Springboard Capstone Project

Bryon Kent

April 02, 2017

Introduction

One of the largest epidemics sweeping the Veterans community is suicide, with roughly 22 people a day committing suicide accoridng to some estimates. This is due to a number of often-times interrelated causes. I set-out to create a model that predicts the risk level for Veteran suicide in a given community so that ones with a higher risk profile could then be honed-in on for specialized programs to increase awareness and to ensure that the proper resources are put in place to help eradicate this phenomena. Given that the Department of Defense (DoD) closeley protects Peronsally Identifiable Information (PII) and Personal Health Infomation (PHI), it was not possible to obtain the necessary data from the DoD to accomplish this directly. As such, it was necessary to develop a risk-profile using the population writ-large with available data from the U.S. Government. This script explores the mortality data in the United States for *(the year 2014)* based on the data released by the Centers for Disease Control and Prevention. This script looks looks into the frequency of suicide cases for the different factors such as age, sex, race, education, and marital status. Following this analysis, additional data will be gathered in order to build a master model. This master model will then be used to explore various predicitve modeling techniques such as GLM, Decision Tree, KSVM, and randomForest.

Pre-Processing

```
#calling libraries

library(dplyr)
library(ggplot2)
# library(acs)
# library(party)
# library(e1071)
# library(MASS)
# library(ROCR)
# library(randomForest)
```

Data Loading

```
DeathRec <- read.csv("/Users/Athena/Documents/My-Github-Repository/DeathRecords.csv", stringsAsFactors =
Race <- read.csv("/Users/Athena/Documents/My-Github-Repository/Race.csv", stringsAsFactors = F, header =
AgeType <- read.csv("/Users/Athena/Documents/My-Github-Repository/AgeType.csv", stringsAsFactors = F, he
Edu2003 <- read.csv("/Users/Athena/Documents/My-Github-Repository/Education2003Revision.csv", stringsAs
Marital_table <- read.csv("/Users/Athena/Documents/My-Github-Repository/MaritalStatus.csv", stringsAsFa
```

Data Exploration

In the following sections we will explore the data behind reported suicide statistics for 2014, such as age, sex, education, and marital status.

Extracting and filtering Suicide Cases

we will extract the suicide case entries from the total death record by filtering MannerOfDeath == 2:

```
Suicide <- DeathRec %>%
         filter(MannerOfDeath == 2 )
```

The column AgeType specifies the units of age as follows:

```
##   Code    Description
## 1    1          Years
## 2    2         Months
## 3    4           Days
## 4    5          Hours
## 5    6        Minutes
## 6    9 Age not stated
```

For the extracted suicide cases we have the following AgeTypes:

```
table(Suicide$AgeType)
```

```
##
##     1     9
## 43132     7
```

There are 9 entries with no-stated age, so they will be excluded in order to avoid bias in the results:

```
Suicide <- Suicide %>%
         filter(AgeType!= 9)
```

Distribution of Suicide Cases Over Age

We will now examine the number of suicide cases, mean age, and standard deviation for both males and females:

```
SuicideSex <- Suicide %>%
            group_by(Sex) %>%
            summarise(Cases=n(),
                      Percentage = 100*n()/length(.$Id),
                      Mean = mean(Age), Std = sd(Age))

SuicideSex
```

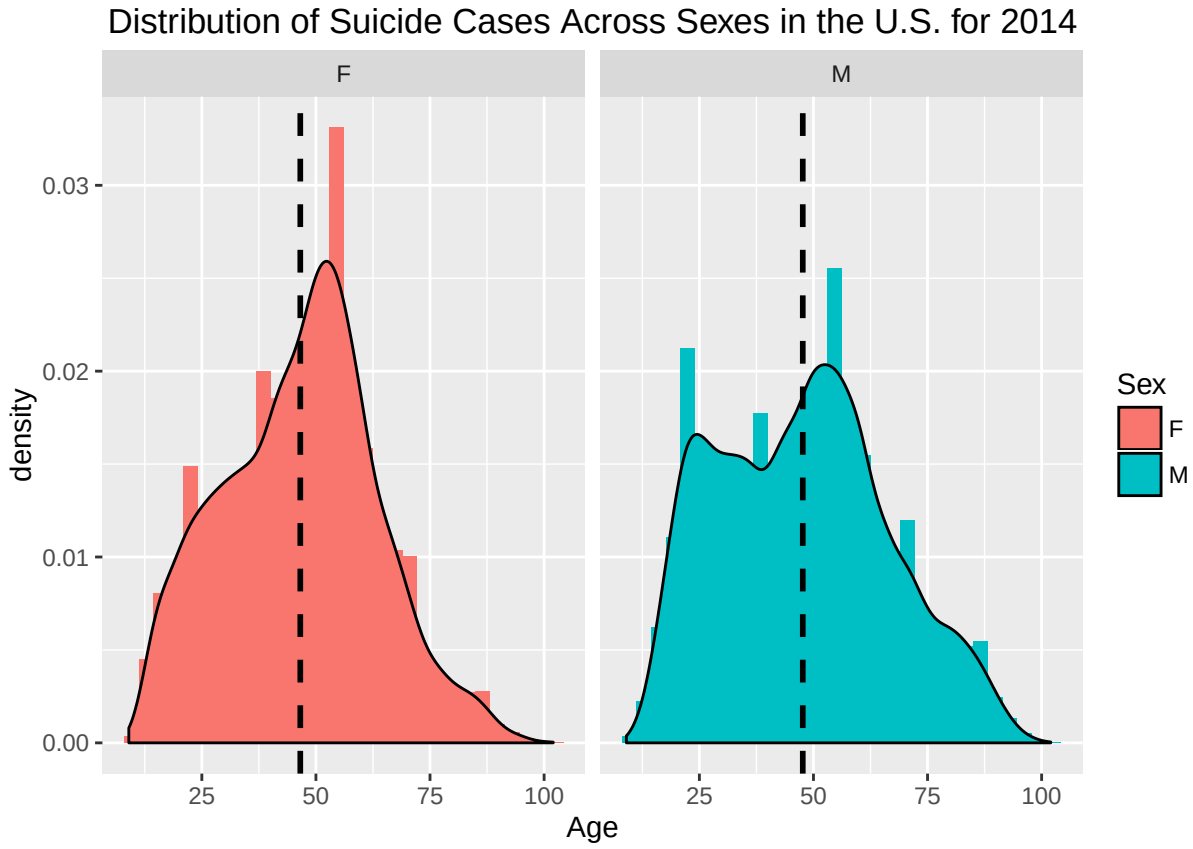
```
## # A tibble: 2 x 5
##     Sex Cases Percentage     Mean      Std
##   <chr> <int>      <dbl>    <dbl>    <dbl>
## 1     F  9775   22.66299 46.57483 16.78307
## 2     M 33357   77.33701 47.65932 18.85484
```

From this, we can see that the mean age for both groups is approximately the same; however, the overwhelming majority of reported cases are male. In order to get a little better fidelity on these inital observations, we will look at the distribution and the mean for both males and females:

```
# group data by Sex and calculate the mean

SuicideDist <- Suicide %>%
             group_by(Sex) %>%
             mutate(Mean = mean(Age))
```

```
ggplot(SuicideDist, aes(x = Age, y = ..density.., fill = Sex)) +
             geom_histogram() +
             geom_density() +
             geom_vline(aes(xintercept = Mean),
                        color = "black", linetype = "dashed", size = 1) +
             facet_grid(.~Sex) +
             labs(title = "Distribution of Suicide Cases Across Sexes in the U.S. for 2014")
```



Distribution of Suicides Cases Across Races

We will now examine the number of suicide cases, mean age, and standard deviation across reported races:

```
SuicideRace <- Suicide %>%
             group_by(Race) %>%
             summarise(Cases = n(),
                       Percentage = 100*n()/length(.$Id),
                       Mean = mean(Age), Std = sd(Age))

SuicideRace
```

```
## # A tibble: 14 x 5
##     Race Cases  Percentage     Mean      Std
##    <int> <int>       <dbl>    <dbl>    <dbl>
## 1      1 38983 90.38069183 48.27907 18.32820
## 2      2  2449  5.67791895 38.43895 16.70289
## 3      3   491  1.13836595 34.86558 15.10745
```

```
## 4      4   218  0.50542521 46.16514 19.29519
## 5      5    76  0.17620328 56.09211 19.31575
## 6      6     6  0.01391079 34.00000 24.45404
## 7      7   139  0.32226653 38.66187 16.47420
## 8     18   149  0.34545117 40.18792 16.73133
## 9     28   189  0.43818974 47.94180 18.42558
## 10    38    15  0.03477696 36.13333 16.85597
## 11    48   101  0.23416489 41.37624 16.90376
## 12    58     9  0.02086618 31.11111 13.75177
## 13    68   225  0.52165446 38.27556 16.69005
## 14    78    82  0.19011407 38.47561 14.39670
```

From this, we can see that the mean age for all race groups ranges between 31-56, with the overwhelming majority of the cases reported falling into category 1 (White).

Percentage of Suicide Cases Per Education Level

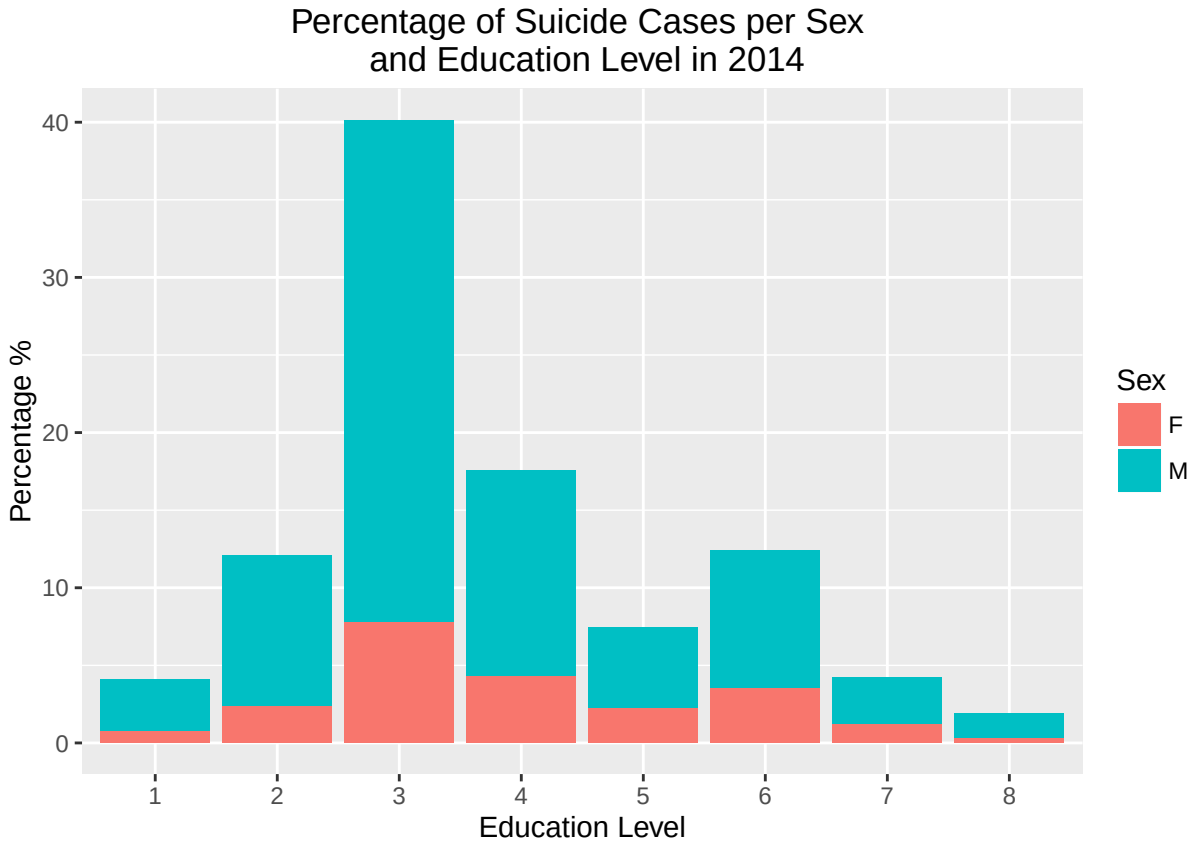
We would now like to see if there are any patterns that emerge related to the level of education obtained and suicide. The education categories within the data are as follows:

```
##   Code                           Description
## 1    1                      8th grade or less
## 2    2            9 - 12th grade, no diploma
## 3    3 high school graduate or GED completed
## 4    4    some college credit, but no degree
## 5    5                       Associate degree
## 6    6                      Bachelor's degree
## 7    7                        Master's degree
## 8    8      Doctorate or professional degree
## 9    9                                Unknown
```

From our intial examination of the *EducationRevision2003.csv*, we noted that there are two categories that do not have a defined education level *(0,9)*, so they will be filtered out to remove noise from the data set:

```
ByEdu <- Suicide %>%
        filter(Education2003Revision!= 0) %>%
        filter(Education2003Revision!= 9) %>%
        group_by(Education2003Revision, Sex) %>%
        summarise(Sum = n(), Percentage = 100*n()/length(.$Id))

ggplot(ByEdu, aes(x = factor(Education2003Revision), y = Percentage, fill = Sex)) +
        geom_bar(stat = "identity") +
        labs(title = "Percentage of Suicide Cases per Sex \n and Education Level in 2014") +
        labs(x = "Education Level", y = "Percentage %")
```



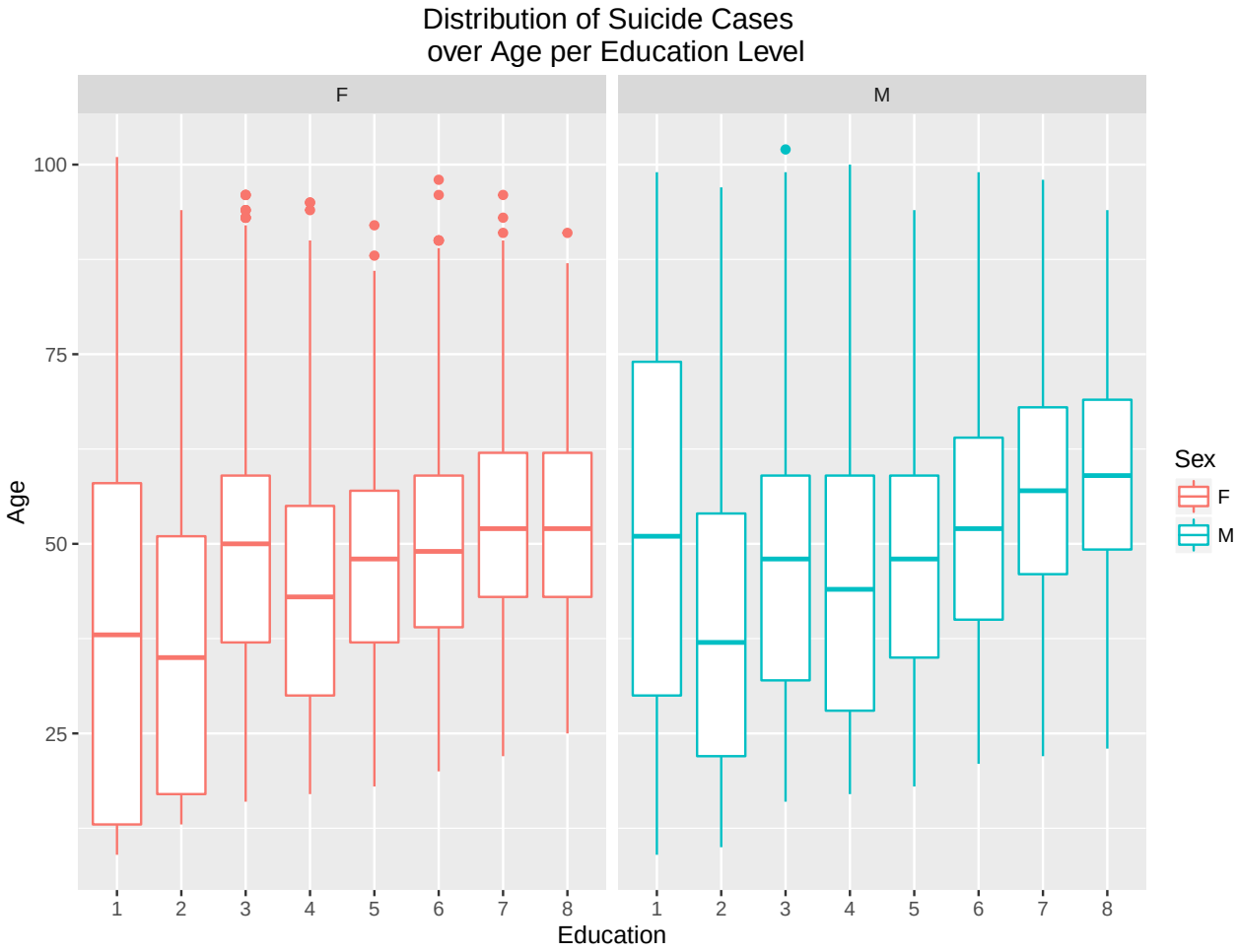
From the graph, we can see that the highest percentage of reported suicides across both sexes fell within category 3 (high school graduate or GED completed) and the lowest percentage of reported suicides across both sexes fell within category 8 (Doctorate or professional degree).

Distribution of Suicide Cases Over Age per Education Level

We will now look at the mean and the range of ages within each educational level by plotting a box plot:

```
ByEdu2 <- Suicide %>%
        filter(Education2003Revision!= 0) %>%
        filter(Education2003Revision!= 9) %>%
        group_by(Education2003Revision, Age, Sex)

ggplot(ByEdu2, aes(y = Age, x = factor(Education2003Revision))) +
        geom_boxplot(aes(color = Sex)) +
        facet_grid(.~Sex) +
        labs(title = "Distribution of Suicide Cases \n over Age per Education Level") +
        labs (x = "Education", y = "Age")
```



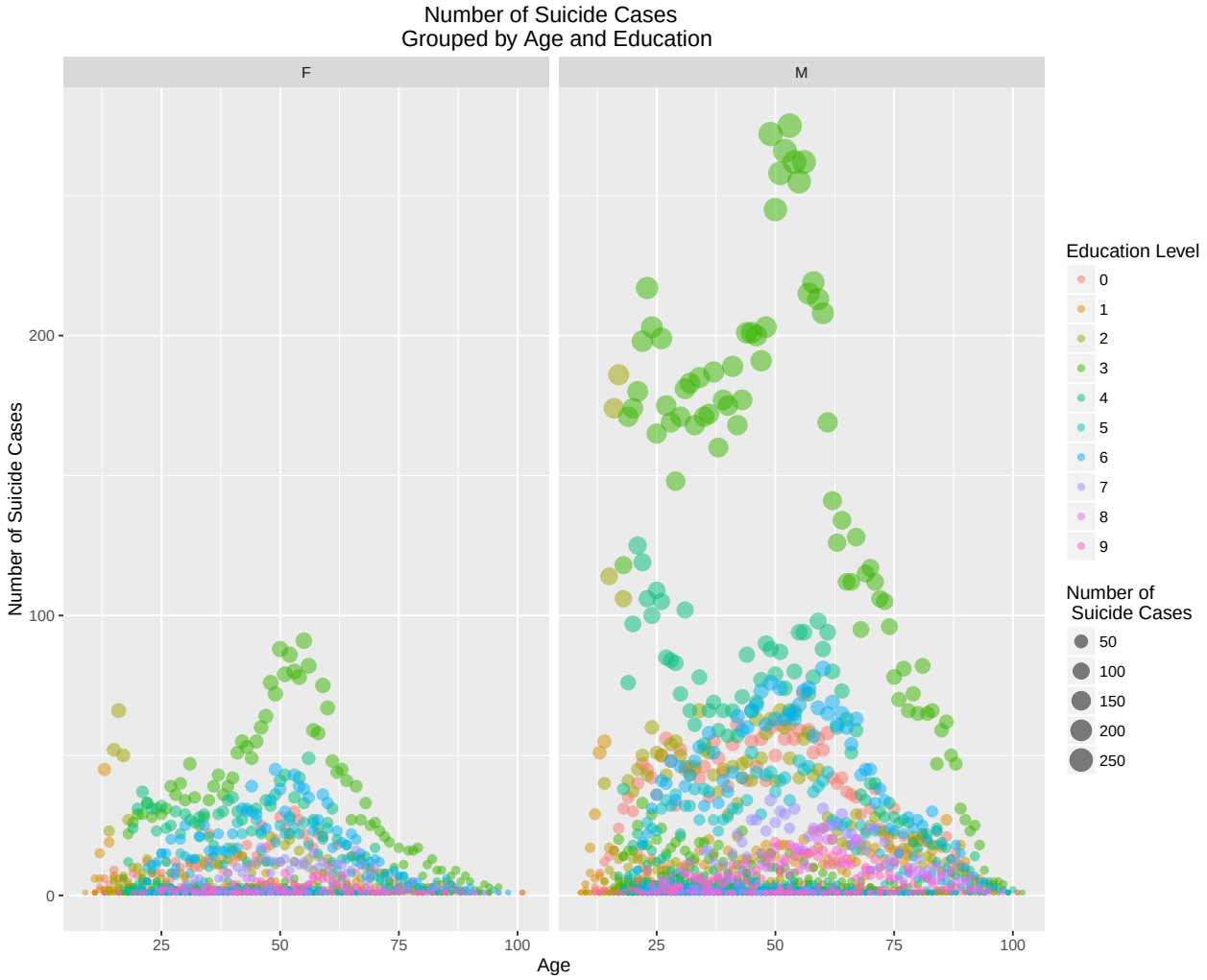
We can see from this that the common age range across the reported education categories falls somewhere in the range of 45-55.

Number of Suicide Cases Grouped by Age, Sex, Race, and Education

We can now look even further into the data to see the number of suicide cases for each age, grouped by the education level, sex, and race:

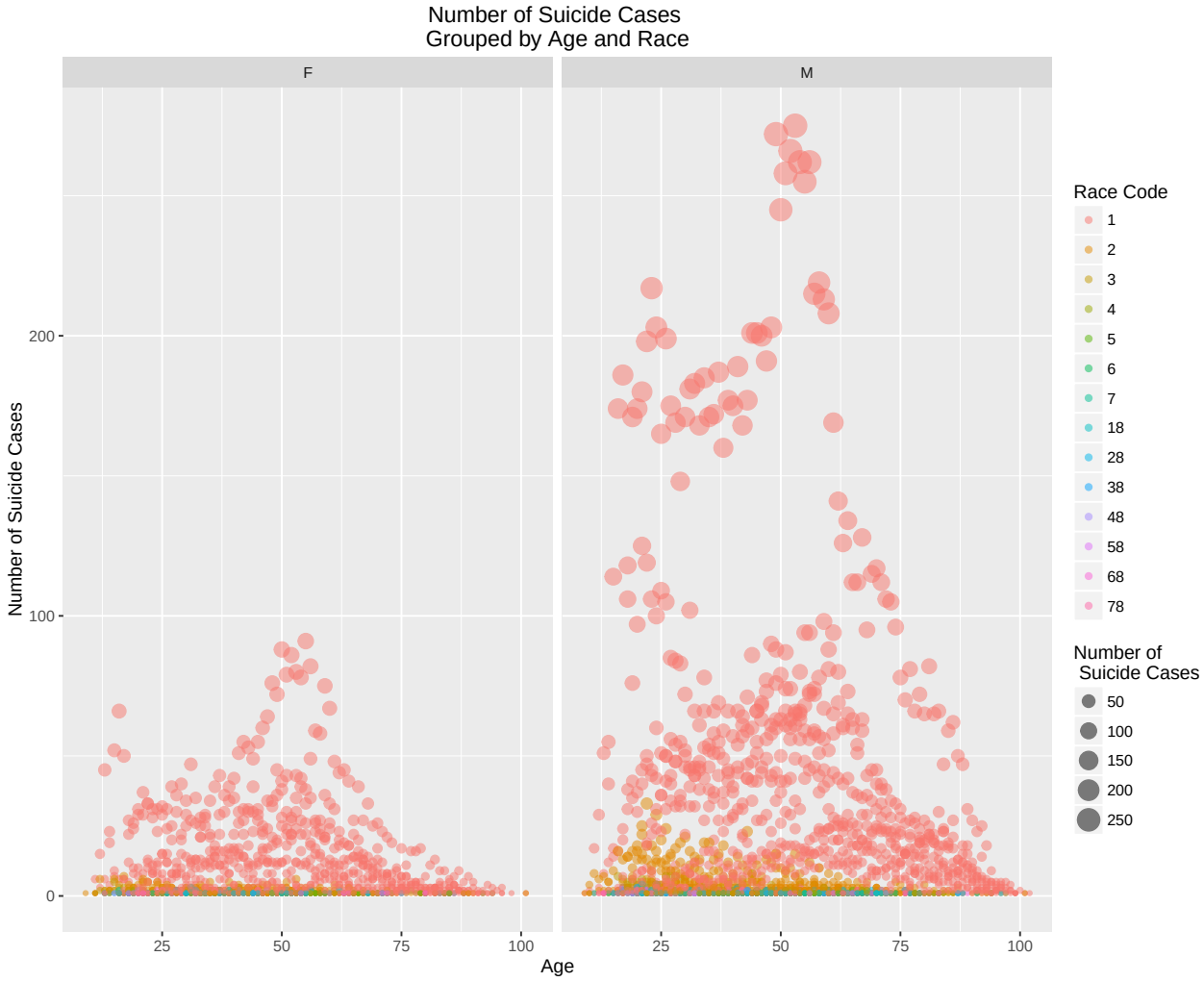
```
ByEdu3 <- Suicide %>%
        group_by(Education2003Revision, Sex, Age, Race) %>%
        summarise(Sum = n())

ggplot(ByEdu3, aes(y = Sum, x = Age)) +
        geom_point(aes(color = factor(Education2003Revision), size = Sum), alpha = 0.5) +
        facet_grid(.~Sex) +
        scale_y_continuous(name = "Number of Suicide Cases") +
        scale_colour_discrete("Education Level") +
        scale_size_continuous("Number of \n Suicide Cases", breaks = seq(0, 300, by = 50)) +
        labs(title = "Number of Suicide Cases \n Grouped by Age and Education")
```



From the graph, above, we can confirm that the median age of suicide across both male and females is between 45-55, and that the highest incidence rate occurs in those individuals that come from an educational background of high school graduate or GED completed.

```
ggplot(ByEdu3, aes(y = Sum, x = Age)) +
        geom_point(aes(color = factor(Race), size = Sum), alpha = 0.5) +
        facet_grid(.~Sex) +
        scale_y_continuous(name = "Number of Suicide Cases") +
        scale_colour_discrete("Race Code") +
        scale_size_continuous("Number of \n Suicide Cases", breaks = seq(0, 300, by = 50)) +
        labs(title = "Number of Suicide Cases \n Grouped by Age and Race ")
```



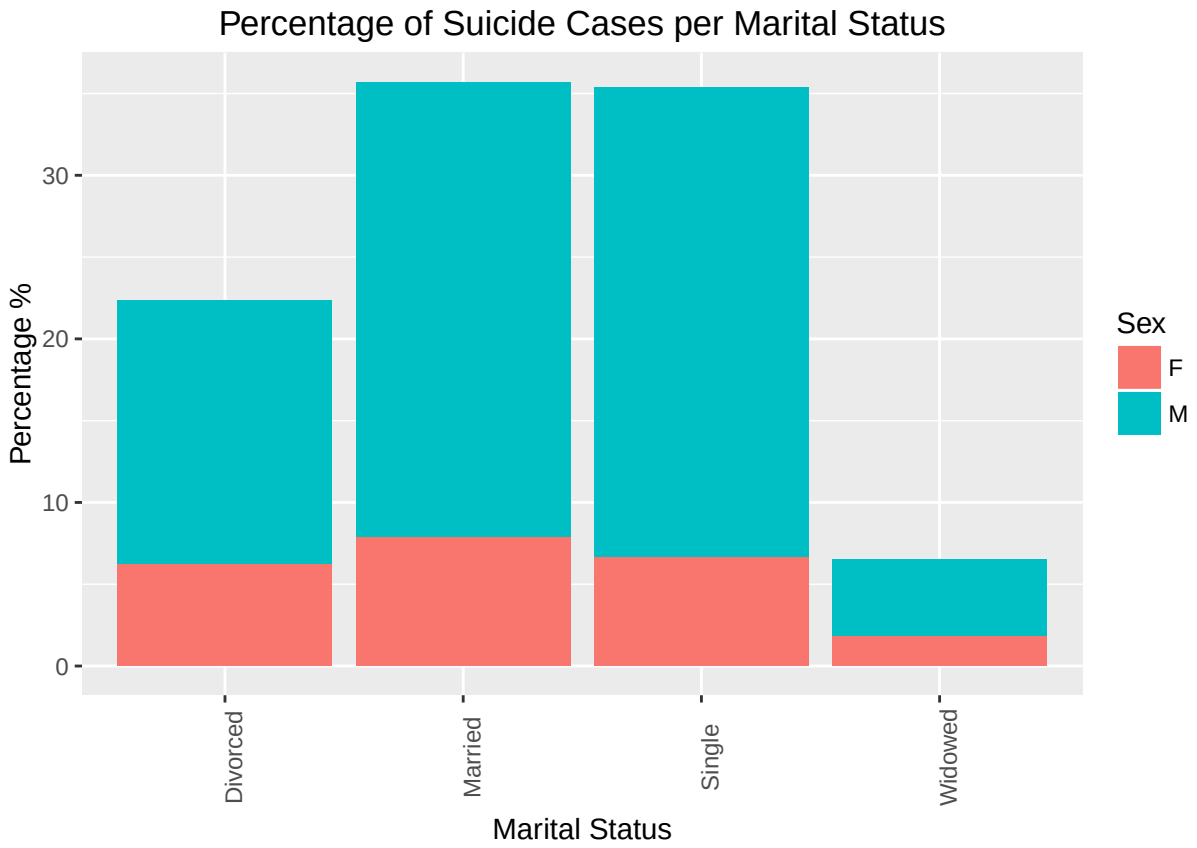
This grpah confirms that the vast majority (~90%) of all reported suicide cases, for both females and males comes from category 1 (White).

Percentage of Suicide Cases Per Marital Status

We will now examine the interplay between reported suicide cases and marital status. We know from our initial observation of the data that there is an unknown category for marital status, so it will be filtered out to remove noise in the data:

```
ByMarital <- Suicide %>% filter(MaritalStatus!= "U") %>%
        group_by(MaritalStatus, Sex) %>%
        summarise(Cases_Suicide = n(), Percentage_Suicide = 100*n()/length(.$Id))

ggplot(ByMarital, aes(x = MaritalStatus, y = Percentage_Suicide, fill = Sex)) +
        geom_bar(stat = "identity") +
        scale_x_discrete(name = "Marital Status",
                         labels = c("Divorced", "Married", "Single",
                                    "Widowed")) +
        theme(axis.text.x = element_text(angle = 90, vjust = 1)) +
        scale_y_continuous(name = "Percentage %")  +
        labs(title = "Percentage of Suicide Cases per Marital Status")
```



We can see from the above graph that the singles category represents the highest percentage in males, but not significantly higher than those that are married. In the case of females, the percantage of suicide is largest in the married category, but not significantly higher than those females that are divorced or single.

Initial Thoughts

We have seen from the data, above that there is a much highr incidents rate of suicide amongst males *(ca. 77.3%)* compared to females *(ca. 22.7%)*, with a mean age for both males and females of ca. 47. There is a dispropotionately high percentage of reported cases that fall into the low/mid-level education range. Additionally, we observed that there was a significant percentage of suicide cases within the singles category *(ca. 35%)*, which is significant given the relatively small percentage that singles represent in the overall population *(ca. 12%)*. This data needs to be tracked across many years to see whether or not there is a trend in the data worth exploring further.

Predicitve Modeling Analysis

Now that we have examined the data from the Centers for Disease Control and Prevention, we will use a few different modelling techniques with to see if we are able to develop a predcitive tool that assists in suicide prevention. A lot of the available data had to be excluded as there was not enough additional informaiton to make it useful. Its important to understand the age standardization of death rates as it ties into how the master data set was developed to build our predicitive models. The new standard is based on the year 2000 population, replacing the existing standard based on the 1940 population. Prior to this new standard, at least three different standards were used among Department of Health and Human Services agencies. Implementation of the year 2000 standard reduced confusion among data users and the burden on State and local agencies. Use of the year 2000 standard also resulted in age-adjusted death rates that are substantially

larger than those based on the 1940 standard. Further, the new standard affects trends in age-adjusted death rates for certain causes of death and narrows race differentials in age-adjusted death rates. Although age standardization is an important and useful tool, it has some limitations. As a result the examination of age-adjusted death rates should be the beginning of an analysis strategy.

Build the Base Model

We started the construction of the base model by pulling the (filtered in some cases) available data for age, education, race, marital status, and sex.

```
Model <- Suicide %>%
        filter(Education2003Revision!= 0) %>%
        filter(Education2003Revision!= 9) %>%
        filter(MaritalStatus!= "U") %>%
        group_by(Age, Education2003Revision,
                Race, MaritalStatus, Sex) %>%
        summarise(Cases = n(), Percentage = 100*n()/length(.$Id))

Model
```

```
## Source: local data frame [5,583 x 7]
## Groups: Age, Education2003Revision, Race, MaritalStatus [?]
##
##      Age Education2003Revision  Race MaritalStatus   Sex Cases  Percentage
##    <int>                 <int> <int>         <chr> <chr> <int>       <dbl>
## 1      9                     1     1             S     M     1 0.002624052
## 2      9                     1     2             S     F     1 0.002624052
## 3      9                     1     2             S     M     1 0.002624052
## 4     10                     1     1             S     M     2 0.005248104
## 5     10                     1     2             S     M     4 0.010496208
## 6     10                     1     3             S     M     1 0.002624052
## 7     10                     2     2             S     M     1 0.002624052
## 8     11                     1     1             S     F     6 0.015744312
## 9     11                     1     1             S     M    17 0.044608885
## 10    11                     1     2             S     F     1 0.002624052
## # ... with 5,573 more rows
```

```
write.csv(Model, file = "Model.csv")
```

There were several other pieces of information that needed to be gathered before the Model was complete. The next thing we gathered was data about years of school. This was used to determine the percentage of the total population that had completed a certain level of education, which would be used as a multipler later in the construction of the model:

```
# years_of_school (contained in a separate .xlsx file, attached)
```

Next, we installed the acs library and obtained a key from the census bureau. This was used to gather the needed data from 2014 for the number and percentages of age, education, race, marital status, and sex amongst the national population, which was combined with the years of school data from before:

```
# api.key.install(key = "CONFIDENTIAL")

# acs.fetch(endyear = 2014, span = 5,
#           geography = geo.make("*"), table.number = "B01003",
#           dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
```

```
#          geography = geo.make("*"), table.number = "B01001",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B01001A",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B01001B",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B01001C",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B01001D",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B01001E",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B02001",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B02006",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B12002A",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B12002B",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B12002C",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B12002D",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B12002E",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B14007A",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B14007B",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B14007C",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B14007D",
#          dataset = "acs", col.names = "pretty")
# acs.fetch(endyear = 2014, span = 5,
#          geography = geo.make("*"), table.number = "B14007E",
#          dataset = "acs", col.names = "pretty")
```

Now that we have obtained all of the information needed to build base model, I built a feeder data.set that determined the various percentage levels for each of the variables. Before we could use this data.set, however, the data needed to be wrangled. For example, the age ranges had to be binned according to available census data (ie. 15-17, 18-19, etc). While condensing these rows, we had to exclude all sucides with an age value less than 15 due to the fact that there was no available marriage data that went below this value. We also had to condense the Race variables into less specifc subgroups (ie. White, Black, American Indian (includes Aleuts and Eskimos), Asian, and Native hawaiian and other pacific islander). This information was fed into a formula to generate the number of people within the general population that were a part of the same sub dataset as the particular row. For example, if a row had the following values for Age, Education, Race, MaritalStatus, and Sex (15-17, 1, 1, D, F), then the formula would be *(Living_in_Population = Total_Population_in_Age_Bucket_Based_on_Sex x Percetange_of_Race_in_Age_Bucket x Percentage_of_Martital_Status_in_Age_Bucket_Based_on_Sex_and_Race x Percentage_of_Education_Level_in_Age_Bucket)*. The output of this formula became Cases2 in the model below, with Cases representing reported suicides. While the description is short, the process to get to this point was an extremely lenghty and laborious, to say the least. With all of this information in the base model, we needed to expand the data.set to account for each value of Cases and Cases2 in order to use the model with our predictive tools. The code used to do this is contained, below (The data.frame created is massive, so I have opted just display the code, without allowing it to run):

```
# master <- Model2

# row <- nrow(master)
# new <- c("Age", "Education2003Revision", "Race",
#         "MaritalStatus", "Sex", "Cases", "Cases2")

# new <- master[1,]

# Endpoint <- "Endpoint"
# Endpoint <- as.data.frame(Endpoint)
# colnames(Endpoint) <- c("Endpoint")
# new <- cbind(new, Endpoint)
# master <- cbind(master, Endpoint)

# colnames(new) <- c("Age", "Education2003Revision", "Race",
#                    "MaritalStatus", "Sex", "Cases", "Cases2", "Endpoint")
# na.omit(new)

# master <- as.data.frame(master)
# master[,8] <- sapply(master[,8], as.integer)
# new <- master[1,]

# counter = 0
# for (i in 1:row){

#  Cases <- master[i,6]
#  Cases2  <- master[i,7]
#  print(c("Cases", Cases, "Cases2", Cases2))
#  for (p in 1:Cases){
#    counter = counter + 1
#    master[i,8] <- 0
#    new <- rbind(new, master[i,])
#    new[counter, 8] <- 1
#  }
```

```
#  for (t in 1:Cases2){
#    counter = counter + 1
#    master[i, 8] <- 0
#    new <- rbind(new, master[i,])
#    new[counter, 8] <- 0
#  }

# }
```

Due to the sheer size of the resulting data.frame, using RStudio on my computer alone was not enough to complete the Master Model in a reasonable time-frame. It took over two weeks to develop roughly 10% of the Master Model data.frame. In order to develop a sufficently large enough subset of the data, eight virtual machines were built in microsoft azure to produce subset Master Model data.frames of the Base Model data.frame across 8 virtual machines, each runnig 4 instances of r a piece. Each of these subset Master Model data.frames were read r:

```
# newXX <- read_csv("~/Desktop/DSP New Files/newXX.csv",
#                   col_types = cols(Sex = col_character(),
#                   X1 = col_skip()))
```

all newXX files (new1-new33) produced in the azure step, above, were read into r and then combined together to create the Master Model data.frame:

```
# new_sample <- rbind(new1, new2, new3, new4, new5, new6, new7, new8, new9, new10, new11,
#                     new12, new13, new14, new15, new16, new17, new18, new19, new20, new21,
#                     new22, new23, new24, new25, new26, new27, new28, new29, new30, new31,
#                     new32, new33)

# new_sample2 <- cbind(new_sample$Age, new_sample$Education2003Revision,
#                     new_sample$Race, new_sample$MaritalStatus, new_sample$Sex,
#                     new_sample$Endpoint)

# colnames(new_sample2) <- c("Age", "Education2003Revision", "Race",
#                            "MaritalStatus", "Sex", "Endpoint")
```

Create random training, validation, and test data sets

Now that the Master Model data.frame is complete, we need to separate it into a training, validation, and test data sets:

```
# Set some input variables to define the splitting.

# Input 1. The data frame that you want to split into training, validation, and test:

# df <- new_sample2

# Input 2. Set the fractions of the dataframe you want to split into training,
# validation, and test:

# fractionTraining <- 0.60
# fractionValidation <- 0.20
# fractionTest <- 0.20

# Compute sample sizes:
```

```
# sampleSizeTraining <- floor(fractionTraining * nrow(df))
# sampleSizeValidation <- floor(fractionValidation * nrow(df))
# sampleSizeTest <- floor(fractionTest * nrow(df))

# Create the randomly-sampled indices for the dataframe. Use setdiff() to
# avoid overlapping subsets of indices:

# indicesTraining <- sort(sample(seq_len(nrow(df)), size=sampleSizeTraining))
# indicesNotTraining <- setdiff(seq_len(nrow(df)), indicesTraining)
# indicesValidation <- sort(sample(indicesNotTraining, size=sampleSizeValidation))
# indicesTest <- setdiff(indicesNotTraining, indicesValidation)

# Finally, output the three dataframes for training, validation and test:

# dfTraining <- df[indicesTraining, ]
# dfValidation <- df[indicesValidation, ]
# dfTest <- df[indicesTest, ]
```

Due to the size of the dfTraining data.frame, r continued to crash while trying to build and evaluate the various predicitive models. So, I deceided to use the dfTest and dfValidation data.frames instead since they contained 0.5 the amount of data of dfTraining. While this worked, it should be noted as this may have had an affect on the overall efficacy of the predicive models.

GLM Model

```
# attach(dfTest)
# attach(dfValidation)

# new_regression_b1 <- glm(
#        Endpoint ~ Age + Education2003Revision + Race + MaritalStatus + Sex, family = binomial())
#        new_regression_b1

# summary(new_regression_b1)

# Predict Output

# predictions_1 <- predict(new_regression_b1, newdata = dfTest, type = "response")
# mse_1 <- mean((dfTest$Endpoint - predicitons_1)^2)
# print(mse_1)
# [1] 0.0003360676

# Validate Input

# predicitons_2 <- predict(new_regression_b1, newdata = dfValidation, type = "response")
# mse_2 <- mean((dfValidation$Endpoint - predicitons_2)^2)
# print(mse_2)
# [1] 0.0003328199

# Input data for determining AUC and ROC Curve

# prob <- predict(new_regression_b1, newdata = data, type ="response")
# pred <- prediction(prob, data$Endpoint)
```

```
# perf <- performance(pred, measure = "tpr", x.measure = "fpr")

# Calculations to determine AUC and Plot ROC Curve

# auc <- performance(pred, measure = "auc")
# auc <- auc@y.values[[1]]

# roc.data <- data.frame(fpr = unlist(perf@x.values),
#                        tpr = unlist(perf@y.values),
#                        model = "GLM")

# ggplot(roc.data, aes(x = fpr, ymin = 0, ymax = tpr)) +
#     geom_ribbon(alpha = 0.2) +
#     geom_line(aes(y = tpr)) +
#     ggtitle(paste0("ROC Curve w/ AUC=", auc))

# Rplot07

# print(auc)
# [1] 0.8316389995
```

We can see in the GLM Model, above, that the MSE is very low for both the test and validation data.frames, indicating a high degree of significance, and an AUC value of 0.83164, which tells us that this is a reasonably good model.

Decision Tree Model

```
# data <- dfTest
# data_2 <- dfValidation

# t <- ctree(
#        Endpoint ~ Age + Education2003Revision +
#        Race + MaritalStatus + Sex, data,
#        controls = ctree_control(
#        teststat ="quad",
#        testtype = "Univariate",
#        mincriterion = .95,
#        minsplit = 10,
#        minbucket = 5,
#        maxdepth = 0
#    )
#  )
# plot(t)
# Rplot08

# Predict Output:

# predictions_3 <- predict(t, data)
# predictions_3 <- as.data.frame(predictions_3)
# mse_3 <- mean((data$Endpoint - predictions_3)^2)
# print(mse_3)
# [1] 0.0003327041
```

```
# Validate Output:

# predictions_4 <- predict(t,data_2)
# predictions_4 <- as.data.frame(predictions_4)
# mse_4 <- mean((data_2$Endpoint - predictions_4)^2)
# print(mse_4)
# [1] 0.0003299487

# Confusion Matrix to determine if overfit:

# predictions_4_1 <- predict(t, data_2)
# predictions_4_1 <- as.data.frame(predictions_4_1)
# confusionMatrix(data_2$Endpoint, predictions_4_1)

# Input data for determining AUC and ROC Curve:

# prob_2 <- predict(t, newdata = data, type ="response")
# pred_2 <- prediction(prob_2, data$Endpoint)
# perf_2 <- performance(pred_2, measure = "tpr", x.measure = "fpr")

# Calculations to determine AUC and Plot ROC Curve:

# auc_2 <- performance(pred_2, measure = "auc")
# auc_2 <- auc_2@y.values[[1]]

# roc.data_2 <- data.frame(fpr = unlist(perf_2@x.values),
#                          tpr = unlist(perf_2@y.values),
#                          model = "ctree")

# ggplot(roc.data_2, aes(x = fpr, ymin = 0, ymax = tpr)) +
#     geom_ribbon(alpha = 0.2) +
#     geom_line(aes(y = tpr)) +
#     ggtitle(paste0("ROC Curve w/ AUC=", auc_2))

# Rplot09

# Print(auc_2)
# [1] 0.8935005411
```

We can see in the Decision Tree Model, above, that the MSE is very low for both the test and validation data.frames, indicating a high degree of significance, and an AUC value of 0.89350, which tells us that this is a reasonably good model, and a better fit than GLM Model.

KSVM Model

The dfTest data.frame was too large for the KSVM Model. So, I took a random sample of 10,000 rows to create a df_subset data.frame to train and evaluate this model:

```
# attach(df)

# create a subset of the larger data.frame to work with in R:

# df_sample <- sample(nrow(df), size = 10000, replace = TRUE)
# df_subset <- df[df_sample,]
```

```
# Divide dfTest data to x (contains all the features) and y only the classes:

# x <- subset(df_subset, select = -Endpoint)
# y <- df_subset$Endpoint

# Create the SVM model and show the summary:

# m <- svm(formula = Endpoint ~ ., data=df_subset)
# summary(m)

# Predict Output (dfTest might be x in this case):

# predictions_5 <- predict(m, x)
# mse_5 <- mean((y - predictions_5)^2)
# print(mse_5)
# [1] 0.008890168

# Examine the confusion matrix results using the command table to compare the results of the SVM
# prediction and the class data in the y variable:

# table(predictions_5, y)

# Tune the SVM to find the best cost and gamma:

# svm_tune <- tune(svm, train.x = x, train.y = y,
#                  Kernel = "radial", ranges = list(cost = 10^(-1:2),
#                                                   gamma = c(.5, 1, 2)))
# print(svm_tune)

# After you find the best cost and gamma, you can create svm model again and try to run again. The * is
# a place holder for the cost and gamma shown in the printed svm_tune:

# svm_model_after_tune <- svm(Endpoint~ ., data = df_subset,
#                             kernel = "radial", cost = 1, gamma = 0.03703704)
# summary(svm_model_after_tune)

# Run predictions again with the new model:

# predictions_5_1 <- predict(svm_model_after_tune, x)

# Examine the confusion matrix result of prediction, using command table to compare the result of SVM p

# table(predictions_5_1, y)

# Validate Output (I chose to use the whole df in this case due to the smaller training set used to
# build the model):

# predictions_6 <- predict(svm_model_after_tune, df)
# mse_6 <- mean((df$Endpoint - predictions_6)^2)
# print(mse_6)
# mse_6 = 0.008517287

# Input data for determining AUC and ROC Curve:
```

```
# prob_3 <- predict(svm_model_after_tune, newdata = df_subset, typ e= "response")
# pred_3 <- prediction(prob_3, y)
# perf_3 <- performance(pred_3, measure = "tpr", x.measure = "fpr")

# Calculations to determine AUC and Plot ROC Curve:

# auc_3 <- performance(pred_3, measure = "auc")
# auc_3 <- auc_3@y.values[[1]]

# roc.data_3 <- data.frame(fpr = unlist(perf_3@x.values),
#                          tpr = unlist(perf_3@y.values),
#                          model = "svm")
# ggplot(roc.data_3, aes(x = fpr, ymin = 0, ymax = tpr)) +
#     geom_ribbon(alpha = 0.2) +
#     geom_line(aes(y = tpr)) +
#     ggtitle(paste0("ROC Curve w/ AUC=", auc_3))

# Rplot010

# print(auc_3)
# [1] 0.9832366
```

We can see in the KSVM Model, above, that the MSE is very low for both the df_subset and df$Endpoint data.frames, indicating a high degree of significance, and an AUC value of 0.98324, which tells us that this model is overfit and as such, should be disgarded.

randomForest Model

The dfTest data.frame was too large for the randomForest Model. So, I took a random sample of 10,000 rows to create a df_subset data.frame to train and evaluate this model:

```
# create a subset of the larger data.frame to work with in R:

# df_sample_2 <- sample(nrow(df), size = 10000, replace = TRUE)
# df_subset_2 <- df[df_sample,]

# Fitting model:

# fit_RF <- randomForest(
#       Endpoint ~ Age + Education2003Revision +
#       Race + MaritalStatus + Sex,
#       data = df_subset_2
# )

# summary(fit_RF)

#Predict Output:

# predictions_7 <- predict(fit_RF,df_subset_2)
# mse_7 <- mean((df_subset_2$Endpoint - predictions_7)^2)
# print(mse_7)
# [1] 0.0007220855
```

```
#Validate Output (I chose to use the whole df in this case due to the smaller training set used to buil

# predictions_8 <- predict(fit_RF, df)
# mse_8 <- mean((df$Endpoint - predictions_8)^2)
# print(mse_8)
# [1] 0.0003319365

# Input data for determining AUC and ROC Curve:

# prob_4 <- predict(fit_RF, newdata = df_subset_2, type = "response")
# pred_4 <- prediction(prob_4, df_subset_2$Endpoint)
# perf_4 <- performance(pred_4, measure = "tpr", x.measure = "fpr")

# Calculations to determine AUC and Plot ROC Curve:

# auc_4 <- performance(pred_4, measure = "auc")
# auc_4 <- auc_4@y.values[[1]]

# roc.data_4 <- data.frame(fpr = unlist(perf_4@x.values),
#                          tpr = unlist(perf_4@y.values),
#                          model = "randomForest")

# ggplot(roc.data_4, aes(x = fpr, ymin = 0, ymax = tpr)) +
#     geom_ribbon(alpha = 0.2) +
#     geom_line(aes(y = tpr)) +
#     ggtitle(paste0("ROC Curve w/ AUC=", auc_4))

# Rplot11

# print(auc_4)
# [1] 0.993157
```

We can see in the randomForest Model, above, that the MSE is very low for both the df_subset and df$Endpoint data.frames, indicating a high degree of significance, and an AUC value of 0.99316, which tells us that this model is overfit and as such, should be disgarded.

Final Thoughts

Over the course of our evaluation of sucicides, occuring in 2014, we developed a clear understanding of the risk factors that may have contributed to these people's choice to commite suicide. While we did not prove a direct causal link during the course of this investigation, anecdotal evidence supports these conclusions. Pulling together all of the information needed to complete this project was an arduous endeavor to say the least: it's a difficult topic to work with day after day. One of the high-points during this investigation was the oppotunity to collaborate with some Korean Data Scientists that were also studying suicide and related risk factors. They took a novel approach to the examination of sucicide were able to succesfully incorporated social media data into their work–something that I plan to do down the road as I continue to work on this project–which I believe is key to moving beyond the current reactionary paradigm and into one that is more preemptive in nature. The next steps in this evolving project will be to:

- Seek to take this generalized abstraction of suicides within the U.S. population and tie it to instances of Veteran suicides
- examine social media data such a twitter of people that have committed suicide to see if there were any warning signs present, then seek to genralize those findings with the aim to incoporate it into our model

- Examine hospital data, such as access to care, average wait-times, and available care offered in order to see if there are additional risk factors that can be incorporated into the model. This will also necessitate the examination of certain mental health factors as well

END