor and faceted by the other categorical predictor (e.g., Figure 2.2.9) | Bar chart for one categorical predictor filled by the target variable and faceted by the other categorical predictor (e.g., Figure 6.2.5) |

Table 6.3: Graphical displays for detecting the interaction between different types of predictors.

depending on how many clusters you choose. Incidentally, the SOA uses the following two lines to create the group assignments:

```
LOS_Age_Clust <- as.factor(km5$cluster)
cluster_vars$LOS_Age_Clust <- LOS_Age_Clust
```

You can shorten them to just one line:

```
cluster_vars$LOS_Age_Clust <- as.factor(km5$cluster)
```

Do remember to scale the variables prior to running a $k$-means cluster analysis. The given code has performed standardization, but you may have to insert additional code to do standardization yourself on the exam.

## Task 4 – Select an interaction *(5 points)*

- *Content:* This task is about detecting the presence of interaction among `Gender`, `Race`, `ER`, and `HCC.Riskscore` (or their transformed version). You are required to make a conjecture on which two variables are intuitively likely to interact and use graphical displays to either confirm or reject the conjecture. Note that here the target variable `Readmission.Status` is a binary variable and some of the variables are factors (`Gender`, `Race`, `ER`) and `HCC.Riskscore` is continuous. The nature of the variables will determine what kind of graphical displays should be used. Table 6.3 summarizes the appropriate type of graphical displays for different kinds of target variables and predictors. For more practice, refer to Task 5 of Section 4.3.

  The model solution ends up using the interaction between `Gender` and `Race`. To be honest, I find the readmission rates to be almost identical not only across the two genders, but also across the four races. This speaks to the weak extent of interaction between `Gender` and `Race`, as well as to the limited predictive power of `Gender` as a predictor for `Readmission.Status`. In fact, you will see in Task 7 that `Gender`, `Race`, and the interaction variables are all eliminated when we perform stepwise selection (interaction variables are also dropped out of the final model in the June 2019 PA exam!). With the aid of perhaps a microscope, you can see that the readmission rate is (marginally) higher for black males than for black females while the opposite is true for Hispanic.

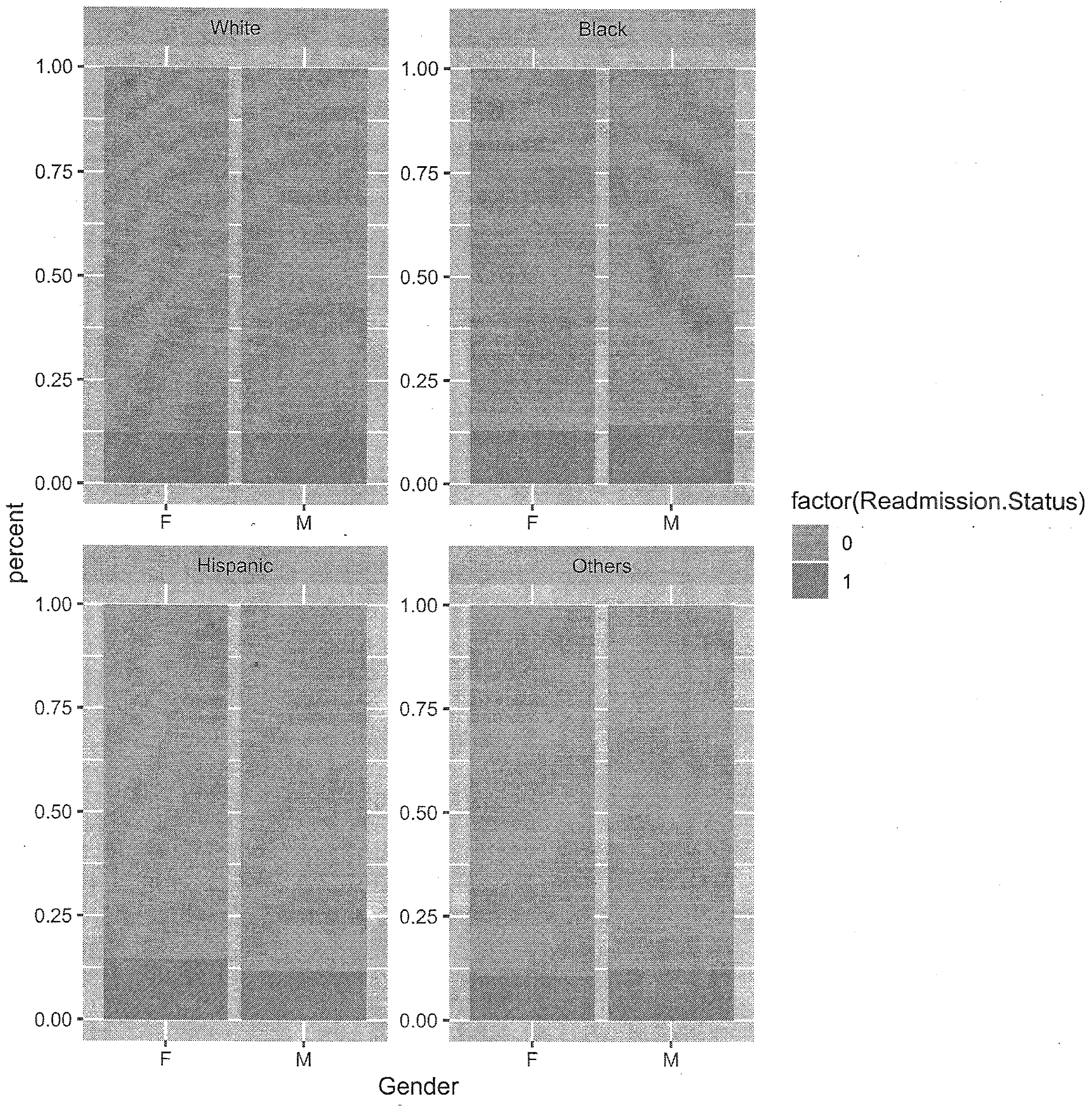

Figure 6.2.5: The bar charts for detecting the interaction between `Gender` and `Race` in the Hospital Readmissions sample project magnified.

- *Coding:* You are supplied with code to investigate the presence of interaction between two factor variables (`Gender` and `Race`) and between a factor variable (`ER`) and a continuous numeric variable (`HCC.Riskscore`). Just adapt the code with the names of the variables you are interested in and replace `HCC.Riskscore` by its log-transformed counterpart `logRiskscore`.

## Task 5 – Select a link function *(8 points)*

- *Content:* The construction of a GLM for `Readmission.Status` begins in Task 5, where you are asked to specify a link function for the GLM out of the five given choices (the target distribution need not be chosen as the binomial distribution is the only appropriate distribution for a binary target variable). The log link can be safely ruled out as it fails to ensure that the predictions are always in the range between 0 and 1. The other four links all produce unit-valued predictions and you can select any two as long as you justify your choices. You are asked to select the superior model with respect to a certain criterion (e.g., AIC, AUC).

  In this task, the SOA's model solution leaves some to be desired and does not explain clearly why the logit and probit links are selected. As the June 2019 exam model solution says,

  > "[b]eing more comfortable with one method, a method being the *default*, or a method being *commonly used* are not satisfactory justifications for selecting a method,"

  while the solution of the sample project says

  > "[f]rom my prior studies it appears that the logit function is the one *most commonly used.* Also, it is the *default choice* in R and is the canonical link function."

  I also find the sentence "the connection [of the probit link] to the normal distribution may make this one easier to explain" not very convincing.

  There are (at least) two ways to improve the SOA's solution:

  1. You could have mentioned that in addition to making appropriate predictions, a good link function should also produce easily interpretable results. The logit link has the clear advantage of ease of interpretation due to its connections to the log link, so the coefficients of the model are closely related to multiplicative changes in the odds of readmission rates. Given that the model with the logit link has the same AUC as the model with the probit link (and a largely indistinguishable AIC), I would choose the logistic regression model on grounds of interpretability even though this project is more about prediction accuracy than about interpretability.
  2. You can also explicitly point out how you make use of the AIC and AUC to decide between the two models. Briefly describe what they are (recall what you learned in Subsections 3.2.2 and 4.1.3) and say that the lower the AIC and the higher the AUC, the better the model.

- *Coding:* The Rmd template has code to run a logistic regression model on the training set and produce an ROC curve, a confusion matrix, and the AUC on the test set. The interaction variable between `Gender` and `Race` has been added for illustration purposes, but you should ensure that it does correspond to what you suggested in Task 4 (in this sample project, this is indeed the desired interaction variable). Also, interaction variables may or may not be added in the code given to you; the June 2019 PA exam is a good example where you should take extra care to add the variable.

As a minor note, the SOA's code does not specify the `positive` argument of the `confusionMatrix()` function. By default, the function will take the first level, namely `"0"`, as the factor level that corresponds to a positive result. As a result, what is called the sensitivity (resp. specificity) in the R output is in fact the specificity (resp. sensitivity) we are interested in. This is not a big problem as the hospital is mostly interested in AUC as the validation metric. You can see that with a cutoff of 0.5, both models fail miserably in classifying readmitted patients as readmitted, with an extremely low sensitivity. The selection of a good cutoff is the subject of Task 9.

### Task 6 – Decide on the factor variable from Task 3 *(5 points)*

- *Content:* This short task explores whether the group assignments created in Task 3 as a result of $k$-means cluster analysis are a useful feature. You are asked to drop `Age` and `LOS` (or `logLOS`) and add the group assignments variable to the data. The model you select in Task 5 is then rerun and its performance on the test set assessed.

  It turns out that replacing `Age` and `LOS` by the cluster groups variable leads to a slight drop in the test AUC and a slight rise in the AIC, no matter whether you use the probit regression model or the logistic regression model. It is therefore reasonable to retain `Age` and `LOS` (or `logLOS`) and ignore the cluster groups variable.

- *Coding:* No code is provided for this task. Although you can adapt the code given in Task 5, there are a few traps you should carefully avoid:

  1. Because we are not sure whether the cluster groups variable or `Age` and `logLOS` will be retained eventually, it is a wise move to save the `readmission` data as a new data frame, say `readmission.cluster`, where `Age` and `logLOS` are dropped and the cluster groups variable is added. This new data frame has to be partitioned into the training set and the test set, called `train.cluster` and `test.cluster` respectively, using the `partition` vector created earlier. We did this kind of re-partition in some case studies earlier, e.g., CHUNK 12 of Section 3.4 and CHUNK 15 of Section 4.3.
  2. When you rerun the probit or logistic regression model on `readmission.cluster`, you should make sure that all references to `train` and `test` are changed to `train.cluster` and `test.cluster`, respectively. Otherwise, you would be fitting the same model as in Task 5. To avoid making mistakes, you may find it helpful to highlight the entire code chunk and select `Edit > Replace and Find` in the menu bar of RStudio.

  Because the cluster groups variable will not be used, we can safely ignore `readmission.cluster`, `train.cluster`, and `test.cluster` for the rest of this project.

### Task 7 – Select features *(15 points)*

- *Content:* This is the task that carries the second highest number of points in the whole project (second to the executive summary task) and is one of the highlights of this sample project. Here you are asked to simplify the model by dropping features with limited predictive power. The catch here is that each level of the factor variables needs to be treated as a separate feature when the `stepAIC()` function is run so that the individual factor levels can be merged with the baseline level (instead of all levels being either retained or removed) to form a bigger baseline level. This calls for explicit binarization prior to fitting the model you select. The existence of the interaction variables makes this task even more complicated. For more practice, refer to Sections 3.4 and 4.3 (Task 5).

Unlike Task 6 of the June 2019 PA exam, the focus here is not on choosing the selection criterion (AIC vs. BIC) or selection process (forward vs. backward), but on binarization, so the default (backward selection with the AIC) suffices. Quite a lot of features are eliminated even when the AIC[xii] (which has a tendency to retain features) is used. Only `logLOS`, `Age`, `logRiskscore`, `DRGOtherSURG`, and `DRGOtherMED` remain, and all interaction variables are removed. To our delight, the AIC of the model decreases slightly and the test AUC increases following stepwise selection.

- *Coding:* No code is provided for this task. You will have to make good use of the code chunk given at the beginning of the Rmd template for doing binarization as well as write some new code involving the `stepAIC()` function from the `MASS` package.

When you look at the SOA's code in the model solution, chances are that you will be lost because the code is unnecessarily complicated. Apparently, the SOA is unaware of the fact that the `dummyVars()` function in the `caret` package can take care of interaction variables easily. Some of the comments in the Rmd solution are also mind-boggling:

▷ *"I do it this way [`fullRank = FALSE`] because if `fullRank` is TRUE I can't be sure about which level becomes the base."*

When `fullRank` is set to `TRUE`, the baseline levels of the categorical variables are dropped. Because we have always re-leveled the variables, these baseline levels are the levels with the most number of observations.

▷ *"I next run the GLM on the binarized data. In doing so, I recognized that the interaction variable created combinations that were redundant. I need to remove all the interactions with male and then re-partition the data."*

Because the SOA has set `fullRank = FALSE`, every level of each categorical variable gets binarized, leading to perfect collinearity. It is unclear why the SOA picks the combinations involving `M` when those with `F` are much more populous.

In the Rmd file that accompanies this section of the manual, I have supplied code that considerably simplifies the SOA's code while leading to the same final model output. If necessary, first run CHUNKs 1-4 to set up the `readmission` data that includes all of the changes made to the data up to this task. If you are on Version 3.6.x of R, note that I have executed the command `RNGkind(sample.kind = "Rounding")` to exactly match the SOA's output.

In CHUNK 5, we use the `dummyVars()` function to binarize `DRG` (created in Task 2), `Race`, `Gender`, and the interaction variables between `Race` and `Gender` (which are inserted by the term `Race*Gender` in the formula). Note that although `Gender` is a two-level factor, it still needs to be binarized due to its interaction with `Race` (`Under65`, on the other hand, need not be binarized). The `fullRank` argument is set to `TRUE` so that the baseline levels of `DRG`, `Race`, and `Gender` are left out. Recall from earlier parts of this project that the baseline levels of `DRG`, `Race`, and `Gender` are `Med.C`, `White`, and `F`, respectively.

[xii] If you use the BIC, then only `Age` and `logRiskscore` remain.

```
# CHUNK 5
binarizer <- caret::dummyVars(~ DRG + Race*Gender,
                              data = readmission,
                              fullRank = TRUE)
binarized_vars <- data.frame(predict(binarizer, newdata = readmission))
head(binarized_vars)

##   DRG.Med.NoC DRG.OtherMED DRG.OtherSURG DRG.Surg.C DRG.Surg.NoC
## 1           0            1             0          0            0
## 2           0            0             0          0            1
## 3           0            0             0          0            1
## 4           1            0             0          0            0
## 5           1            0             0          0            0
## 6           0            0             0          1            0
##   DRG.UNGROUP Race.Black Race.Hispanic Race.Others Gender.M
## 1           0          0             0           0        1
## 2           0          0             0           0        1
## 3           0          0             0           0        1
## 4           0          0             0           0        1
## 5           0          0             0           0        1
## 6           0          0             0           0        1
##   RaceBlack.GenderM RaceHispanic.GenderM RaceOthers.GenderM
## 1                 0                    0                  0
## 2                 0                    0                  0
## 3                 0                    0                  0
## 4                 0                    0                  0
## 5                 0                    0                  0
## 6                 0                    0                  0
```

As expected, only the non-baseline levels of `DRG`, `Race`, and `Gender` are binarized. In addition, there are a total of $3 \times 1 = 3$ interaction variables formed by multiplying the three non-baseline levels of `Race` (`Black`, `Hispanic`, `Others`) with the single non-baseline level of `Gender` (`M`). With CHUNK 5, the binarization can be done very neatly without the need to worry about which dummy variables are redundant.

In CHUNK 6, we attach the binarized variables and remove the three original factor variables.

```
# CHUNK 6
readmission.bin <- cbind(readmission, binarized_vars)
readmission.bin$DRG <- NULL
readmission.bin$Race <- NULL
readmission.bin$Gender <- NULL
summary(readmission.bin)

##  Readmission.Status       ER              Age            logLOS
##  Min.   :0.0000     Min.   :0.0000   Min.   : 24.00   Min.   :0.000
##  1st Qu.:0.0000     1st Qu.:0.0000   1st Qu.: 67.00   1st Qu.:1.099
##  Median :0.0000      Median :0.0000    Median : 75.00    Median :1.609
##  Mean   :0.1259      Mean   :0.5083    Mean   : 73.64    Mean   :1.653
##  3rd Qu.:0.0000      3rd Qu.:1.0000    3rd Qu.: 83.00    3rd Qu.:2.079
##  Max.   :1.0000      Max.   :9.0000    Max.   :101.00    Max.   :3.584
##   logRiskscore         Under65         DRG.Med.NoC       DRG.OtherMED
##  Min.   :-2.5383   Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.: 0.1017   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median : 0.6238   Median :0.0000   Median :0.0000   Median :0.00000
##  Mean   : 0.6000   Mean   :0.1684   Mean   :0.1843   Mean   :0.08022
##  3rd Qu.: 1.1547   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   : 2.5102   Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##  DRG.OtherSURG       DRG.Surg.C       DRG.Surg.NoC     DRG.UNGROUP
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.000   Min.   :0.000000
##  1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:0.000000
##  Median :0.00000   Median :0.0000   Median :0.000   Median :0.000000
##  Mean   :0.05128   Mean   :0.2316   Mean   :0.173   Mean   :0.008446
##  3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.:0.000   3rd Qu.:0.000000
##  Max.   :1.00000   Max.   :1.0000   Max.   :1.000   Max.   :1.000000
##    Race.Black      Race.Hispanic      Race.Others         Gender.M
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.00000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.0000
##  Median :0.0000   Median :0.00000   Median :0.00000   Median :0.0000
##  Mean   :0.1063   Mean   :0.01926   Mean   :0.03404   Mean   :0.4309
##  3rd Qu.:0.0000   3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.00000   Max.   :1.00000   Max.   :1.0000
##  RaceBlack.GenderM RaceHispanic.GenderM RaceOthers.GenderM
##  Min.   :0.00000   Min.   :0.000000     Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.000000     1st Qu.:0.00000
##  Median :0.00000   Median :0.000000     Median :0.00000
##  Mean   :0.04403   Mean   :0.008012     Mean   :0.01598
##  3rd Qu.:0.00000   3rd Qu.:0.000000     3rd Qu.:0.00000
##  Max.   :1.00000   Max.   :1.000000     Max.   :1.00000
```

As soon as the binarized data is set up, the rest of the coding can be completed in the same way as the model solution. The model output is shown below for completeness. In CHUNK 7 we re-partition the data into the training and test sets using the `partition` vector again.

```
# CHUNK 7
train.bin <- readmission.bin[partition, ]
test.bin <- readmission.bin[-partition, ]
```

CHUNK 8 reruns the probit regression model to ensure that we get the same model output as Task 5 as a way to check that the binarization is done properly.

```
# CHUNK 8
glmprobit <- glm(Readmission.Status ~ . ,
                data = train.bin,
                family = binomial(link = "probit"))

summary(glmprobit)

##
## Call:
## glm(formula = Readmission.Status ~ ., family = binomial(link = "probit"),
##     data = train.bin)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2689  -0.5756  -0.3944  -0.2384   3.2478
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.4371865  0.0741325 -19.387  < 2e-16 ***
## ER                  -0.0025486  0.0093214  -0.273  0.78454
## Age                 -0.0045131  0.0008786  -5.137 2.79e-07 ***
## logLOS               0.0343572  0.0112270   3.060  0.00221 **
## logRiskscore         0.7121243  0.0124659  57.126  < 2e-16 ***
## Under65             -0.0408896  0.0314227  -1.301  0.19316
## DRG.Med.NoC         -0.0216338  0.0230263  -0.940  0.34746
## DRG.OtherMED         0.0679331  0.0298657   2.275  0.02293 *
## DRG.OtherSURG        0.0540437  0.0362048   1.493  0.13551
## DRG.Surg.C           0.0086575  0.0215849   0.401  0.68835
## DRG.Surg.NoC        -0.0045170  0.0236140  -0.191  0.84830
## DRG.UNGROUP          0.0704562  0.0778915   0.905  0.36571
## Race.Black           0.0237364  0.0321619   0.738  0.46050
## Race.Hispanic        0.0763654  0.0711823   1.073  0.28335
## Race.Others         -0.0712955  0.0606013  -1.176  0.23941
## Gender.M            -0.0138744  0.0168726  -0.822  0.41090
## RaceBlack.GenderM    0.0159955  0.0493390   0.324  0.74579
## RaceHispanic.GenderM -0.1665655  0.1147644  -1.451  0.14668
## RaceOthers.GenderM   0.1305341  0.0854834   1.527  0.12676
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 38117  on 50081  degrees of freedom
## Residual deviance: 33928  on 50063  degrees of freedom
## AIC: 33966
##
## Number of Fisher Scoring iterations: 5

predsprobit <- predict(glmprobit, newdata = test.bin, type = "response")
```

```
roc(test.bin$Readmission.Status, predsprobit, auc = TRUE)

##
## Call:
## roc.default(response = test.bin$Readmission.Status, predictor =
predsprobit,     auc = TRUE)
##
## Data: predsprobit in 14643 controls (test.bin$Readmission.Status 0) < 2051
cases (test.bin$Readmission.Status 1).
## Area under the curve: 0.7324
```

Finally, CHUNK 9 runs the `stepAIC()` function on the full probit regression model, saves it as an object also named `glmprobit`, and computes its test AUC.

```
# CHUNK 9
library(MASS)
glmprobit <- stepAIC(glmprobit)

summary(glmprobit)

predsprobit <- predict(glmprobit, newdata = test.bin, type = "response")

roc(test.bin$Readmission.Status, predsprobit, auc = TRUE)
```

```
##
## Call:
## glm(formula = Readmission.Status ~ Age + logLOS + logRiskscore +
##     DRG.OtherMED + DRG.OtherSURG, family = binomial(link = "probit"),
##     data = train.bin)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2840  -0.5759  -0.3948  -0.2391   3.2705
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.5092975  0.0469985 -32.114  < 2e-16 ***
## Age           -0.0037101  0.0005667  -6.547 5.88e-11 ***
## logLOS         0.0343635  0.0112216   3.062   0.0022 **
## logRiskscore   0.7115770  0.0124458  57.174  < 2e-16 ***
## DRG.OtherMED   0.0710677  0.0273260   2.601   0.0093 **
## DRG.OtherSURG  0.0558636  0.0341427   1.636   0.1018
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 38117  on 50081  degrees of freedom
## Residual deviance: 33938  on 50076  degrees of freedom
## AIC: 33950
##
## Number of Fisher Scoring iterations: 5
##
## Call:
## roc.default(response = test.bin$Readmission.Status, predictor =
predsprobit,     auc = TRUE)
##
## Data: predsprobit in 14643 controls (test.bin$Readmission.Status 0) < 2051
cases (test.bin$Readmission.Status 1).
## Area under the curve: 0.7334
```

The model output is identical to that obtained by the SOA, but our code for doing binarization is much more transparent and less prone to coding errors.

The model solution suggests that a

> "reasonable additional step here would be to note that the coefficients of `DRGOtherSURG` and `DRGOtherMED` are similar. There may be improvement by combining them into a single factor level."

If you are interested, CHUNK 10 combines these two levels as a single level using the code provided at the beginning of the Rmd template (you can use more informative names than I do!) and CHUNK 11 evaluates the test performance of the resulting probit regression model. The new model has the same test AUC (0.7334) but a slightly lower AIC (well, a unit drop is still an improvement!).

```
# CHUNK 10
library(plyr)
var <- "DRG"
var.levels <- levels(readmission[, var])
readmission[, var] <- mapvalues(readmission[, var],
                                var.levels,
                                c("Non-other", "Non-other", "Other", "Other",
                                  "Non-other", "Non-other", "Non-other"))
#Relevel
table <- as.data.frame(table(readmission[, var]))
  max <- which.max(table[, 2])
  level.name <- as.character(table[max, 1])
  readmission[, var] <- relevel(readmission[, var], ref = level.name)

table(readmission[, var])


##
## Non-other     Other
##     57995      8781
```

```
# CHUNK 11
train <- readmission[partition, ]
test <- readmission[-partition, ]

glmprobit <- glm(Readmission.Status ~ Age + logLOS + logRiskscore + DRG,
                 data = train,
                 family = binomial(link = "probit"))

summary(glmprobit)

##
## Call:
## glm(formula = Readmission.Status ~ Age + logLOS + logRiskscore +
##     DRG, family = binomial(link = "probit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.2798  -0.5759  -0.3948  -0.2393   3.2702
##
## Coefficients:
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -1.5094134  0.0469984 -32.116  < 2e-16 ***
## Age           -0.0037109  0.0005667  -6.548 5.82e-11 ***
## logLOS         0.0345254  0.0112140   3.079  0.00208 **
## logRiskscore   0.7114709  0.0124424  57.181  < 2e-16 ***
## DRGOther       0.0652577  0.0220938   2.954  0.00314 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 38117  on 50081  degrees of freedom
## Residual deviance: 33939  on 50077  degrees of freedom
## AIC: 33949
##
## Number of Fisher Scoring iterations: 5

predsprobit <- predict(glmprobit, newdata = test, type = "response")

roc(test.bin$Readmission.Status, predsprobit, auc = TRUE)

##
## Call:
## roc.default(response = test.bin$Readmission.Status, predictor =
predsprobit,     auc = TRUE)
##
## Data: predsprobit in 14643 controls (test.bin$Readmission.Status 0) < 2051
cases (test.bin$Readmission.Status 1).
## Area under the curve: 0.7334
```

## Task 8 – Interpret the model *(6 points)*

- *Content:* Similar to Task 8 of the June 2019 PA exam, you are asked to rerun the final model on the full dataset and interpret its output. If you use the logistic regression model all along, you will have an easier time doing this task because the coefficient estimates are strongly related to the percentage changes ($e^{\hat{\beta}_j}-1$) or multiplicative changes ($e^{\hat{\beta}_j}$) in the odds of readmission. The model solution adopts the probit regression model, whose coefficients are much more difficult to interpret. To get a feel for the coefficients, the model solution considers the "median patient," whose predictor variable values or levels are the median of each variable, i.e., `logLOS = 1.609`, `Age = 75`, `logRiskscore = 0.6238`, `DRGOtherSURG = DRGOtherMED = 0` (these choices can be obtained from the model summary in Task 2). Then we vary one and only one feature at a time and look at the effect of each feature on the predicted readmission probability. We also demonstrated this way of interpretation towards the end of Subsection 4.2.3 (see Task 9 there). The bottom line is:

  Never say that the model is difficult to interpret and write nothing!

- *Coding:* No code is again provided for this task because you only have to rerun the final model on the full dataset by changing the `data` argument of the `glm()` function from `train.bin` to `readmission.bin`. If you interpret the model based on the "median patient," you will have to manually create a data frame carrying all the combinations of predictor values of interest.

## Task 9 – Set the cutoff *(9 points)*

- *Content:* This task is perhaps the most interesting task in the whole project and demonstrates a business application of predictive analytics driven by practical considerations. It features a cost-benefit approach to optimizing the cutoff for predicting readmission. As we learned in Subsection 4.1.3, lowering the cutoff has the effect of predicting more readmitted patients, leading to a rise in sensitivity but a drop in specificity. The key here is to trade off the two types of mistakes (classifying readmitted patients as non-readmitted and classifying non-readmitted patients as readmitted) with the goal of minimizing the cost to the hospital. Note that AUC, which we used in earlier tasks, does not directly address which cutoff is the best.

  Here is the methodology. For a given cutoff, we generate the confusion matrix for the final model on the full dataset (not the test set):

| | Reference ( = Actual) | |
|---|---|---|
| Prediction | 0 | 1 |
| 0 | $n_{00}$ (\$0) | $n_{01}$ (\$25) |
| 1 | $n_{10}$ (\$2) | $n_{11}$ (\$2) |

  where $n_{ij}$ is the number of patients (dependent on the cutoff) who are predicted to be of class $i$ and belong to class $j$. We then associate each entry of the confusion matrix with the corresponding cost. Since each patient predicted by the GLM to be readmitted (i.e., those in the second row of the confusion matrix) will receive the intervention each for a cost of 2 and not be readmitted, and each false negative (i.e., those belonging to $n_{01}$) will be readmitted each for a cost of 25, the overall cost to the hospital is

$$\text{cost} = 2n_{10} + 2n_{11} + 25n_{01}.$$

We repeat the whole procedure for a range or grid of cutoff values, compute the corresponding cost, and select the cutoff that minimizes the cost. To convince the hospital that our model is useful and leads to savings, the minimum cost should be lower than not employing the intervention program at all (same as setting the cutoff at 1) and employing the intervention program to every discharged patient (same as setting the cutoff at 0).

- *Coding:* The Rmd template has generously provided code to calculate the cost as a linear combination of entries in the confusion matrix (note that a `confusionMatrix` object is a list with `table` being one of its components). For some unknown reason, the cost of the intervention program is erroneously set to 4 and you will have to change it to 2. Then run the whole chunk with different values of `cutoff` and record the corresponding cost.

**Task 10 – Consider alternative models and model construction techniques** ***(12 points)*** This task parallels Task 10 of the June 2019 PA exam and asks that you describe the pros and cons of alternative predictive models, including regularized regression, base classification trees, and random forests, for predicting readmission. No fitting is needed. You will find the discussions in Subsections 3.2.5, 5.1.1, and 5.1.2 useful. Note that because the focus of this project is on prediction accuracy, advantages and disadvantages that relate to interpretability are secondary.

**Task 11 – Executive summary** ***(20 points)*** Like the June 2019 PA exam, this sample project ends with the executive summary task with the hospital being the intended audience. Compared to the executive summary in the June exam, the executive summary in the model solution here is perhaps too brief. Items that you can add to enrich the model solution are (this list is not exhaustive):

- *Key variables in the data:* Point out what variables are in the patient level dataset. At a minimum, describe the characteristics of the target variable.

- *Feature selection:* The model solution has almost no mention of how feature selection is performed, despite Task 7 being an important task in this project. There is no need to describe the details of binarization, but the motivation for feature selection is something of interest to the hospital.

- *Model validation:* If you wish, you can describe the AUC, which is the validation metric in this project, in more detail.

- *Model interpretation:* The model solution merely points out the directional effect of each feature on the readmission rate. Consider following the June 2019 exam model solution and using a table to give more precise statements for each coefficient estimate. Also comment on whether the signs of the coefficient estimates make sense or not in the context of this project.

The following are also worth noting:

- The model solution nicely makes a mention that `Race` and `Age` (and perhaps `Gender`) may be inappropriate predictors due to legal or ethical concerns. The use of these variables as predictors should be discussed with the hospital before the GLM is put to real use.

- Although you were asked to perform cluster analysis in Task 3, there is no need to mention what you did there because the cluster labels variable is not used in the final model. Unlike the PCA in the June 2019 PA exam, the cluster analysis did not replace any variables when constructing the GLMs in Tasks 5 and 7.

- The second to last paragraph of the model solution describes what we did in Task 9, something specific to this project.

## December 2018 Exam PA

This is the very first PA exam offered, with a completely different structure. Instead of giving you a set of well-defined tasks with concrete guidance to work on, this exam adopts a very open-ended format. Candidates were required to write a long report with four parts:

1. Executive Summary
2. Data Exploration and Feature Selection
3. Model Selection and Validation
4. Findings and Recommendations

The open-ended exam format, the little guidance provided for each part, and the existence of several challenging data issues together make this exam project very challenging to most candidates.

Unlike the June 2019 PA exam and the Hospital Readmissions sample project, this exam project tests both GLMs and decision trees. The objective is to construct two predictive models to explain the number of injuries per 2,000 employee hours, a numeric target variable, using a variety of variables that relate to different characteristics of a given mine. These variables include both categorical and numeric variables. Such models will help the mine workers union, our client, identify key determinants of mine safety.

**Data cleaning.** Compared to the June 2019 PA exam and the Hospital Readmissions sample project, the December 2018 PA exam involves extensive data cleaning (as demonstrated in the model solution), but provides very little guidance and sample code to help you. Some data cleaning steps require a good understanding of what the variables really mean in this particular context (e.g., associating a low total number of employees hours, `EMP_HRS_TOTAL`, with the possibility of mine closure, which can be checked with the mine status variable, `MINE_STATUS`). It is expected that exams under the current format will provide much clearer guidance on how data cleaning should be performed, but the December 2018 exam does illustrate some instructive data cleaning techniques. The most notable changes are described below.

- *Missing data:* The Rmd template has provided code to remove observations with a missing value. There are a total of 27 such observations. Instead of using the `is.na()` function, you can also use the `complete.cases()` function, as we learned in Section 1.3.

```
cc <- complete.cases(data.all)
data.nomissing <- data.all[cc, ]
nrow(data.all) - nrow(data.nomissing)

## [1] 27
```

- *Deletion of high-dimensional factor variables:* Two of the factor variables, `PRIMARY` and `US_STATE`, have an extraordinarily large number of levels (79 for `PRIMARY` and 55 for `US_STATE`).

PRIMARY is noted in the Rmd template and the exam expects you to make the same observation for US_STATE. The presence of so many possible classes can increase the dimension of the data disproportionately and dilute the predictive power of the models to be constructed. To simplify matters, the model solution drops the two variables altogether. Alternatively, you can binarize the two factor variables, run a PCA, and see how much variation the first few PCs can capture, very much like Task 3 of the June 2019 PA exam. This way, we can avoid losing too much potentially useful information for prediction. Yes, there are many alternative data cleaning steps for this exam!

- *Retaining only "functioning" mines:* Since the mine workers union specifically would like us to analyze "functioning" mines (the term "functioning mines" appears only once in the project statement!), only `"Active"`, `"Full-time permanent"`, and `"Intermittent"` mines are retained. These three categories are taken from the MINE_STATUS variable. With the benefit of hindsight, this data cleaning step may seem obvious, but not so much in an exam environment, especially when you are given so little guidance to follow.

  The model solution uses a `for` loop to delete observations with non-functioning mines. Alternatively, you can use logical subsetting with a composite logical test to do the extraction.

```
data.reduced2 <- data.reduced
data.reduced2 <-
  data.reduced2[data.reduced2$MINE_STATUS == "Active" |
                data.reduced2$MINE_STATUS == "Full-time permanent" |
                data.reduced2$MINE_STATUS == "Intermittent", ]
```

- *Deleting observations with extreme, unrealistic values:* The target variable in this exam project is the injury rate per 2,000 employee hours, a variable we have to manually create:

$$\texttt{INJ\_RATE\_PER2K} = \frac{\texttt{NUM\_INJURIES}}{\texttt{EMP\_HRS\_TOTAL}/2,000}.$$

  Even with only functioning mines retained, there are still a substantial number of mines with an unreasonably high injury rate due to an unusually low number of employee hours, which can be confirmed by a histogram. The model solution makes the bold move of removing all observations with fewer than 2,000 employee hours. The cutoff of 2,000 employee hours is equivalent to 1 full-time employee year, but is otherwise rather arbitrary. A total of 12,013 observations are deleted as a result.

```
nrow(data.reduced2) - nrow(data.reduced3)

## [1] 12013
```

- *Mine status:* The project statement mentions that coal miners appear to use different terminology in mine status than others. This issue needs to be explicitly dealt with. Using a two-way classification table, we can see how MINE_STATUS lines up with COMMODITY:

```
table(data.reduced3$MINE_STATUS, data.reduced3$COMMODITY)

##
##                         Coal Metal Nonmetal Sand & gravel Stone
##   Active                3423     0        0             0     0
##   Closed by MSHA           0     0        0             0     0
##   Full-time permanent      0   686     2107          7817 10319
##   Intermittent             0   203      465          7376  3724
##   Non-producing            0     0        0             0     0
##   Permanently abandoned    0     0        0             0     0
##   Temporarily closed       0     0        0             0     0
```

That there are no closed, non-producing, permanently abandoned, and temporarily closed mines is a result of the retention of only "functioning" mines. It appears that what coal mines mean by "Active" is similar to or the same as what other mines mean by "Full-time permanent". Again, there are different ways to resolve this terminology issue. The method adopted by the model solution is to adjust the mine status variable and combine "Active" and "Full-time permanent" as one level called "Open".

```
data.reduced3$ADJ_STATUS[data.reduced3$MINE_STATUS == "Active"] <- "Open"
data.reduced3$ADJ_STATUS[data.reduced3$MINE_STATUS ==
                           "Full-time permanent"] <- "Open"
data.reduced3$ADJ_STATUS[data.reduced3$MINE_STATUS ==
                           "Intermittent"] <- "Intermittent"
data.reduced3$ADJ_STATUS <- as.factor(data.reduced3$ADJ_STATUS)
data.reduced3$MINE_STATUS <- NULL
table(data.reduced3$ADJ_STATUS, data.reduced3$COMMODITY)

##
##                Coal Metal Nonmetal Sand & gravel Stone
##   Intermittent     0   203      465          7376  3724
##   Open          3423   686     2107          7817 10319
```

Coding-wise, the model solution follows the good practice of saving the data frame as a new data frame every time a major change is made. This allows us to preserve a pristine copy of the old data frame in case any data cleaning steps are done incorrectly. Table 6.4 keeps track of the five data frames created.

The model solution has a big chunk exploring the distribution of the injury rate. To make it easier for graders to look at your work, you can make a separate chunk for each graph you produce with comments.

**Data exploration.** As soon as data cleaning is completed, the data exploration part is pretty straightforward, even though no code is provided. Since the target variable, injury rate, is a numeric variable, split box plots (for categorical predictors) and scatterplots (for numeric predictors) are useful graphical displays for visualizing the relationship between different variables and injury rate.

| Data Frame | Number of Observations | Property |
|---|---|---|
| `data.all` | 53,746 | Original file |
| `data.nomissing` | 53,719 | `data.all` with missing values dropped |
| `data.reduced` | 53,719 | `data.nomissing` with `PRIMARY` and `US_STATE` dropped |
| `data.reduced2` | 48,133 | `data.reduced` with only functioning mines retained |
| `data.reduced3` | 36,120 | `data.reduced2` with observations with unreasonably low values of `EMP_HRS_TOTAL` removed |

Table 6.4: The five data frames created for the December 2018 PA exam project.

The model solution censors the plots and removes outliers (i.e., those with an injury rate higher than 0.5) to enhance visual differences.

You may be tempted to apply the log transformation to the injury rate to deal with its pronounced right skew. However, the large number of observations with zero injury rate makes the log transformation not directly applicable (unless you add a constant to each observation to bring the injury rate strictly positive).

**Decision trees.** The first type of predictive model tested in this exam project is base decision trees. The training and test sets should be created prior to any fitting and used consistently across all models to ensure a fair comparison.

- *Content:* The exam expects that you construct multiple (at least two, for full credits) decision trees, describe the tree construction methodology, and evaluate their predictive performance. The model solution builds several trees with different specifications of the control parameters to achieve different levels of complexity. (The first tree, not discussed here, uses injury rate to predict injury rate.)

  ▷ *Preliminary tree (*`tree.reduced.pruned`*):* This tree is constructed with a minimum complexity parameter of zero and pruned using the complexity parameter that corresponds to the smallest cross-validation error. Even with pruning, however, the tree is far too complex for practical use.

  ▷ *Tree 1:* This tree is motivated from the first tree and constructed using a minimum complexity parameter of 0.05 to reduce tree complexity. It turns out that this tree has only one split and is too simple.

  ▷ *Tree 2:* This tree, built with a minimum complexity parameter of 0.0005, allows for a sufficiently large number of splits while pre-pruning splits that are not worthwhile to pursue. When the optimal complexity parameter is used, the resulting tree has 12 splits and is still pretty complex.

  ▷ *Tree 3:* The last tree is based on the `cptable` of Tree 2 and adopts a complexity parameter that results in a much simpler tree with comparable cross-validation error. The SOA's choice of the fourth value of `cp`, corresponding to 3 splits, is rather ad hoc. If you use the one-standard-error rule introduced in Section 5.3 of this manual, you will look for the simplest tree whose cross-validation error is below $0.9138860 + 0.01573993 = 0.92962593$. The third value of cp, corresponding to 2 spits, will be chosen.

In addition to explaining how you produce these trees, you should take the initiative to interpret the output of the constructed trees. The following items, taken from what we learned in Section 5.3, are worth mentioning:

▷ How many splits does a tree have? Provided that there are not too many splits, describe sequentially how the splits are made.

▷ Which variables are the most important predictors as shown in the first few splits?

▷ Are the findings intuitive? This is where you can tie your results to the underlying business problem.

▷ Which variables appear to have interactions?

The different trees can be ranked by their test set loglikelihood using the `LLfunction()` function defined in the Rmd template (also see Problem 1.5.2). In this project, this validation metric is preferred to other metrics such as RMSE due to the skewed nature of the injury rate. The SOA chooses Tree 3 after balancing both prediction accuracy and interpretability. You can also make a case for Tree 2.

- *Coding:* Despite the unfamiliar setting involving a Poisson distribution for the target and an offset term, no special knowledge is required and code is given for you to construct and prune a decision tree and calculate its test set loglikelihood without material changes (if you are interested, run `?rpart` and look at the `method` argument). The only adjustments required are to remove the injury rate from the model formula, change `data.reduced` to the training set in the `data` argument of the `rpart()` function and to the test set in the `newdata` argument of the `predict()` function and in the first argument of the `LLfunction()` function.

**GLMs.**

- *Content:* The second type of predictive model tested is Poisson GLMs. If you simply modify the given code and fit a Poisson GLM, you will see the following "unusual" output (note the message "`(1 not defined because of singularities)`" and the warning "`prediction from a rank-deficient fit may be misleading`" towards the end of the output):

```
glm.reduced <- glm(NUM_INJURIES ~ . - EMP_HRS_TOTAL - INJ_RATE_PER2K,
                   family = poisson(),
                   offset = log(EMP_HRS_TOTAL/2000),
                   data = train)
summary(glm.reduced)

##
## Call:
## glm(formula = NUM_INJURIES ~ . - EMP_HRS_TOTAL - INJ_RATE_PER2K,
##     family = poisson(), data = train, offset = log(EMP_HRS_TOTAL/2000))
##
```

```
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -7.3938  -0.6300  -0.3933  -0.2493   9.1127
##
## Coefficients: (1 not defined because of singularities)
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 1.656e+07  7.208e+06   2.298  0.02155 *
## COMMODITYMetal             -1.510e-01  2.999e-02  -5.037 4.73e-07 ***
## COMMODITYNonmetal          -2.082e-01  3.621e-02  -5.750 8.93e-09 ***
## COMMODITYSand & gravel      3.069e-01  5.265e-02   5.830 5.53e-09 ***
## COMMODITYStone              2.478e-02  3.076e-02   0.805  0.42059
## SEAM_HEIGHT                -4.801e-04  8.215e-05  -5.844 5.08e-09 ***
## TYPE_OF_MINESand & gravel          NA         NA      NA       NA
## TYPE_OF_MINESurface         2.320e-01  3.793e-02   6.116 9.62e-10 ***
## TYPE_OF_MINEUnderground    -2.400e-01  7.349e-02  -3.266  0.00109 **
## AVG_EMP_TOTAL              -1.687e-04  2.695e-05  -6.258 3.91e-10 ***
## PCT_HRS_UNDERGROUND        -1.656e+07  7.208e+06  -2.298  0.02155 *
## PCT_HRS_SURFACE            -1.656e+07  7.208e+06  -2.298  0.02155 *
## PCT_HRS_STRIP              -1.656e+07  7.208e+06  -2.298  0.02155 *
## PCT_HRS_AUGER              -1.656e+07  7.208e+06  -2.298  0.02155 *
## PCT_HRS_CULM_BANK          -1.656e+07  7.208e+06  -2.298  0.02155 *
## PCT_HRS_DREDGE             -1.656e+07  7.208e+06  -2.298  0.02155 *
## PCT_HRS_OTHER_SURFACE      -1.656e+07  7.208e+06  -2.298  0.02155 *
## PCT_HRS_SHOP_YARD          -1.656e+07  7.208e+06  -2.298  0.02155 *
## PCT_HRS_MILL_PREP          -1.656e+07  7.208e+06  -2.298  0.02155 *
## PCT_HRS_OFFICE             -1.656e+07  7.208e+06  -2.298  0.02155 *
## ADJ_STATUSOpen             -9.383e-02  3.978e-02  -2.358  0.01835 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 27251  on 27089  degrees of freedom
## Residual deviance: 24508  on 27070  degrees of freedom
## AIC: 38965
##
## Number of Fisher Scoring iterations: 6
```

```
glm.predict <- predict(glm.reduced, newdata = test, type = "response")
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type
== :  prediction from a rank-deficient fit may be misleading
```

```
# For GLM with an offset, this predict function includes the effect of the
off-set, producing the number of injuries.
print("loglikelihood")
```

```
## [1] "loglikelihood"

LLfunction(test$NUM_INJURIES, glm.predict)

## [1] 854.8607
```

Two rather tricky data issues have to be resolved:

1. There are NAs for the "Sand & gravel" level of TYPE_OF_MINES. This happens because "Sand & gravel" co-exist in COMMODITY and TYPE_OF_MINES.

   ```
   table(data.reduced3$COMMODITY, data.reduced3$TYPE_OF_MINE)

   ##
   ##                  Mill Sand & gravel Surface Underground
   ##   Coal           1004             0    1500         919
   ##   Metal           130             0     514         245
   ##   Nonmetal        386             0    2040         146
   ##   Sand & gravel     0         15193       0           0
   ##   Stone           355             0   13250         438
   ```

   When the GLM is fitted, two identical dummy variables for "Sand & gravel", one for COMMODITY and one for TYPE_OF_MINES, are produced, leading to perfect collinearity. One solution, as adopted by the model solution, is to combine the two variables using the paste() function and leave out the "Sand & gravel" level as the baseline level using the relevel() function given in the Rmd template.

   ```
   data.reduced4 <- data.reduced3
   data.reduced4$MINE_CHAR <- paste(data.reduced4$TYPE_OF_MINE,
                                    data.reduced4$COMMODITY)
   data.reduced4$MINE_CHAR <- relevel(as.factor(data.reduced4$MINE_CHAR),
                                       ref = "Sand & gravel Sand & gravel")

   table(data.reduced4$MINE_CHAR)

   ##
   ## Sand & gravel Sand & gravel                   Mill Coal
   ##                       15193                        1004
   ##                  Mill Metal               Mill Nonmetal
   ##                         130                         386
   ##                  Mill Stone                Surface Coal
   ##                         355                        1500
   ##               Surface Metal            Surface Nonmetal
   ##                         514                        2040
   ##               Surface Stone            Underground Coal
   ##                       13250                         919
   ##           Underground Metal        Underground Nonmetal
   ##                         245                         146
   ```

```
##          Underground Stone
##                        438
```

You can then drop the COMMODITY and TYPE_OF_MINE variables, or take the extra care to drop these two variables in the formula argument of the glm() function, as the model solution does. In any case, we have to re-partition the data into the training and test sets to reflect the new MINE_CHAR variable.

The model solution comments that some candidates removed either COMMODITY or TYPE_OF_MINES entirely. This is not preferred as this would result in a loss of potentially predictive information.

2. As the project statement says, all the PCT_HRS_### variables add up to 1, leading to another source of collinearity.

```
# Extract the PCT_HRS_### variables
# Calculate their sum for each observation
head(apply(data.reduced4[, 5:14], 1, sum))
```

```
## 1 2 3 4 5 6
## 1 1 1 1 1 1
```

Any of these variables can be dropped to solve the problem, but the SOA drops PCT_HRS_STRIP due to it having the greatest association with the "Sand & gravel Sand & gravel" level of the MINE_CHAR variable. This can be checked by calculating the mean of each PCT_HRS_### variable within those observations belonging to the "Sand & gravel Sand & gravel" level.

As soon as you resolve these two data issues, you can fit the Poisson GLM, simplify it using stepwise selection, and evaluate its test set loglikelihood. The SOA also makes two notable additions:

1. *Log-transforming* AVG_EMP_TOTAL*:* To improve the model fit in the presence of skewed variables like AVG_EMP_TOTAL, the log-transformed version is used.
2. *Interaction variables:* Tree 3 constructed earlier indicates that there may be interactions between AVG_EMP_TOTAL, PCT_HRS_STRIP, and PCT_HRS_UNDERGROUND. These interaction variables can be inserted into the GLM to improve its predictive power. It turns out that the interaction variables indeed are statistically significant and lead to a non-trivial amount of improvement in prediction accuracy.

- *Coding:* The Rmd template has code for you to run a Poisson GLM using log(EMP_HRS_TOTAL/2000) as an offset. There are certain changes you have to make and new code to write:

  1. Adjust the data argument of the glm() function and the newdata arguments of the predict() function so that the model is fitted to the training set and predictions are generated on the test set.
  2. Adjust the formula argument of the glm() function to leave out unwanted variables and put in desired interaction variables.
  3. Run the stepAIC() function from the MASS package to perform stepwise selection.

**Findings and recommendations.** The findings part of this exam project resembles the interpretation task (Task 8) of the June 2019 PA exam and the Hospital Readmissions sample project. The same approach can be taken: translate the coefficient estimates into precise statements on the effect of the variables on the injury rate (due to the log link, $e^{\hat{\beta}_j}$ gives the multiplicative change in the baseline injury rate) and comment on whether these statements are reasonable. Due to the large number of features in the final model, you may choose to interpret only the statistically significant features.

The recommendations part is specific to this exam project and more difficult to write. The model solution suggests further steps to leverage the two variables deleted at the beginning of the project, `PRIMARY` and `US_STATE`, based on the mine workers union's subject matter expertise. A cluster analysis or PCA can help with that analysis.

**NOTE**

Two original practice exams are expected to be released in late February/early March and late April. Stay tuned!