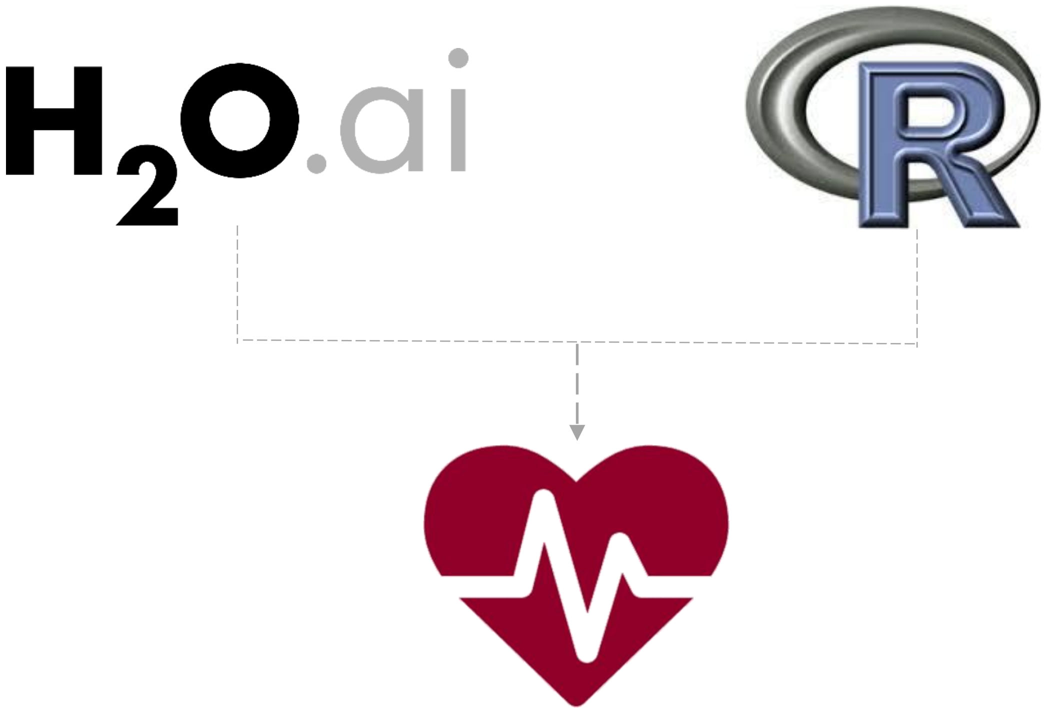
Colaberry **Powered by: Data Science Center of Excellence**

Analyzing Diabetes with H2O & R (feat. Plotly)

By Josh W. Smitherman - Chief Data Scientist
JW@colaberry.com



Better Healthcare Insights

About the Author

Josh W. Smitherman is Chief Data Scientist at Colaberry Inc. where he leads the data science practice of integrated advanced analytical capabilities into client's and organization's value chain processes. Josh holds a Masters of Operations Research from Southern Methodist University in Dallas Texas, Master Certificate in Applied Statistics from Penn State, and a B.B.A in Management Information Systems from Texas Tech University. Certified Lean Six Sigma Black Belt, where he has implemented continuous improvement and analytical processes within many organizations. He has worked on data science projects over the last 10 years in retail, CPG, healthcare (clinical trails and patient outcome health), consumer and auto financial services, oil and gas, and supply chain. He has worked with companies like Wal-Mat, GE, Dr. Pepper Snapple Group, Pergo, Mondelez, MyHealth Access Network, CBRE, Costco, Lowe's, Home Depot, State Farm, UK healthcare networks, and many others.

Contents

1. Introduction
- What is R? What is H2O.ai and how they work together…………………. 4
- Review of NHANES dataset……………………………………………….. 5
- Diabetes analytics approach using H2O.ai + R…………………………. 6

2. Expository data analytics (EDA) feat. Plotly
- EDA with Plotly…………………………………………………………… 7
- H2O imputation for missing values……………………………………… 11
- Summary of EDA………………………………………………………… 21

3. Clustering using H2O+R
- Clustering NHANES - diabetes analysis……………………………….. 22
- Determine the appropriate number of clusters……………………….. 23
- Run kmeans and review cluster sizes…………………………………. 25
- Clustering and diabetes………………………………………………… 28

4. Predicting diabetes using H2O+R
- Predictive models overview in H2O…………………………………….. 31
- Running Distributed Random Forest by cluster……………………….. 32
- Running Gradient Boosting Machine by cluster……………………….. 35
- Running General Linear Model by cluster…………………………….. 37
- Running Deep Learning by cluster…………………………………….. 39
- Bringing it all together…………………………………………………… 41

5. Summary / Conclusion
- What did we learn/do?.. 46
- Review of H2O capabilities……………………………………………... 46
- Future enhancements & recommendations…………………………… 46

1. Introduction

What is R?

R is a widely used (and free) software environment for statistical computing and graphics. Many academic institutions and organizations contribute to the wide array of packages that enable R to quickly gain the latest and greatest machine learning, statistical, and/or graphical capabilities. When connected with RStudio (an R integrated development environment or IDE) helps users to quickly develop and deploy analytical results.

If you are not familiar with these tools, access the links below for more information:

https://www.r-project.org/

https://www.rstudio.com/

What is H2O.ai?

H2O.ai is an in memory engine that allows you to run R code (and others languages such as Python, Java, etc.) in a distributed fashion. This helps when you need to run large or higher computational algorithms for quicker and deeper analytics verses a single node. H2O.ai has a nice set of pre-built packages for clustering, gradient boosting machine, random forest, deep learning, and others. Another great appeal to using this environment is from a programming API standpoint to process fast computations along with applications for analytics results.

If you are not familiar with these tools, access the links below for more information:

http://www.h2o.ai/

http://www.h2o.ai/resources/

http://learn.h2o.ai/content/index.html

How do R+H2O.ai work together?

As mentioned before, H2O.ai is an in-memory environment where R code is executed, the in-memory computation is optimized to produce scalable data science results from your algorithms and functions. In my own experience, this capability means what took me hours/days now I can do in seconds/minutes without the worry of writing parallelized code. Also, with the many parameters in the well built packages such as DRF or GBM, allows tremendous flexibility in the interactions and parameter setting capabilities due to H2O.ai enhanced scalability features.

Review of NHANES dataset

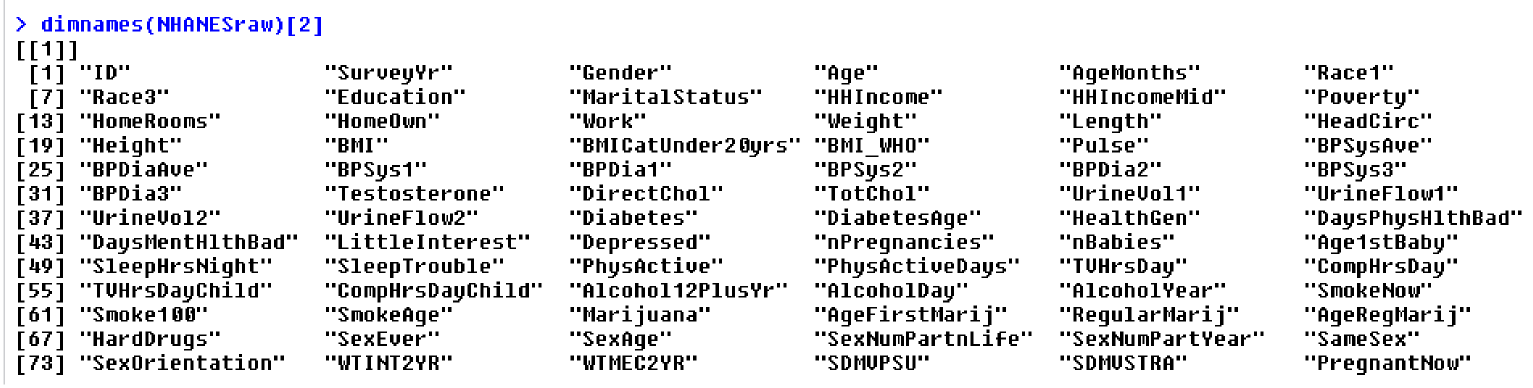
To study the attributes and feature of diabetes we will use the NHANES or National Health and Nutrition Examination Study dataset. According to the NHANES website, the NHANES data represents:

"*The National Health and Nutrition Examination Survey (NHANES) is a program of studies designed to assess the health and nutritional status of adults and children in the United States. The survey is unique in that it combines interviews and physical examinations*."

US National Health and Nutrition Examination Study
http://www.cdc.gov/nchs/nhanes.htm

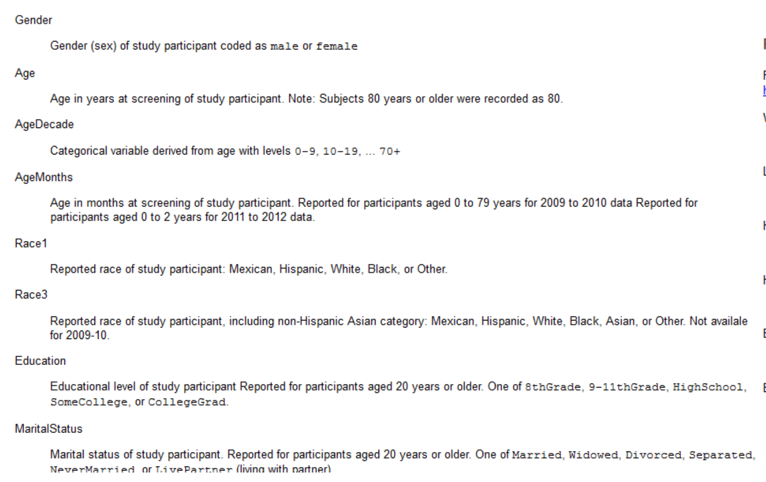
The R implantation of the NHANES dataset is found in the package "NHANES" and the dataset name we will access is called NHANESraw (NHANES 2009-2012 with adjusted weighting). Installing the packages and accessing the library provides the dataset within your R session.

```
install.packages("NHANES")
library(NHANES)
dimnames(NHANESraw)[2]
```



```
> dimnames(NHANESraw)[2]
[[1]]
 [1] "ID"               "SurveyYr"         "Gender"           "Age"              "AgeMonths"        "Race1"
 [7] "Race3"            "Education"        "MaritalStatus"    "HHIncome"         "HHIncomeMid"      "Poverty"
[13] "HomeRooms"        "HomeOwn"          "Work"             "Weight"           "Length"           "HeadCirc"
[19] "Height"           "BMI"              "BMICatUnder20yrs" "BMI_WHO"          "Pulse"            "BPSysAve"
[25] "BPDiaAve"         "BPSys1"           "BPDia1"           "BPSys2"           "BPDia2"           "BPSys3"
[31] "BPDia3"           "Testosterone"     "DirectChol"       "TotChol"          "UrineVol1"        "UrineFlow1"
[37] "UrineVol2"        "UrineFlow2"       "Diabetes"         "DiabetesAge"      "HealthGen"        "DaysPhysHlthBad"
[43] "DaysMentHlthBad"  "LittleInterest"   "Depressed"        "nPregnancies"     "nBabies"          "Age1stBaby"
[49] "SleepHrsNight"    "SleepTrouble"     "PhysActive"       "PhysActiveDays"   "TVHrsDay"         "CompHrsDay"
[55] "TVHrsDayChild"    "CompHrsDayChild"  "Alcohol12PlusYr"  "AlcoholDay"       "AlcoholYear"      "SmokeNow"
[61] "Smoke100"         "SmokeAge"         "Marijuana"        "AgeFirstMarij"    "RegularMarij"     "AgeRegMarij"
[67] "HardDrugs"        "SexEver"          "SexAge"           "SexNumPartnLife"  "SexNumPartYear"   "SameSex"
[73] "SexOrientation"   "WTINT2YR"         "WTMEC2YR"         "SDMVPSU"          "SDMVSTRA"         "PregnantNow"
```

To access the field definitions type ?NHANESraw. Examples below:



Gender

Gender (sex) of study participant coded as `male` or `female`

Age

Age in years at screening of study participant. Note: Subjects 80 years or older were recorded as 80.

AgeDecade

Categorical variable derived from age with levels `0-9`, `10-19`, ... `70+`

AgeMonths

Age in months at screening of study participant. Reported for participants aged 0 to 79 years for 2009 to 2010 data Reported for participants aged 0 to 2 years for 2011 to 2012 data.

Race1

Reported race of study participant: Mexican, Hispanic, White, Black, or Other.

Race3

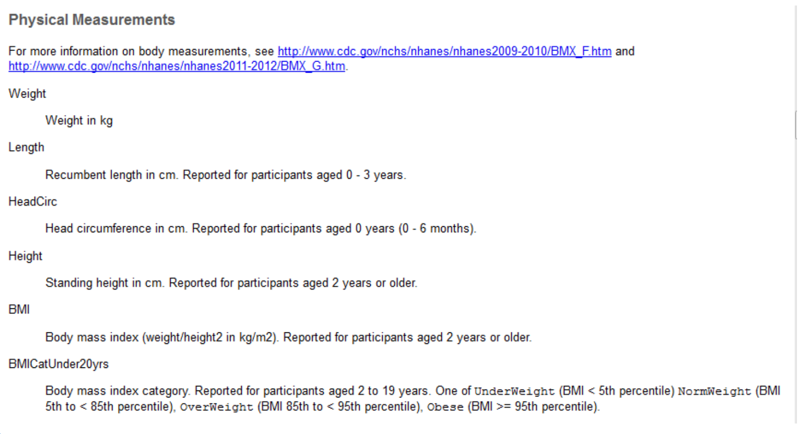
Reported race of study participant, including non-Hispanic Asian category: Mexican, Hispanic, White, Black, Asian, or Other. Not availale for 2009-10.

Education

Educational level of study participant Reported for participants aged 20 years or older. One of `8thGrade`, `9-11thGrade`, `HighSchool`, `SomeCollege`, or `CollegeGrad`.

MaritalStatus

Marital status of study participant. Reported for participants aged 20 years or older. One of `Married`, `Widowed`, `Divorced`, `Separated`, `NeverMarried`, or `LivePartner` (living with partner)



Physical Measurements

For more information on body measurements, see http://www.cdc.gov/nchs/nhanes/nhanes2009-2010/BMX_F.htm and http://www.cdc.gov/nchs/nhanes/nhanes2011-2012/BMX_G.htm.

Weight

Weight in kg

Length

Recumbent length in cm. Reported for participants aged 0 - 3 years.

HeadCirc

Head circumference in cm. Reported for participants aged 0 years (0 - 6 months).

Height

Standing height in cm. Reported for participants aged 2 years or older.

BMI

Body mass index (weight/height2 in kg/m2). Reported for participants aged 2 years or older.

BMICatUnder20yrs

Body mass index category. Reported for participants aged 2 to 19 years. One of `UnderWeight` (BMI < 5th percentile) `NormWeight` (BMI 5th to < 85th percentile), `OverWeight` (BMI 85th to < 95th percentile), `Obese` (BMI >= 95th percentile).

1. Introduction

Diabetes analytics approach using H2O.ai + R

We will use the NHANESraw dataset in drawing insights into the features that help categorize and predict diabetes. Our analytical plan is as follows:

1) Explore the dataset by performing exploratory data analytics (EDA) using statistical tools and graphs (via Plotly).

2) Import the dataset within the H2O.ai environment and perform imputation on missing values and compare those results to the EDA analysis to ensure structural difference are noted.

3) Perform clustering of the numerical features using H2O.ai functions for grouping patients into categories that help in identifying segments that are more prone to diabetes.

4) Perform predictive analytics using H2O.ai functions Gradient Boosting Machine (GBM), Distributed Random Forest (DRF), and Deep Learning on the target variable Diabetes using all the features in the dataset (both imputed and note imputed) to help explain the likelihood that someone could contract the dieses. We will perform accuracy and quality model verification steps as well to ensure model performance is noted along the way.

Installing the needed packages for this demonstration

We will be using R to create the instance of H2O.ai and load medical data from the NHANES data set. To walk through the examples in this document, please ensure you have these packages installed.

```
install.packages(c(“h2o”, “plotly”, “NHANES”,”reshape”,”ggplot2”,”dplyr”))
```

1) H2o
    1) Alternative methods to install h2o
    ```
    install.packages("h2o", repos=(c("http://h2o-release.s3.amazonaws.com/h2o/rel-jacobi/2/R",
    getOption("repos"))))
    ```

    For the latest recommended version, download the latest stable H2O-3 build
    from the H2O download page:
    1. Go to http://h2o.ai/download.
    2. Choose the latest stable H2O-3 build.
    3. Click the \Install in R" tab.
    4. Copy and paste the commands into your R session.

2) plotly
3) NHANES
4) reshape
5) ggplot2
6) dplyr

2. Expository data analytics (EDA) feat. Plotly

Diabetes prevalence within the dataset and by features

We will perform our EDA with analysis on the target variable, Diabetes. We want to know:

- How many cases have diabetes verses those that do not have diabetes? (prevalence measure)
- What are the distributions of key features such as BMI, Cholesterol, Age, etc. for those with diabetes verses those that do not have diabetes?
- What correlations (linear or non- linear relationships) can be detected across the set of features both in relation to Diabetes / None-Diabetes and the overall holistic view?

What is Plotly?

Plotly is an open source framework that enables data scientist to develop interactive, browser-based charting built on JavaScript graphing library, plotly.js. This allows data scientist to use API languages such as R, Python, Java, etc. to build powerful visualizations with interactive capabilities.

We will use the Plotly API for R in this demonstration. To load and use the package in R, reference the below code:

```
install.packages(“plotly")
# or install development version from GitHub
devtools::install_github("ropensci/plotly")
library(plotly)
```

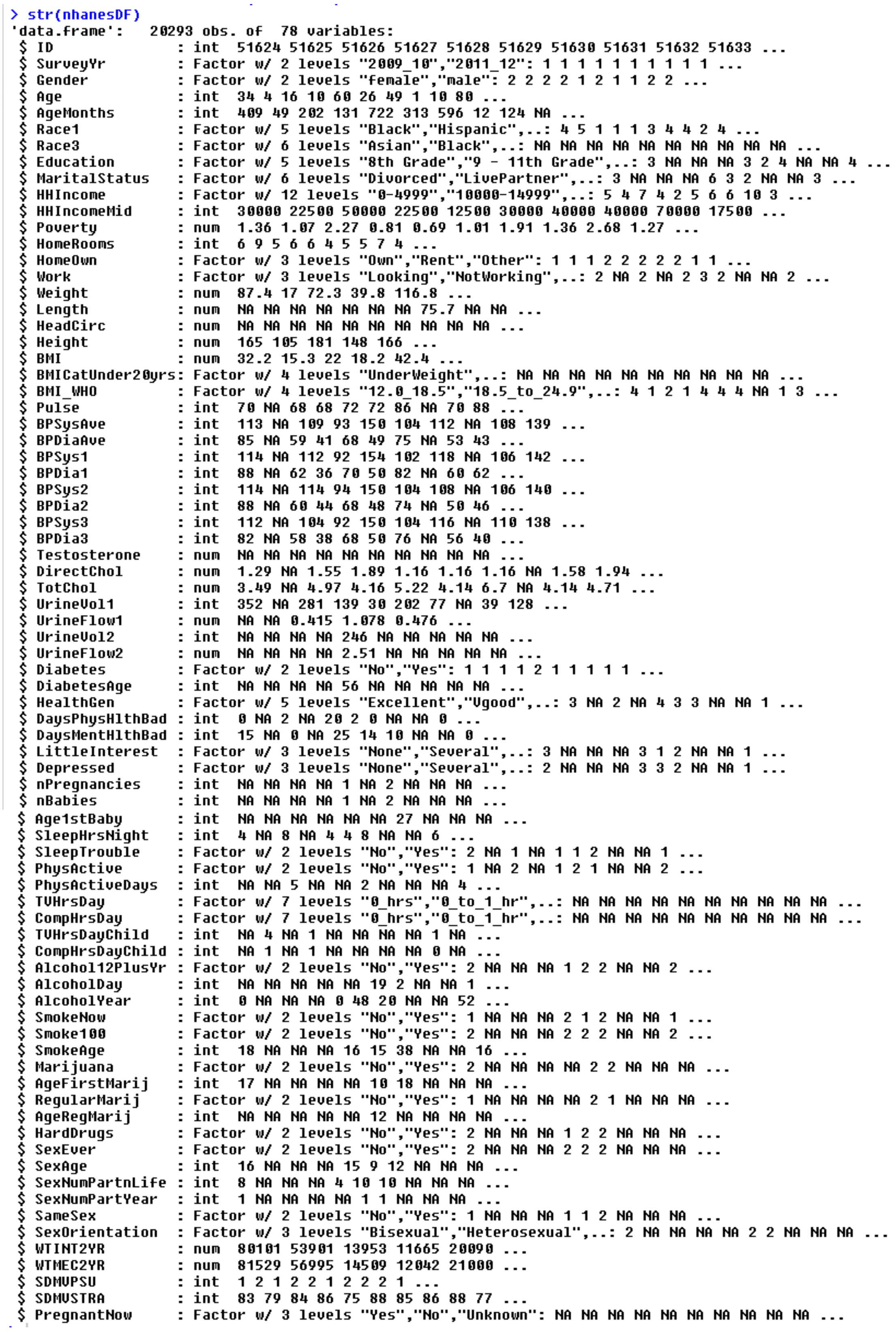
Examples and references by API can be found at the below web link:
https://plot.ly/

NHANES basic summary statistic

Let’s begin with a basic summary of the statistics and structure of the NHANES data set.

```
options(digits=2)

library(NHANES)
## Extract NHANES to DF
nhanesDF <- as.data.frame(NHANESraw)
str(nhanesDF)
summary(nhanesDF$Diabetes)
```

2. Expository data analytics (EDA) feat. Plotly

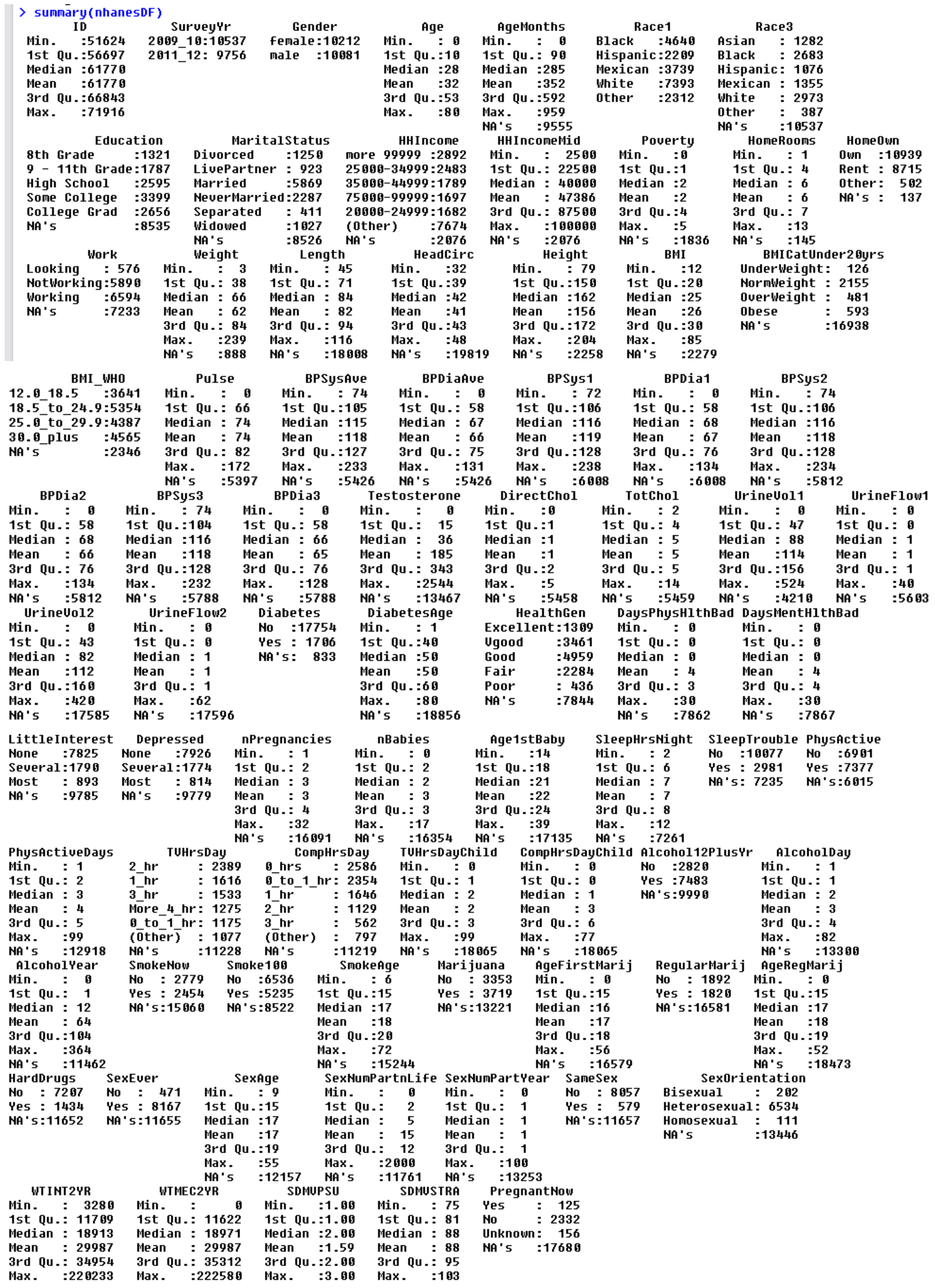


```
> str(nhanesDF)
'data.frame':   20293 obs. of  78 variables:
 $ ID              : int  51624 51625 51626 51627 51628 51629 51630 51631 51632 51633 ...
 $ SurveyYr        : Factor w/ 2 levels "2009_10","2011_12": 1 1 1 1 1 1 1 1 1 1 ...
 $ Gender          : Factor w/ 2 levels "female","male": 2 2 2 2 1 2 1 1 2 2 ...
 $ Age             : int  34 4 16 10 60 26 49 1 10 80 ...
 $ AgeMonths       : int  409 49 202 131 722 313 596 12 124 NA ...
 $ Race1           : Factor w/ 5 levels "Black","Hispanic",..: 4 5 1 1 1 3 4 4 2 4 ...
 $ Race3           : Factor w/ 6 levels "Asian","Black",..: NA NA NA NA NA NA NA NA NA NA ...
 $ Education       : Factor w/ 5 levels "8th Grade","9 - 11th Grade",..: 3 NA NA NA 3 2 4 NA NA 4 ...
 $ MaritalStatus   : Factor w/ 6 levels "Divorced","LivePartner",..: 3 NA NA NA 6 3 2 NA NA 3 ...
 $ HHIncome        : Factor w/ 12 levels "0-4999","10000-14999",..: 5 4 7 4 2 5 6 6 10 3 ...
 $ HHIncomeMid     : int  30000 22500 50000 22500 12500 30000 40000 40000 70000 17500 ...
 $ Poverty         : num  1.36 1.07 2.27 0.81 0.69 1.01 1.91 1.36 2.68 1.27 ...
 $ HomeRooms       : int  6 9 5 6 6 4 5 5 7 4 ...
 $ HomeOwn         : Factor w/ 3 levels "Own","Rent","Other": 1 1 1 2 2 2 2 2 1 1 ...
 $ Work            : Factor w/ 3 levels "Looking","NotWorking",..: 2 NA 2 NA 2 3 2 NA NA 2 ...
 $ Weight          : num  87.4 17 72.3 39.8 116.8 ...
 $ Length          : num  NA NA NA NA NA NA NA 75.7 NA NA ...
 $ HeadCirc        : num  NA NA NA NA NA NA NA NA NA NA ...
 $ Height          : num  165 105 181 148 166 ...
 $ BMI             : num  32.2 15.3 22 18.2 42.4 ...
 $ BMICatUnder20yrs: Factor w/ 4 levels "UnderWeight",..: NA NA NA NA NA NA NA NA NA NA ...
 $ BMI_WHO         : Factor w/ 4 levels "12.0_18.5","18.5_to_24.9",..: 4 1 2 1 4 4 4 NA 1 3 ...
 $ Pulse           : int  70 NA 68 68 72 72 86 NA 70 88 ...
 $ BPSysAve        : int  113 NA 109 93 150 104 112 NA 108 139 ...
 $ BPDiaAve        : int  85 NA 59 41 68 49 75 NA 53 43 ...
 $ BPSys1          : int  114 NA 112 92 154 102 118 NA 106 142 ...
 $ BPDia1          : int  88 NA 62 36 70 50 82 NA 60 62 ...
 $ BPSys2          : int  114 NA 114 94 150 104 108 NA 106 140 ...
 $ BPDia2          : int  88 NA 60 44 68 48 74 NA 50 46 ...
 $ BPSys3          : int  112 NA 104 92 150 104 116 NA 110 138 ...
 $ BPDia3          : int  82 NA 58 38 68 50 76 NA 56 40 ...
 $ Testosterone    : num  NA NA NA NA NA NA NA NA NA NA ...
 $ DirectChol      : num  1.29 NA 1.55 1.89 1.16 1.16 1.16 NA 1.58 1.94 ...
 $ TotChol         : num  3.49 NA 4.97 4.16 5.22 4.14 6.7 NA 4.14 4.71 ...
 $ UrineVol1       : int  352 NA 281 139 30 202 77 NA 39 128 ...
 $ UrineFlow1      : num  NA NA 0.415 1.078 0.476 ...
 $ UrineVol2       : int  NA NA NA NA 246 NA NA NA NA NA ...
 $ UrineFlow2      : num  NA NA NA NA 2.51 NA NA NA NA NA ...
 $ Diabetes        : Factor w/ 2 levels "No","Yes": 1 1 1 1 2 1 1 1 1 1 ...
 $ DiabetesAge     : int  NA NA NA NA 56 NA NA NA NA NA ...
 $ HealthGen       : Factor w/ 5 levels "Excellent","Vgood",..: 3 NA 2 NA 4 3 3 NA NA 1 ...
 $ DaysPhysHlthBad : int  0 NA 2 NA 20 2 0 NA NA 0 ...
 $ DaysMentHlthBad : int  15 NA 0 NA 25 14 10 NA NA 0 ...
 $ LittleInterest  : Factor w/ 3 levels "None","Several",..: 3 NA NA NA 3 1 2 NA NA 1 ...
 $ Depressed       : Factor w/ 3 levels "None","Several",..: 2 NA NA NA 3 3 2 NA NA 1 ...
 $ nPregnancies    : int  NA NA NA NA 1 NA 2 NA NA NA ...
 $ nBabies         : int  NA NA NA NA 1 NA 2 NA NA NA ...
 $ Age1stBaby      : int  NA NA NA NA NA NA 27 NA NA NA ...
 $ SleepHrsNight   : int  4 NA 8 NA 4 4 8 NA NA 6 ...
 $ SleepTrouble    : Factor w/ 2 levels "No","Yes": 2 NA 1 NA 1 1 2 NA NA 1 ...
 $ PhysActive      : Factor w/ 2 levels "No","Yes": 1 NA 2 NA 1 2 1 NA NA 2 ...
 $ PhysActiveDays  : int  NA NA 5 NA NA 2 NA NA NA 4 ...
 $ TVHrsDay        : Factor w/ 7 levels "0_hrs","0_to_1_hr",..: NA NA NA NA NA NA NA NA NA NA ...
 $ CompHrsDay      : Factor w/ 7 levels "0_hrs","0_to_1_hr",..: NA NA NA NA NA NA NA NA NA NA ...
 $ TVHrsDayChild   : int  NA 4 NA 1 NA NA NA NA 1 NA ...
 $ CompHrsDayChild : int  NA 1 NA 1 NA NA NA NA 0 NA ...
 $ Alcohol12PlusYr : Factor w/ 2 levels "No","Yes": 2 NA NA NA 1 2 2 NA NA 2 ...
 $ AlcoholDay      : int  NA NA NA NA NA 19 2 NA NA 1 ...
 $ AlcoholYear     : int  0 NA NA NA 0 48 20 NA NA 52 ...
 $ SmokeNow        : Factor w/ 2 levels "No","Yes": 1 NA NA NA 2 1 2 NA NA 1 ...
 $ Smoke100        : Factor w/ 2 levels "No","Yes": 2 NA NA NA 2 2 2 NA NA 2 ...
 $ SmokeAge        : int  18 NA NA NA 16 15 38 NA NA 16 ...
 $ Marijuana       : Factor w/ 2 levels "No","Yes": 2 NA NA NA NA 2 2 NA NA NA ...
 $ AgeFirstMarij   : int  17 NA NA NA NA 10 18 NA NA NA ...
 $ RegularMarij    : Factor w/ 2 levels "No","Yes": 1 NA NA NA NA 2 1 NA NA NA ...
 $ AgeRegMarij     : int  NA NA NA NA NA 12 NA NA NA NA ...
 $ HardDrugs       : Factor w/ 2 levels "No","Yes": 2 NA NA NA 1 2 2 NA NA NA ...
 $ SexEver         : Factor w/ 2 levels "No","Yes": 2 NA NA NA 2 2 2 NA NA NA ...
 $ SexAge          : int  16 NA NA NA 15 9 12 NA NA NA ...
 $ SexNumPartnLife : int  8 NA NA NA 4 10 10 NA NA NA ...
 $ SexNumPartYear  : int  1 NA NA NA NA 1 1 NA NA NA ...
 $ SameSex         : Factor w/ 2 levels "No","Yes": 1 NA NA NA 1 1 2 NA NA NA ...
 $ SexOrientation  : Factor w/ 3 levels "Bisexual","Heterosexual",..: 2 NA NA NA 2 2 NA NA NA ...
 $ WTINT2YR        : num  80101 53901 13953 11665 20090 ...
 $ WTMEC2YR        : num  81529 56995 14509 12042 21000 ...
 $ SDMVPSU         : int  1 2 1 2 2 1 2 2 2 1 ...
 $ SDMVSTRA        : int  83 79 84 86 75 88 85 86 88 77 ...
 $ PregnantNow     : Factor w/ 3 levels "Yes","No","Unknown": NA NA NA NA NA NA NA NA NA NA ...
```

From the structure function in R, we can see that there are 20,293 observations and 78 variables. There is a mix of numerical and factor data types. Now, lets look briefly at statistics on each variables.

summary(nhanesDF)

2. Expository data analytics (EDA) feat. Plotly



```
> summary(nhanesDF)
       ID            SurveyYr        Gender           Age          AgeMonths         Race1            Race3
 Min.   :51624   2009_10:10537   female:10212   Min.   : 0    Min.   :  0    Black   :4640   Asian   : 1282
 1st Qu.:56697   2011_12: 9756   male  :10081   1st Qu.:10    1st Qu.: 90    Hispanic:2209   Black   : 2683
 Median :61770                                  Median :28    Median :285    Mexican :3739   Hispanic: 1076
 Mean   :61770                                  Mean   :32    Mean   :352    White   :7393   Mexican : 1355
 3rd Qu.:66843                                  3rd Qu.:53    3rd Qu.:592    Other   :2312   White   : 2973
 Max.   :71916                                  Max.   :80    Max.   :959                    Other   :  387
                                                              NA's   :9555                   NA's    :10537
          Education        MaritalStatus        HHIncome      HHIncomeMid        Poverty        HomeRooms      HomeOwn
 8th Grade     :1321   Divorced    :1250   more 99999 :2892   Min.   :  2500   Min.   :0      Min.   : 1    Own  :10939
 9 - 11th Grade:1787   LivePartner : 923   25000-34999:2483   1st Qu.: 22500   1st Qu.:1      1st Qu.: 4    Rent : 8715
 High School   :2595   Married     :5869   35000-44999:1789   Median : 40000   Median :2      Median : 6    Other:  502
 Some College  :3399   NeverMarried:2287   75000-99999:1697   Mean   : 47386   Mean   :2      Mean   : 6    NA's :  137
 College Grad  :2656   Separated   : 411   20000-24999:1682   3rd Qu.: 87500   3rd Qu.:4      3rd Qu.: 7
 NA's          :8535   Widowed     :1027   (Other)    :7674   Max.   :100000   Max.   :5      Max.   :13
                       NA's        :8526   NA's       :2076   NA's   :2076     NA's   :1836   NA's   :145
        Work          Weight          Length          HeadCirc          Height           BMI            BMICatUnder20yrs
 Looking   : 576   Min.   :  3   Min.   : 45     Min.   :32       Min.   : 79     Min.   :12      UnderWeight:  126
 NotWorking:5890   1st Qu.: 38   1st Qu.: 71     1st Qu.:39       1st Qu.:150     1st Qu.:20      NormWeight : 2155
 Working   :6594   Median : 66   Median : 84     Median :42       Median :162     Median :25      OverWeight :  481
 NA's      :7233   Mean   : 62   Mean   : 82     Mean   :41       Mean   :156     Mean   :26      Obese      :  593
                   3rd Qu.: 84   3rd Qu.: 94     3rd Qu.:43       3rd Qu.:172     3rd Qu.:30      NA's       :16938
                   Max.   :239   Max.   :116     Max.   :48       Max.   :204     Max.   :85
                   NA's   :888   NA's   :18008   NA's   :19819    NA's   :2258    NA's   :2279

        BMI_WHO          Pulse          BPSysAve        BPDiaAve         BPSys1           BPDia1          BPSys2
 12.0_18.5   :3641   Min.   :  0    Min.   : 74    Min.   :  0    Min.   : 72    Min.   :  0    Min.   : 74
 18.5_to_24.9:5354   1st Qu.: 66    1st Qu.:105    1st Qu.: 58    1st Qu.:106    1st Qu.: 58    1st Qu.:106
 25.0_to_29.9:4387   Median : 74    Median :115    Median : 67    Median :116    Median : 68    Median :116
 30.0_plus   :4565   Mean   : 74    Mean   :118    Mean   : 66    Mean   :119    Mean   : 67    Mean   :118
 NA's        :2346   3rd Qu.: 82    3rd Qu.:127    3rd Qu.: 75    3rd Qu.:128    3rd Qu.: 76    3rd Qu.:128
                     Max.   :172    Max.   :233    Max.   :131    Max.   :238    Max.   :134    Max.   :234
                     NA's   :5397   NA's   :5426   NA's   :5426   NA's   :6008   NA's   :6008   NA's   :5812
     BPDia2          BPSys3          BPDia3        Testosterone      DirectChol        TotChol        UrineVol1       UrineFlow1
 Min.   :  0    Min.   : 74    Min.   :  0    Min.   :    0    Min.   :0       Min.   : 2     Min.   :  0    Min.   : 0
 1st Qu.: 58    1st Qu.:104    1st Qu.: 58    1st Qu.:   15    1st Qu.:1       1st Qu.: 4     1st Qu.: 47    1st Qu.: 0
 Median : 68    Median :116    Median : 66    Median :   36    Median :1       Median : 5     Median : 88    Median : 1
 Mean   : 66    Mean   :118    Mean   : 65    Mean   :  185    Mean   :1       Mean   : 5     Mean   :114    Mean   : 1
 3rd Qu.: 76    3rd Qu.:128    3rd Qu.: 76    3rd Qu.:  343    3rd Qu.:2       3rd Qu.: 5     3rd Qu.:156    3rd Qu.: 1
 Max.   :134    Max.   :232    Max.   :128    Max.   :2544     Max.   :5       Max.   :14     Max.   :524    Max.   :40
 NA's   :5812   NA's   :5788   NA's   :5788   NA's   :13467    NA's   :5458    NA's   :5459   NA's   :4210   NA's   :5603
   UrineVol2        UrineFlow2      Diabetes      DiabetesAge          HealthGen      DaysPhysHlthBad DaysMentHlthBad
 Min.   :  0     Min.   : 0      No  :17754   Min.   : 1        Excellent:1309   Min.   : 0      Min.   : 0
 1st Qu.: 43     1st Qu.: 0      Yes : 1706   1st Qu.:40        Vgood    :3461   1st Qu.: 0      1st Qu.: 0
 Median : 82     Median : 1      NA's:  833   Median :50        Good     :4959   Median : 0      Median : 0
 Mean   :112     Mean   : 1                   Mean   :50        Fair     :2284   Mean   : 4      Mean   : 4
 3rd Qu.:160     3rd Qu.: 1                   3rd Qu.:60        Poor     : 436   3rd Qu.: 3      3rd Qu.: 4
 Max.   :420     Max.   :62                   Max.   :80        NA's     :7844   Max.   :30      Max.   :30
 NA's   :17585   NA's   :17596                NA's   :18856                      NA's   :7862    NA's   :7867

 LittleInterest  Depressed     nPregnancies       nBabies         Age1stBaby      SleepHrsNight   SleepTrouble PhysActive
 None   :7825   None   :7926   Min.   : 1      Min.   : 0      Min.   :14      Min.   : 2      No  :10077   No  :6901
 Several:1790   Several:1774   1st Qu.: 2      1st Qu.: 2      1st Qu.:18      1st Qu.: 6      Yes : 2981   Yes :7377
 Most   : 893   Most   : 814   Median : 3      Median : 2      Median :21      Median : 7      NA's: 7235   NA's:6015
 NA's   :9785   NA's   :9779   Mean   : 3      Mean   : 3      Mean   :22      Mean   : 7
                               3rd Qu.: 4      3rd Qu.: 3      3rd Qu.:24      3rd Qu.: 8
                               Max.   :32      Max.   :17      Max.   :39      Max.   :12
                               NA's   :16091   NA's   :16354   NA's   :17135   NA's   :7261
 PhysActiveDays       TVHrsDay        CompHrsDay      TVHrsDayChild    CompHrsDayChild Alcohol12PlusYr  AlcoholDay
 Min.   : 1      2_hr      : 2389   0_hrs     : 2586   Min.   : 0      Min.   : 0      No  :2820       Min.   : 1
 1st Qu.: 2      1_hr      : 1616   0_to_1_hr: 2354   1st Qu.: 1      1st Qu.: 0      Yes :7483       1st Qu.: 1
 Median : 3      3_hr      : 1533   1_hr      : 1646   Median : 2      Median : 1      NA's:9990       Median : 2
 Mean   : 4      More_4_hr: 1275   2_hr      : 1129   Mean   : 2      Mean   : 3                      Mean   : 3
 3rd Qu.: 5      0_to_1_hr: 1175   3_hr      :  562   3rd Qu.: 3      3rd Qu.: 6                      3rd Qu.: 4
 Max.   :99      (Other)  : 1077   (Other)   :  797   Max.   :99      Max.   :77                      Max.   :82
 NA's   :12918   NA's      :11228   NA's      :11219   NA's   :18065   NA's   :18065                   NA's   :13300
  AlcoholYear     SmokeNow      Smoke100        SmokeAge       Marijuana      AgeFirstMarij   RegularMarij  AgeRegMarij
 Min.   :  0    No  : 2779   No  :6536   Min.   : 6      No  : 3353   Min.   : 0      No  : 1892   Min.   : 0
 1st Qu.:  1    Yes : 2454   Yes :5235   1st Qu.:15      Yes : 3719   1st Qu.:15      Yes : 1820   1st Qu.:15
 Median : 12    NA's:15060   NA's:8522   Median :17      NA's:13221   Median :16      NA's:16581   Median :17
 Mean   : 64                             Mean   :18                   Mean   :17                   Mean   :18
 3rd Qu.:104                             3rd Qu.:20                   3rd Qu.:18                   3rd Qu.:19
 Max.   :364                             Max.   :72                   Max.   :56                   Max.   :52
 NA's   :11462                           NA's   :15244                NA's   :16579                NA's   :18473
 HardDrugs     SexEver          SexAge        SexNumPartnLife  SexNumPartYear  SameSex          SexOrientation
 No  : 7207   No  :  471   Min.   : 9      Min.   :   0     Min.   :  0     No  : 8057   Bisexual    :  202
 Yes : 1434   Yes : 8167   1st Qu.:15      1st Qu.:   2     1st Qu.:  1     Yes :  579   Heterosexual: 6534
 NA's:11652   NA's:11655   Median :17      Median :   5     Median :  1     NA's:11657   Homosexual  :  111
                           Mean   :17      Mean   :  15     Mean   :  1                  NA's        :13446
                           3rd Qu.:19      3rd Qu.:  12     3rd Qu.:  1
                           Max.   :55      Max.   :2000     Max.   :100
                           NA's   :12157   NA's   :11761    NA's   :13253
    WTINT2YR          WTMEC2YR          SDMVPSU         SDMVSTRA     PregnantNow
 Min.   :  3280   Min.   :     0   Min.   :1.00   Min.   : 75   Yes    :  125
 1st Qu.: 11709   1st Qu.: 11622   1st Qu.:1.00   1st Qu.: 81   No     : 2332
 Median : 18913   Median : 18971   Median :2.00   Median : 88   Unknown:  156
 Mean   : 29987   Mean   : 29987   Mean   :1.59   Mean   : 88   NA's   :17680
 3rd Qu.: 34954   3rd Qu.: 35312   3rd Qu.:2.00   3rd Qu.: 95
 Max.   :220233   Max.   :222580   Max.   :3.00   Max.   :103
```

From the summary function in R we find common statistics by each variable. Since Diabetes is our target variable in this study, let's start with that. We see of the 20,293 observations 17,754 (~87% of total) do not have diabetes, 1,706 (~8% of total) claim to have diabetes, and 833 (~4% of total) are NA's.

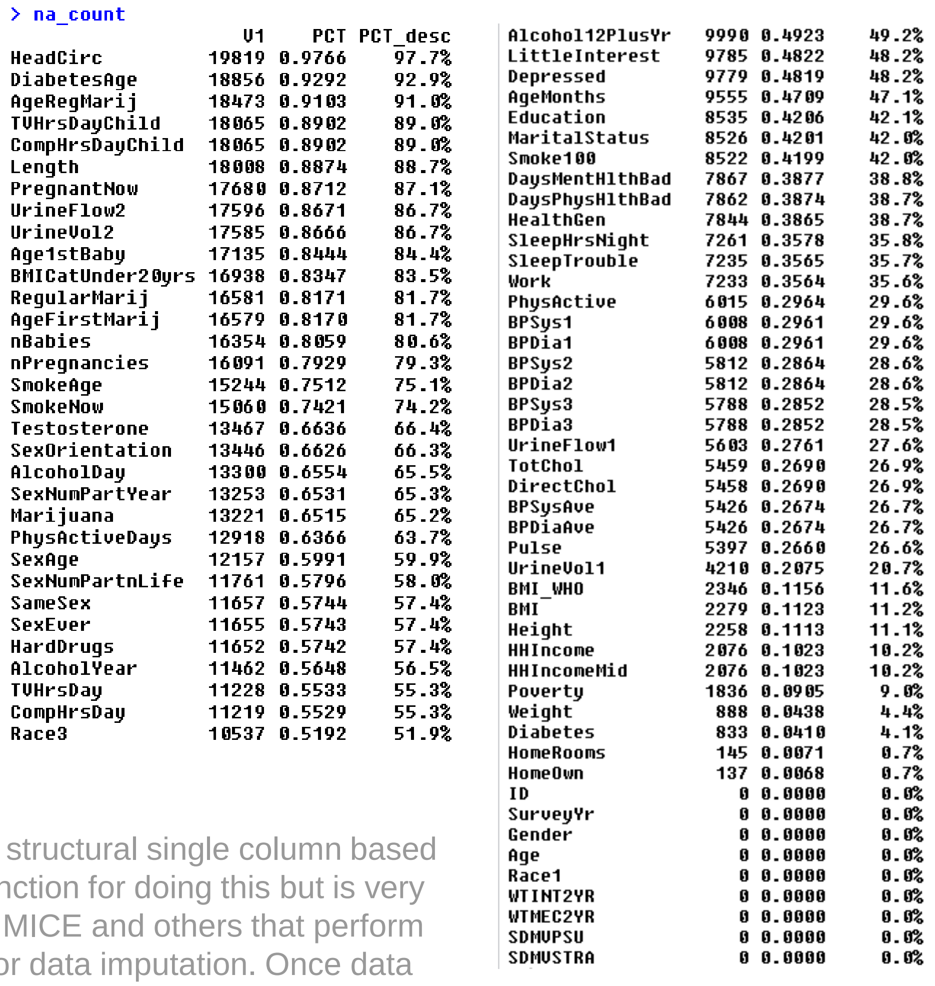
Since our target variable only occurs within 8% of the observations, this is called a class imbalance problem. This is important in that our target variable is not equally distributed among the observations, and therefore distribution and sampling needs to be handled with care when making predictions in such cases.

We will observe how H2O.ai handles these cases both in clustering and prediction settings.

The other issue we can see within this data set is the prevalence of NA's that varies across all our variables.

Let's determine how many NA's by variable we are looking at and determine a course of action to handle these events. We will see later on how H2O.ai handles the treatment of NA's within observations, but for now let's run the below code to determine how many NA's by feature we are dealing with at this time:

```
## Count NA's by columns
na_count <- as.list(sapply(nhanesDF, function(y) sum(length(which(is.na(y))))))
na_count <- as.data.frame(na_count)
na_count <- as.data.frame(t(na_count))
library(scales)
na_count$PCT <- na_count$V1 / nrow(nhanesDF)
na_count$PCT_desc <- percent(na_count$V1 / nrow(nhanesDF))
na_count <- na_count[order(-na_count$V1),]
na_count
```



```
> na_count
                    V1    PCT PCT_desc
HeadCirc         19819 0.9766    97.7%
DiabetesAge      18856 0.9292    92.9%
AgeRegMarij      18473 0.9103    91.0%
TVHrsDayChild    18065 0.8902    89.0%
CompHrsDayChild  18065 0.8902    89.0%
Length           18008 0.8874    88.7%
PregnantNow      17680 0.8712    87.1%
UrineFlow2       17596 0.8671    86.7%
UrineVol2        17585 0.8666    86.7%
Age1stBaby       17135 0.8444    84.4%
BMICatUnder20yrs 16938 0.8347    83.5%
RegularMarij     16581 0.8171    81.7%
AgeFirstMarij    16579 0.8170    81.7%
nBabies          16354 0.8059    80.6%
nPregnancies     16091 0.7929    79.3%
SmokeAge         15244 0.7512    75.1%
SmokeNow         15060 0.7421    74.2%
Testosterone     13467 0.6636    66.4%
SexOrientation   13446 0.6626    66.3%
AlcoholDay       13300 0.6554    65.5%
SexNumPartYear   13253 0.6531    65.3%
Marijuana        13221 0.6515    65.2%
PhysActiveDays   12918 0.6366    63.7%
SexAge           12157 0.5991    59.9%
SexNumPartnLife  11761 0.5796    58.0%
SameSex          11657 0.5744    57.4%
SexEver          11655 0.5743    57.4%
HardDrugs        11652 0.5742    57.4%
AlcoholYear      11462 0.5648    56.5%
TVHrsDay         11228 0.5533    55.3%
CompHrsDay       11219 0.5529    55.3%
Race3            10537 0.5192    51.9%
Alcohol12PlusYr   9990 0.4923    49.2%
LittleInterest    9785 0.4822    48.2%
Depressed         9779 0.4819    48.2%
AgeMonths         9555 0.4709    47.1%
Education         8535 0.4206    42.1%
MaritalStatus     8526 0.4201    42.0%
Smoke100          8522 0.4199    42.0%
DaysMentHlthBad   7867 0.3877    38.8%
DaysPhysHlthBad   7862 0.3874    38.7%
HealthGen         7844 0.3865    38.7%
SleepHrsNight     7261 0.3578    35.8%
SleepTrouble      7235 0.3565    35.7%
Work              7233 0.3564    35.6%
PhysActive        6015 0.2964    29.6%
BPSys1            6008 0.2961    29.6%
BPDia1            6008 0.2961    29.6%
BPSys2            5812 0.2864    28.6%
BPDia2            5812 0.2864    28.6%
BPSys3            5788 0.2852    28.5%
BPDia3            5788 0.2852    28.5%
UrineFlow1        5603 0.2761    27.6%
TotChol           5459 0.2690    26.9%
DirectChol        5458 0.2690    26.9%
BPSysAve          5426 0.2674    26.7%
BPDiaAve          5426 0.2674    26.7%
Pulse             5397 0.2660    26.6%
UrineVol1         4210 0.2075    20.7%
BMI_WHO           2346 0.1156    11.6%
BMI               2279 0.1123    11.2%
Height            2258 0.1113    11.1%
HHIncome          2076 0.1023    10.2%
HHIncomeMid       2076 0.1023    10.2%
Poverty           1836 0.0905     9.0%
Weight             888 0.0438     4.4%
Diabetes           833 0.0410     4.1%
HomeRooms          145 0.0071     0.7%
HomeOwn            137 0.0068     0.7%
ID                   0 0.0000     0.0%
SurveyYr             0 0.0000     0.0%
Gender               0 0.0000     0.0%
Age                  0 0.0000     0.0%
Race1                0 0.0000     0.0%
WTINT2YR             0 0.0000     0.0%
WTMEC2YR             0 0.0000     0.0%
SDMVPSU              0 0.0000     0.0%
SDMVSTRA             0 0.0000     0.0%
```

The code produces a data frame that contains all the variables within the NHANES dataset and counts and calculates the percentage of NA's found within each variable. The data frame is sorted in descending order so that we can see the variables that contain the highest percentage of NA's found within that specific variable.

Since 32 out of the 78 total variables (~41%) contain 50% or more NA's within each variable we will need to formulate a strategy on how we can go forward with our EDA, clustering, and predictive analytics. We will perform the below using H2O.ai

1) Impute the missing values with structural single column based methods. H2O.ai provides a function for doing this but is very limited. See packages such as MICE and others that perform much more detailed methods for data imputation. Once data is imputed, we will run our models, still within H2O.ai framework.
2) H2O.ai framework provides treatment of NA's within the clustering and predictive analytical functions. We will us these built capabilities and compare them to imputed value method.

What is data imputation?

There are situations in data analytical projects where missing values are not relevant and therefore can be ignored without any consequences to the analytical results. Then, there are those situations where ignoring missing values is not possible if the analytical results are to be completed. The later is the situation with the NHANES data set and analytics of diabetes. What is needed is a method which is called data imputation and is defined next.

What is data imputation?

Definition from Wikipedia

"*the process of replacing missing data with substituted values. When substituting for a data point, it is known as "unit imputation"; when substituting for a component of a data point, it is known as "item imputation". Because missing data can create problems for analyzing data, imputation is seen as a way to avoid pitfalls involved with listwise deletion of cases that have missing values. That is to say, when one or more values are missing for a case, most statistical packages default to discarding any case that has a missing value, which may introduce bias or affect the representativeness of the results. Imputation preserves all cases by replacing missing data with an estimated value based on other available information. Once all missing values have been imputed, the data set can then be analysed using standard techniques for complete data*."

Although the definition may sound simple, the considerations and execution of data imputation are very complex. Consider these questions below given our data set of what know about the NHANES dataset to this point:

1) What is the amount of data needed for data imputation? For example, how much is too much of NA's within a column to perform data imputation?
2) What statistical method do you use to formulate the value (mean, median, etc.)?
3) What other values within the data set could be used to "profile" the row and determine the value?

As you can see we can continue our line of questioning to nail down our imputation process.

Fortunately for us, H2O.ai has a simple function that allows us imputed data in a paralyzed fashion on our local machines, while considering these and other types of questions that we can provide within the function. We will loop through all the numeric columns and perform a simple imputation on the missing values.

Data preparations before imputation

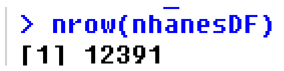
Before we impute the data, let's make a few assumptions around our study of diabetes so that we don't end up imputing data when we really don't even need to in the first place.

1) Consider only adults in our study (> 17 years of age)
   - Children skew our results when we consider BMI, age, and other factors

2) Impute only numerical data
   - For this study we will only impute the numerical data (none of the factor variables) due to simplicity (you can certainly impute factor types as well in h2o.ai, you just need to recode them to numerical types)

3) Assume some threshold of NA's to be dropped form the study
   - Let's assume in this study that any field with greater than 75% NA's is just too many to draw any meaningful imputation close to the "actual" (supposedly) values.

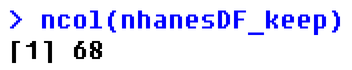
```
## Remove children from study
nhanesDF <- nhanesDF[which(nhanesDF$Age > 17),]
nrow(nhanesDF)
```



```
> nrow(nhanesDF)
[1] 12391
```

Now you should have 12,391 observation with this filter applied. Next we want to re-calculate and remove anything over 75% NA's

```
## Count NA's by columns
na_count <- as.list(sapply(nhanesDF, function(y) sum(length(which(is.na(y))))))
na_count <- as.data.frame(na_count)
na_count <- as.data.frame(t(na_count))
library(scales)
na_count$PCT <- na_count$V1 / nrow(nhanesDF)
na_count$PCT_desc <- percent(na_count$V1 / nrow(nhanesDF))
na_count$variable <- colnames(nhanesDF)
na_count <- na_count[order(-na_count$V1),]

na_count_50 <- na_count[which(na_count$PCT < .75),]

keep <- dimnames(na_count_50)[[1]]
nhanesDF_keep <- nhanesDF[,c(keep)]
ncol(nhanesDF_keep)
```



```
> ncol(nhanesDF_keep)
[1] 68
```

This reduces the number of variables to 68. Next filter to those fields that are of numerical type. We drop ID as we do not want to impute that field.

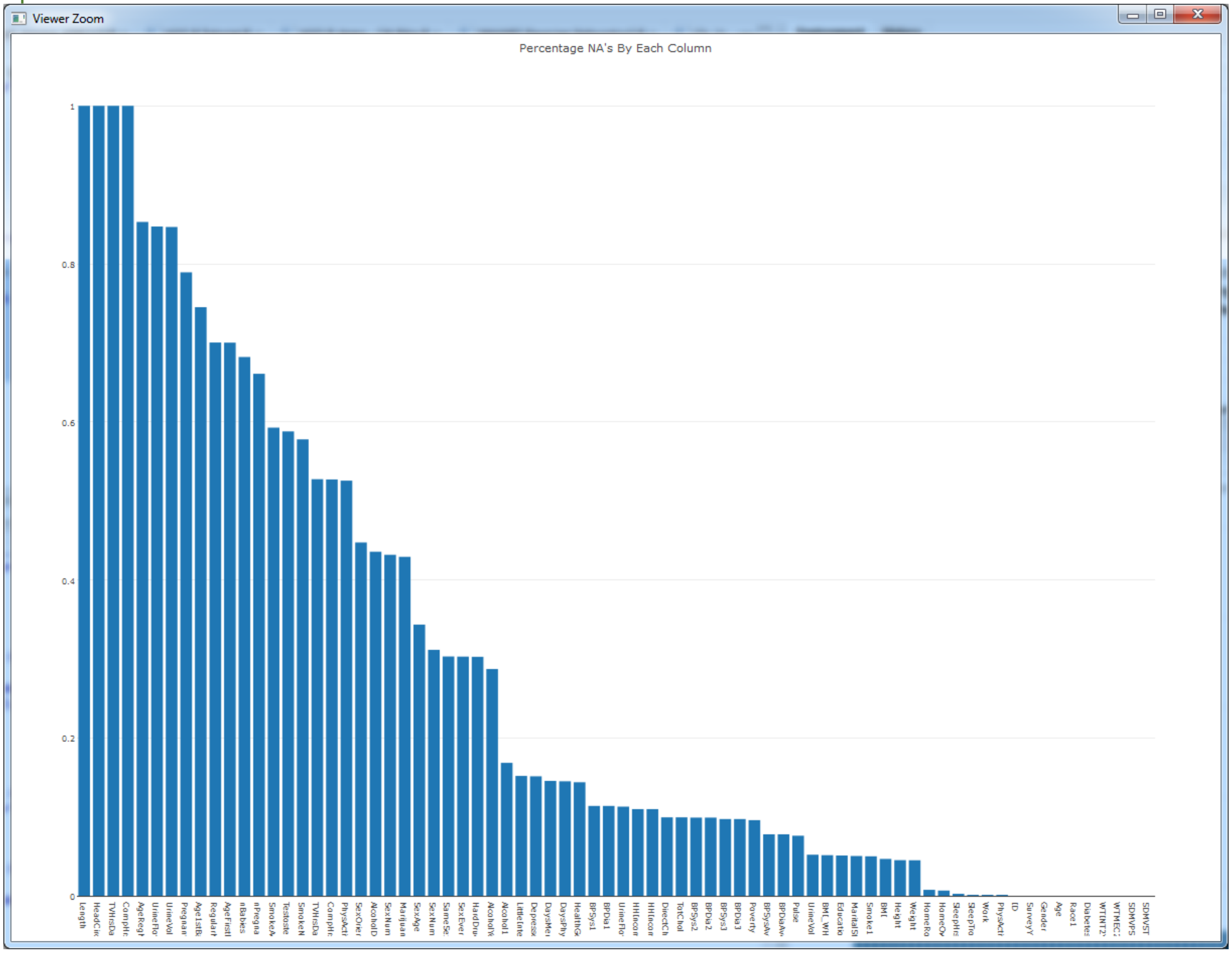
```
nhanes_numDF <- nhanesDF_keep[, sapply(nhanesDF_keep, is.numeric)]
nhanes_numDF$ID <- NULL
```

2. Expository data analytics (EDA) feat. Plotly

Data preparations before imputation (continued …)

To summarize our data prep approach for data imputation, we filtered out children from our study, imputed on the numerical values only, and kept the column where NA's where less than 75%. A summary chart of the percentage of NA's is seen below to visualize the prevalence of NA's by each column.

```
library(plotly)
library(ggplot2)
p <- plot_ly(na_count, x = variable, y = PCT, name = "NA Chart", type = "bar") %>%
  layout(
    title = "Percentage NA's By Each Column",
    xaxis = list(title = ""),
    margin = list(l = 60, r=60),
    yaxis = list(title = ""),
    font = list(size=8)
  )
p
```

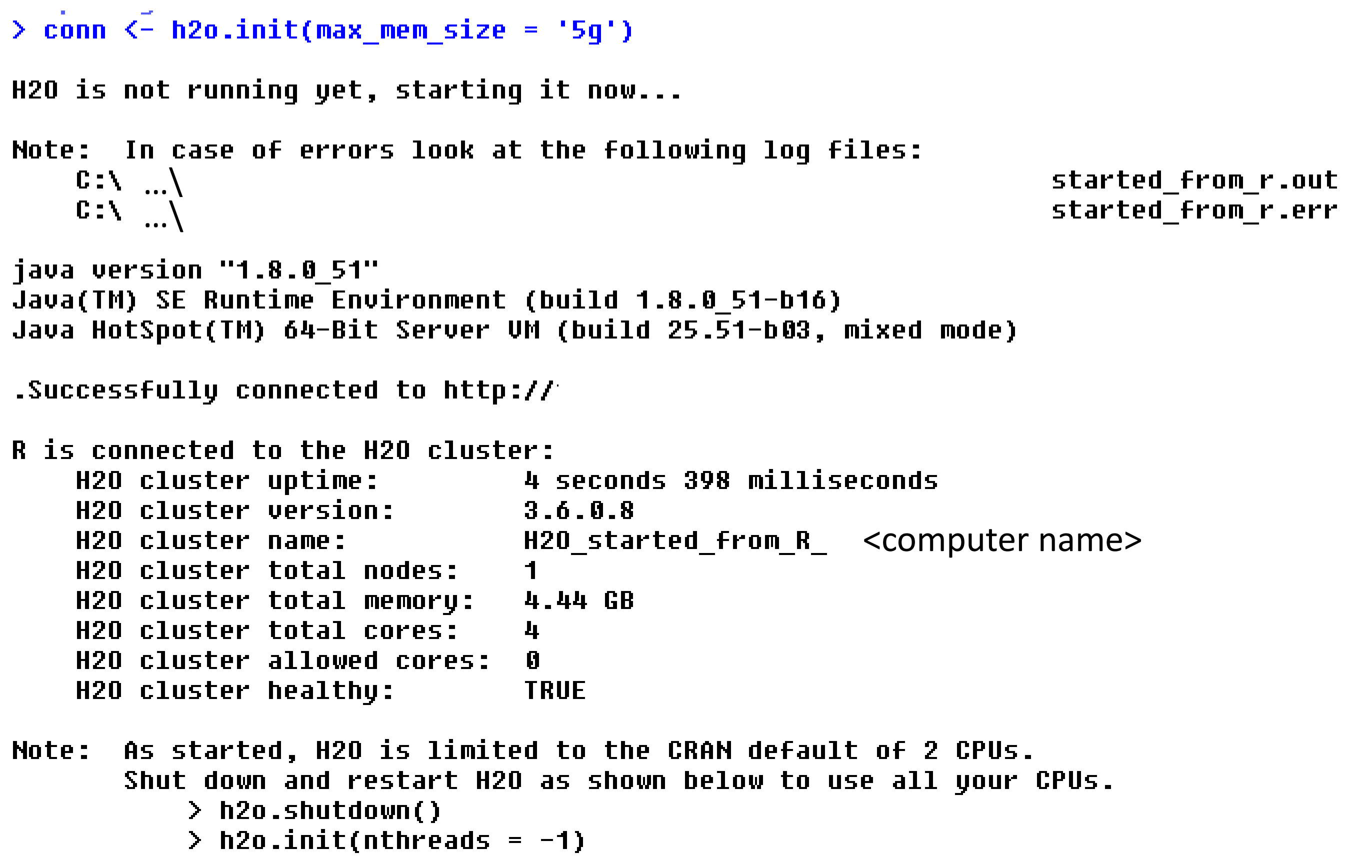


Creating an H2O instance in R

To execute the imputation, we will fire up an H2O instance. You will first need to install the package if you haven't done so already (see introduction).

To start an H2O instance execute the below R code:

```
## Start H2O
library(h2o)
conn <- h2o.init(max_mem_size = '5g')
```



```
> conn <- h2o.init(max_mem_size = '5g')

H2O is not running yet, starting it now...

Note:  In case of errors look at the following log files:
    C:\ ...\                                                      started_from_r.out
    C:\ ...\                                                      started_from_r.err

java version "1.8.0_51"
Java(TM) SE Runtime Environment (build 1.8.0_51-b16)
Java HotSpot(TM) 64-Bit Server VM (build 25.51-b03, mixed mode)

.Successfully connected to http://

R is connected to the H2O cluster:
    H2O cluster uptime:         4 seconds 398 milliseconds
    H2O cluster version:        3.6.0.8
    H2O cluster name:           H2O_started_from_R_  <computer name>
    H2O cluster total nodes:    1
    H2O cluster total memory:   4.44 GB
    H2O cluster total cores:    4
    H2O cluster allowed cores:  0
    H2O cluster healthy:        TRUE

Note:  As started, H2O is limited to the CRAN default of 2 CPUs.
       Shut down and restart H2O as shown below to use all your CPUs.
           > h2o.shutdown()
           > h2o.init(nthreads = -1)
```

Yours will look different than what mine is showing depending on your system and resources on your machine. The key information you want to notice is the total nodes, memory, and cores. This will give you an indication of the parameters h2o will use to optimize in-memory computations on your machine.

Data imputation in H2O+R

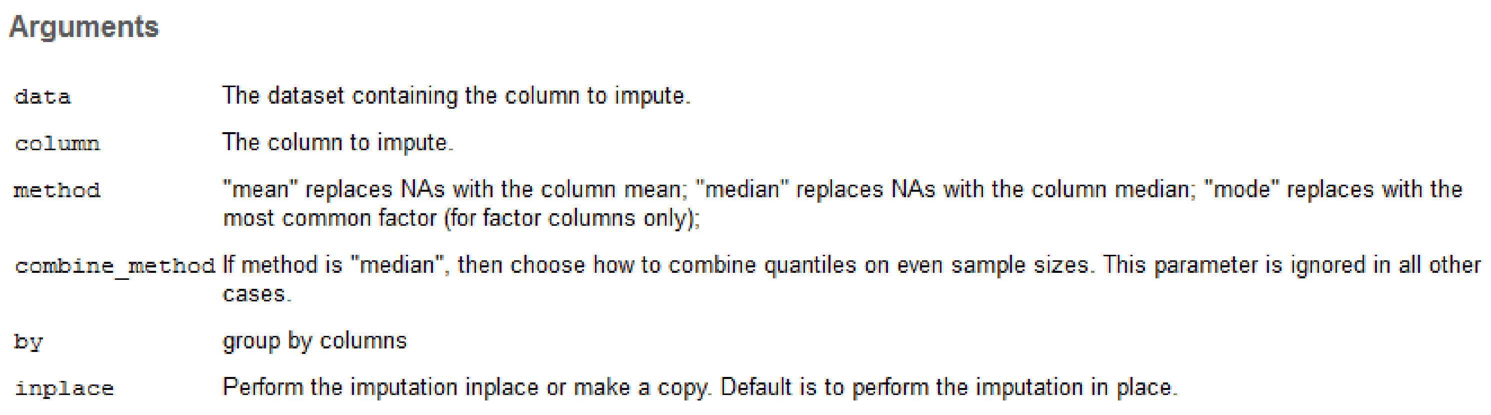
We will perform a simple, single column imputation on the medians of each variable. The subject of imputation is a whole (and interesting) field of work that expands greatly into more effective and detail methods of imputation. To get further details on these methods of data imputation, suggest reviewing Stef van Buuren and Karin Groothuis-Oudshoorn work in the below volume found within the "*Journal of Statistical Software*".

Multivariate Imputation by Chained Equations in R
http://www.jstatsoft.org/article/view/v045i03/v45i03.pdf
R package in CRAN is "mice"

We will be using the h2o.imput() wrapped within a standard R loop to perform the data imputation on each numeric column that we have partition into its own data frame.

```
##---------------------------------------
## Data Imputation
##---------------------------------------
nhanes_numDF_hex <- as.h2o(nhanes_numDF)
nhanes_numDF_Imp_hex <- nhanes_numDF_hex

for (i in 1:ncol(nhanes_numDF_Imp_hex)) {
  nhanes_numDF_Imp_hex[,i] <- h2o.impute(nhanes_numDF_hex,
                                  colnames(nhanes_numDF_hex[,i]),
                                  method = "median", combine_method="lo")[,i]
}
summary(nhanes_numDF_Imp_hex)
```

Reviewing the h2o package reference we see the below arguments the impute function takes to perform the data imputation.



Arguments

`data`	The dataset containing the column to impute.
`column`	The column to impute.
`method`	"mean" replaces NAs with the column mean; "median" replaces NAs with the column median; "mode" replaces with the most common factor (for factor columns only);
`combine_method`	If method is "median", then choose how to combine quantiles on even sample sizes. This parameter is ignored in all other cases.
`by`	group by columns
`inplace`	Perform the imputation inplace or make a copy. Default is to perform the imputation in place.

Once you run the code above, H2O will perform all the imputation (very fast) and stored the results we have named "nhanes_numDF_Imp_hex". Next we will perform our detail exploratory data analysis (EDA) on the original data as well imputed to insure structural distribution has been maintained.

Exploratory data analysis review

To perform our analysis, we will use plotly & ggplot2 to visualize our results along with some statistics that tell us facts regarding distribution, spread, means, correlation, variance. If you haven't installed plotly and ggplot2 in R please refer to the introduction section, then come back here to proceed. These are important basic facts to consider when approaching any data set prior to further analysis.

Proportions and summary tables

Let's start by taking a quick look at the proportion of diabetes by a set of variables. We will just look at a couple to get the general idea.

Let's run a simple proportion tables on diabetes.

```
tbl <- table(nhanesDF$Diabetes)
prop.table(tbl)
```

```
  No  Yes
0.86 0.14
```

As we can see that those with diabetes are 14% of our observations vs. those without are 86%.

Let's break this down by the categorical variables gender, race, and education.

Let's run a simple proportion tables on diabetes.

```
tbl <- with(nhanesDF, table(Gender, Diabetes))
prop.table(tbl, 1)
```

```
        Diabetes
Gender     No  Yes
  female 0.87 0.13
  male   0.86 0.14
```

```
tbl <- with(nhanesDF, table(Race1, Diabetes))
prop.table(tbl, 1)
```

```
          Diabetes
Race1        No  Yes
  Black    0.82 0.18
  Hispanic 0.86 0.14
  Mexican  0.86 0.14
  White    0.89 0.11
  Other    0.88 0.12
```

```
tbl <- with(nhanesDF, table(Education, Diabetes))
prop.table(tbl, 1)
```

```
                Diabetes
Education           No   Yes
  8th Grade      0.762 0.238
  9 - 11th Grade 0.831 0.169
  High School    0.868 0.132
  Some College   0.870 0.130
  College Grad   0.901 0.099
```

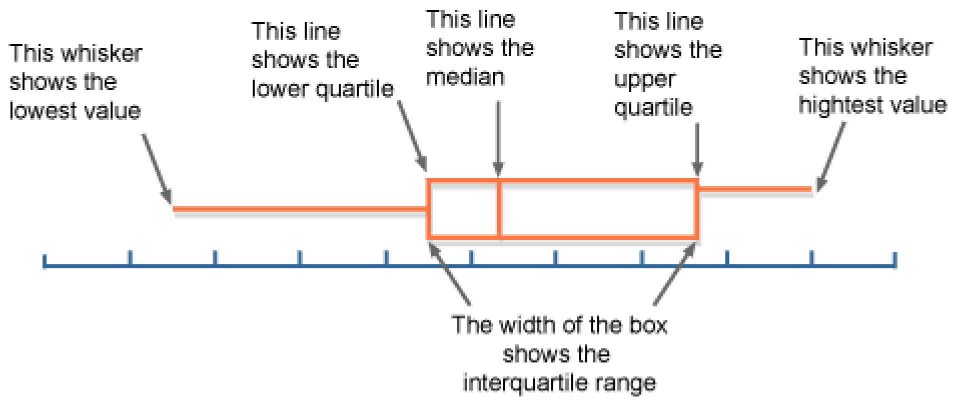
The proportions seem to hold about the same across race and gender. However, the proportion of diabetes found in those with an 8th grade education only (with out further progression) seem to have higher prevalence of diabetes.

Let's now look at some of the distributions by numerical values grouped by diabetes.

2. Expository data analytics (EDA) feat. Plotly

Box plots for distribution analysis

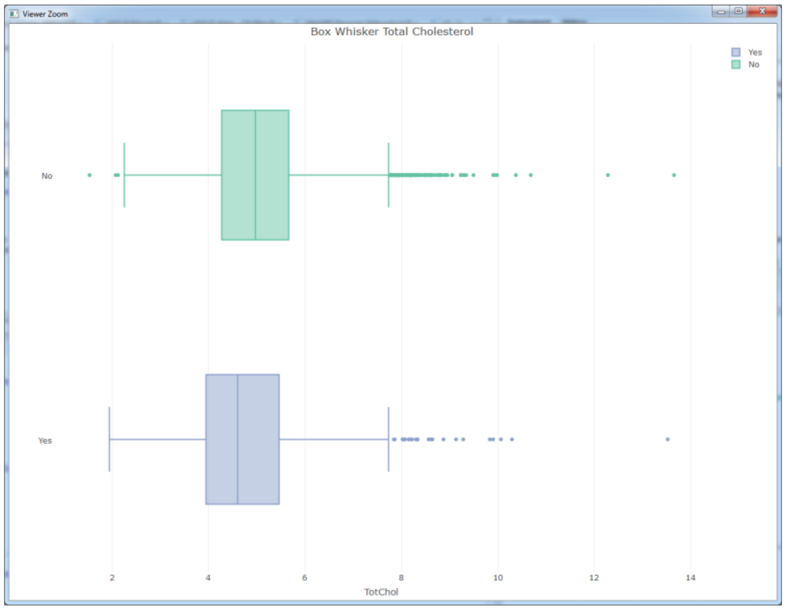
Before diving into the NHANES box plot data, lets do a brief review of box plots in general. Box plot are one of my go to tools for studying variation within continuous data. It allows me to see the shape, central tendency, and variations between and within central limits. Mastering the interpretation of these charts can be very valuable in your daily consumption of statistical information. Below is a simply diagram that shows the basic layout. More information on these charts can be found at: https://en.wikipedia.org/wiki/Box_plot

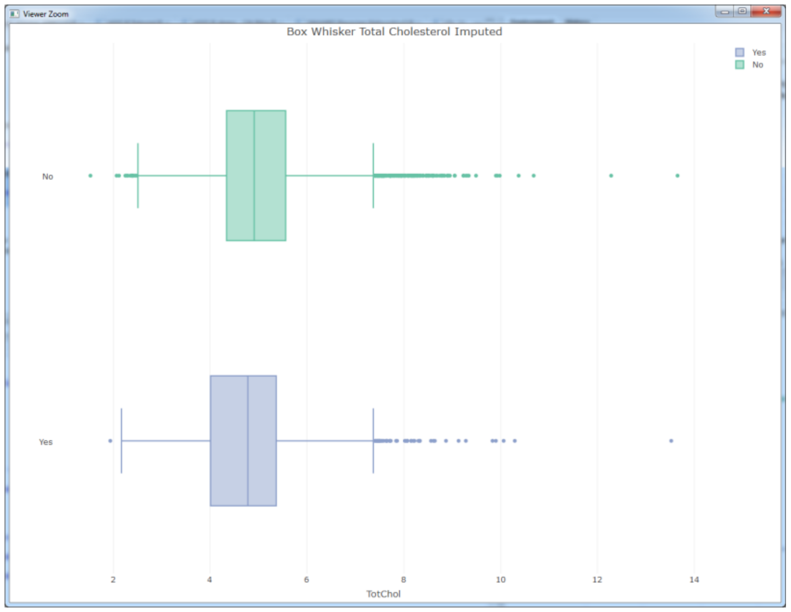

This whisker shows the lowest value
This line shows the lower quartile
This line shows the median
This line shows the upper quartile
This whisker shows the hightest value
The width of the box shows the interquartile range

Now let’s look at a few continuous values and see how they look separated by diabetes. Well, perform one on each by the imputed data and none imputed data because we want to ensure structurally, there are not significant changes due to imputation.

Let’s start with total cholesterol (I’m just randomly selecting these of the key variable that intuitively are associated with diabetes).

```
## TotChol
plot_ly(nhanes_numDF, x = TotChol, color = Diabetes, type = "box")%>%
  layout(title = "Box Whisker Total Cholesterol", margin = list(l = 65))
plot_ly(nhanes_numDF_Imp, x = TotChol, color = Diabetes, type = "box")%>%
  layout(title = "Box Whisker Total Cholesterol Imputed",margin = list(l = 65))
```



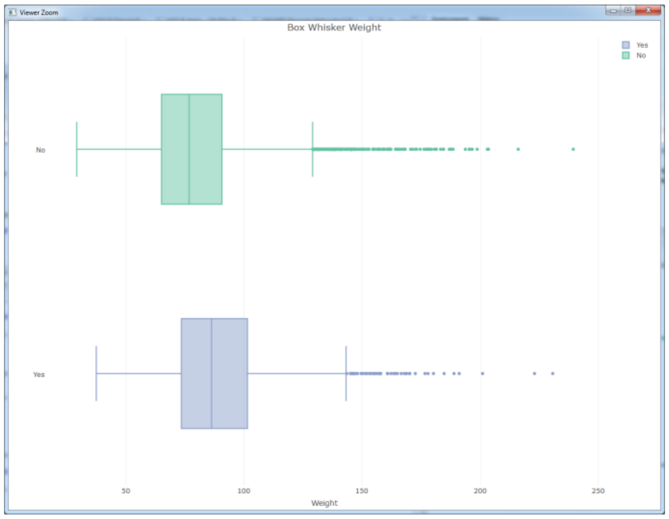


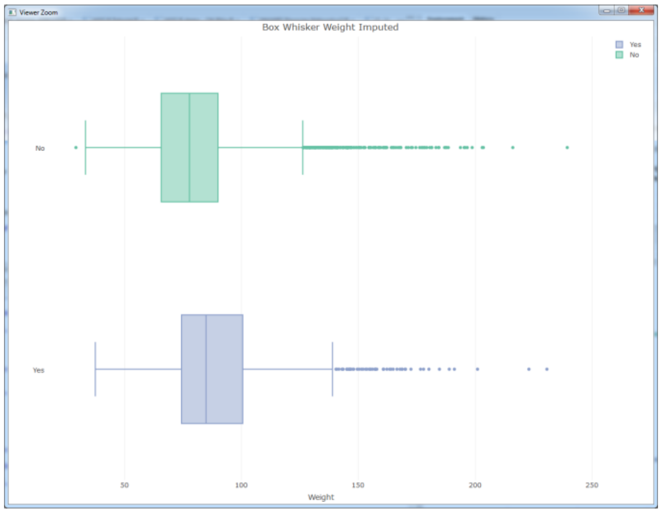
As you can see, not much structural change in the data due to imputation. In fact, those with diabetes and those without seem to have about the same look and shape of the distribution of cholesterol.

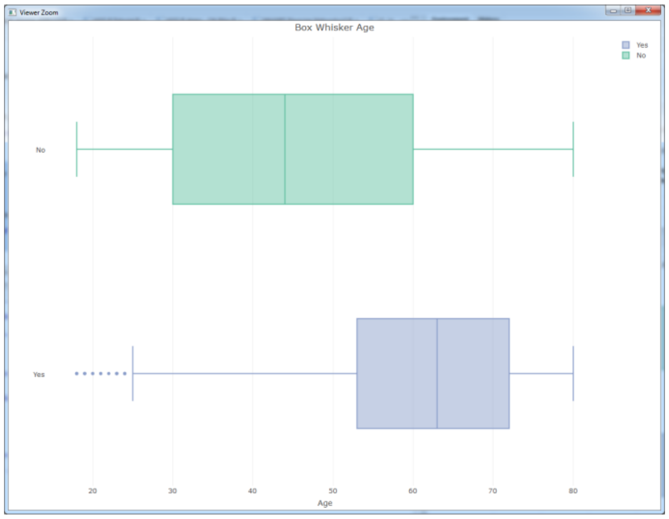
Box plots for distribution analysis (continued …)

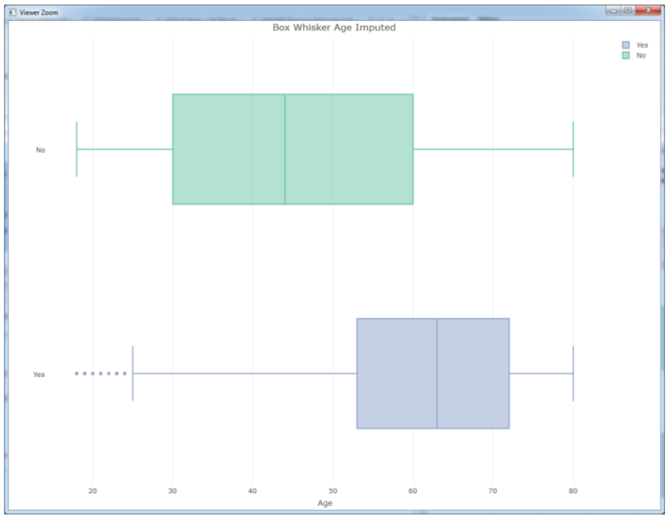
Let's now look at two additional variables, age and weight. From the graph below we see there isn't much structural difference between the imputed data and the actual values without NA's so we conclude that it's ok to proceed with the imputed data. Notice also the how much more the age median and bounds are for those with diabetes and those without. Also, notice the difference in weight between diabetes and no diabetes.

```
## Weight
plot_ly(nhanes_numDF, x = Weight, color = Diabetes, type = "box")%>%
  layout( title = "Box Whisker Weight",margin = list(l = 65))
plot_ly(nhanes_numDF_Imp, x = Weight, color = Diabetes, type = "box")%>%
  layout(title = "Box Whisker Weight Imputed",margin = list(l = 65))

## Age
plot_ly(nhanes_numDF, x = Age, color = Diabetes, type = "box")%>%
  layout(title = "Box Whisker Age",margin = list(l = 65))
plot_ly(nhanes_numDF_Imp, x = Age, color = Diabetes, type = "box")%>%
  layout(title = "Box Whisker Age Imputed",margin = list(l = 65))
```









Dot plot of variables and diabetes (non-imputed data)

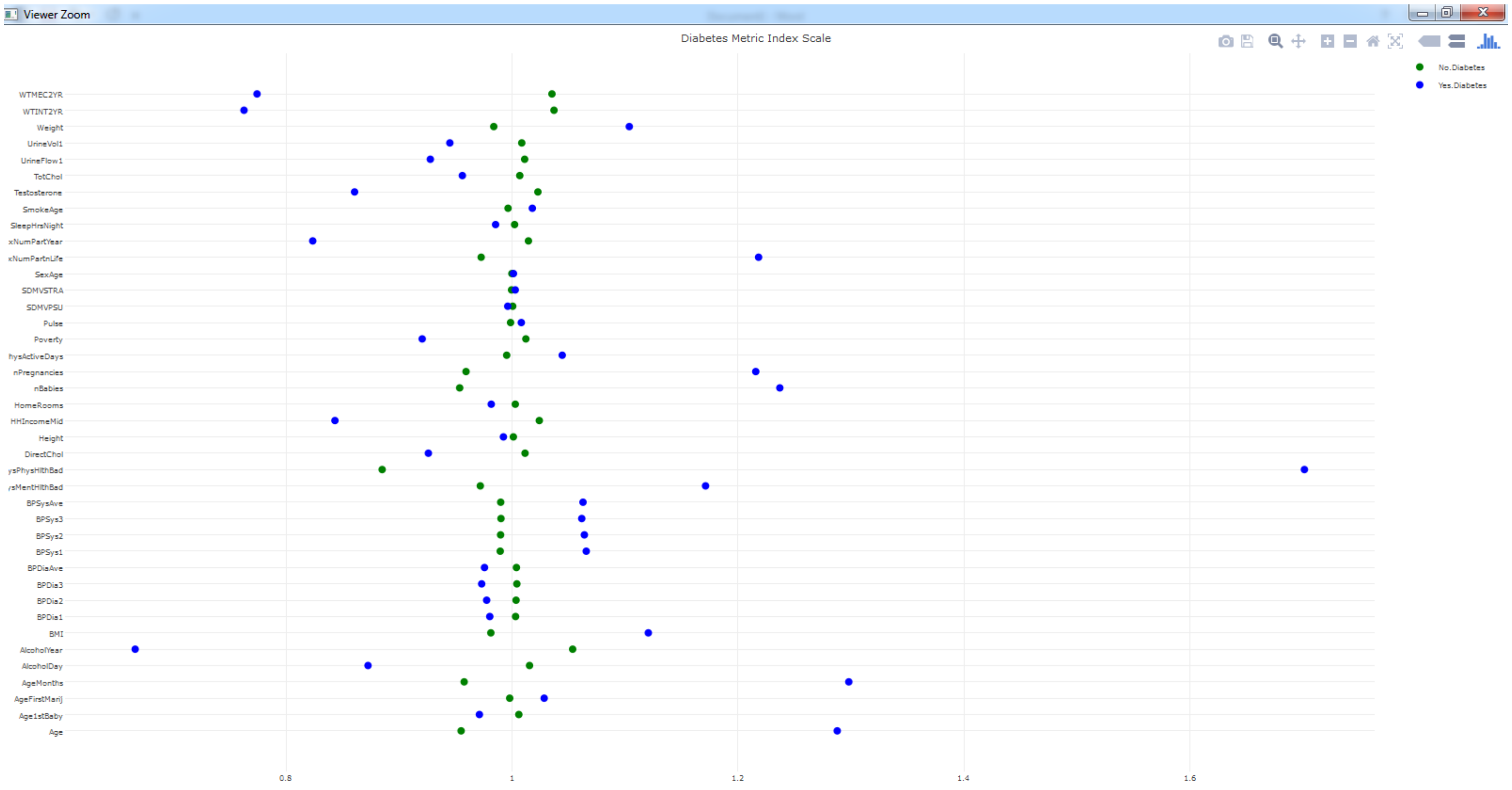
Now, we would like to understand what is the average variance from the central tendency among observations that have diabetes and do not have diabetes by each variable. A good chart for this is a dot plot where we will take each variables, and find the average for diabetes and non-diabetes observations, then index them against the overall variable average. Each group, diabetes vs. no diabetes, will be represented by a colored dot. This will tell use how much or less each group is compared to the overall average.

We will first run on the non-imputed data then run the same on the imputed values.

```
nhanes_numDF$Diabetes <- nhanesDF$Diabetes
nhanes_numDF$Study <- "NHANES"
nhanes_numDF$ID <- NULL

aggDiabetes <-aggregate(nhanes_numDF[,1:40], by=list(nhanes_numDF$Diabetes), FUN=mean,
na.rm=TRUE)
aggNHANES <- aggregate(nhanes_numDF[,1:40], by=list(nhanes_numDF$Study), FUN=mean,
na.rm=TRUE)

library(reshape)
DiabetesMetrics <- melt(aggDiabetes, id=c("Group.1"))
DiabetesMetrics <- cast(DiabetesMetrics, variable ~ Group.1)
colnames(DiabetesMetrics) <- c("Metric", "No.Diabetes", "Yes.Diabetes")

nhanesMetrics <- melt(aggNHANES, id=c("Group.1"))
colnames(nhanesMetrics) <- c("Study", "Metric", "Overall")
nhanesMetrics$Study <- NULL
nhanesMetrics_Final <- merge(DiabetesMetrics, nhanesMetrics, by="Metric")
nhanesMetrics_Final$No.Diabetes.Index <- 0
nhanesMetrics_Final$Yes.Diabetes.Index <- 0

for(i in 1:nrow(nhanesMetrics_Final)){
  nhanesMetrics_Final[i,5] <- nhanesMetrics_Final[i,2]/nhanesMetrics_Final[i,4]
  nhanesMetrics_Final[i,6] <- nhanesMetrics_Final[i,3]/nhanesMetrics_Final[i,4]
}

p <- plot_ly(nhanesMetrics_Final, x = No.Diabetes.Index, y = Metric, name = "No.Diabetes",
             mode = "markers", marker = list(color = "pink", size=8)) %>%
  add_trace(x = Yes.Diabetes.Index, name = "Yes.Diabetes", marker = list(color = "blue", size=8)) %>%
  layout(
    title = "Diabetes Metric Index Scale",
    xaxis = list(title = ""),
    margin = list(l = 60, r=60),
    yaxis = list(title = ""),
    font = list(size=8)
  )
p
```

2. Expository data analytics (EDA) feat. Plotly

Dot plot of variables and diabetes (non-imputed data) (continued…)



From this chart we can see all the numeric variables on the right, the blue dots are those with diabetes mean for that variable, and the green colored dot are those with out diabetes mean for each of the variables.

The scales are read as 1 is the overall mean and any value above or below the overall mean. So for example, we see that age for the green (no diabetes) is at ~.96 which means 4% below average, however the blue dot (with diabetes) indicates they are 28% higher in age than the average (value 1.28).

From this chart then we see that those with diabetes have higher/lower, compared to the total average, the below variables:

- Weight (+10%)
- Sex Number of Praters (+22%)
- Number of Pregnancies (+22%)
- Number of Babies (+23%)
- Days of Physical Bad Health (+70%)
- Days with Mentally Bad Health (+17%)
- All Systolic Blood Pressure Readings (+6%)
- BMI (+12%)

- Alcohol Days (-13%)
- Alcohol Years (-44%)
- Testosterone (-14%)
- All Diastolic Blood Pressure Readings (-3%)
- Urine Volume Readings (-5%)

2. Expository data analytics (EDA) feat. Plotly

EDA conclusions

From this chart we can see all the numeric variables on the right, the blue dots are those with diabetes mean for that variable, and the green colored dot are those with out diabetes mean for each of the variables.

The scales are read as 1 is the overall mean and any value above or below the overall mean. So for example, we see that age for the green (no diabetes) is at ~.96 which means 4% below average, however the blue dot (with diabetes) indicates they are 28% higher in age than the average (value 1.28).

From this chart then we see that those with diabetes have higher/lower, compared to the total average, the below variables:

Clustering NHANES - diabetes analysis

H2O provides kmeans functions for clustering observations into homogeneous groups. The nice feature within H2O is it's ability to handle missing values, so we will run our clustering on both the imputed data set we created in the previous section, and run the clustering on the original numerical fields that include missing values. Then we will compare the results.

From an analytical goals standpoint, what we hope to achieve in this effort is to identify groups of people who are more or less prone to contracting diabetes based solely on the numerical observations.

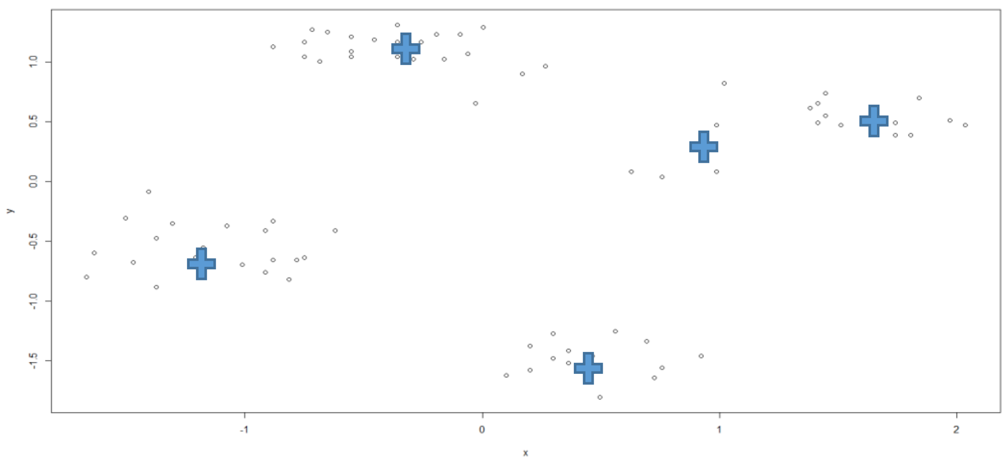
Here is how we will tackle the problem:

1) Overview what clustering is and how kmeans performs these capabilities very briefly (this is not a tutorial to clustering methods. I'll provide links if you need additional references and context)
2) Determine the appropriate number of clusters (even so, still more of an art than a science)
3) Run the kmeans function on both the imputed and NA numerical data sets
4) For each data set, use multidimensional scaling methods to determine separations of each cluster group
5) Analyze the features that make up each cluster to determine the driving forces behind each group
6) Add the cluster group identifiers back to each original data set for predictive analytics we will perform in later sections

Brief overview of clustering analysis

We want to cluster observations into groups to help execute different activities on each group for better maximization of objectives. For example, in our case we want to identify groups with certain characteristics so that when we identify how many within each group has higher / lower observations of diabetes we can perform the needed activates of preventive care on a subset of individuals verses the whole enchilada.

We will use the popular kmeans method to perform our clustering activities within H2O. Kmeans partitions observations based on what is called a centroid, which is a calculation of the groups center. It set a K points as initial centroids, then loops through multiple iterations to find the optimal center point for each group. This type of process in machine learning and statistics is called an unsupervised learning algorithm because we never tell the algorithm a training, or testing data set to learn and tune. It simply attempts to find an optimal point by reduction of the sum of square error for all the observations based on a distance metric (Euclidean, Manhattan, etc.).



Suppose we had the data to the left plotted in a chart. We can use the kmeans method to find the center points and identify a unique cluster identifier to each group that partitions each observation in a homogeneous cluster group. The ✚ is the center point the kmeans method assigned to each group.

Links for more information on kmeans:

https://en.wikipedia.org/wiki/K-means_clustering
http://www.statmethods.net/advstats/cluster.html
https://cran.r-project.org/web/packages/HSAUR/vignettes/Ch_cluster_analysis.pdf

3. Clustering using H2O+R

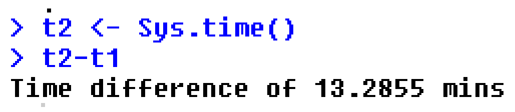
Determine the appropriate number of clusters

Although the aged old question has always been around how many clusters should one use tends to be found out more through trail and error in most cases. We will perform H2O kmeans through an iterative loop to determine the appropriate number of clusters.

The process to loop through multiple K clusters and extracting the needed sum of square error statistics by each K can be computational draining. Luckily H2O in-memory optimization of the kmeans functions allows us to process this much faster verses running it on local versions of kmeans.

The below R code will produce a data frame and supportive plotly graphic to help us determine the number of clusters we should use in this example (this took my h2o instance about 13 minutes to perform 100 loops):

```
## Determine k
dfK <- data.frame(No_Cluster=numeric(),
                  TotWithinSS=numeric(),
                  BetweenSS=numeric(),
                  TotSS=numeric(),
                  stringsAsFactors=FALSE)

k_iterations = 100
t1 <- Sys.time()
for (i in 2:k_iterations){
  kmeans_imp_hex <- h2o.kmeans(training_frame = nhanes_numDF_Imp_hex,
                            k = i, max_iterations = 1000, standardize = T)
  dfK[i-1,1] <- i
  dfK[i-1,2] <- getTotWithinSS(kmeans_imp_hex)
  dfK[i-1,3] <- getTotWithinSS(kmeans_imp_hex)
  dfK[i-1,4] <- getTotWithinSS(kmeans_imp_hex)
}
t2 <- Sys.time()
t2-t1
```



```
> t2 <- Sys.time()
> t2-t1
Time difference of 13.2855 mins
```

The three metric for sum of square error is defined below:

- TotWithinSS: sum of squared distances within each cluster mean
- BetweenSS: sum of squared distances of each cluster mean
- TotSS: sum of squared distances of each data point to the global sample mean

The key benefit of using H2O in this exercise is its ability to cycle through quickly the kmeans function. This process took my computer with H2O only 13 min, verses without H2O would have taken 20 hours. That is very impressive performance considering this is all in-memory processing.

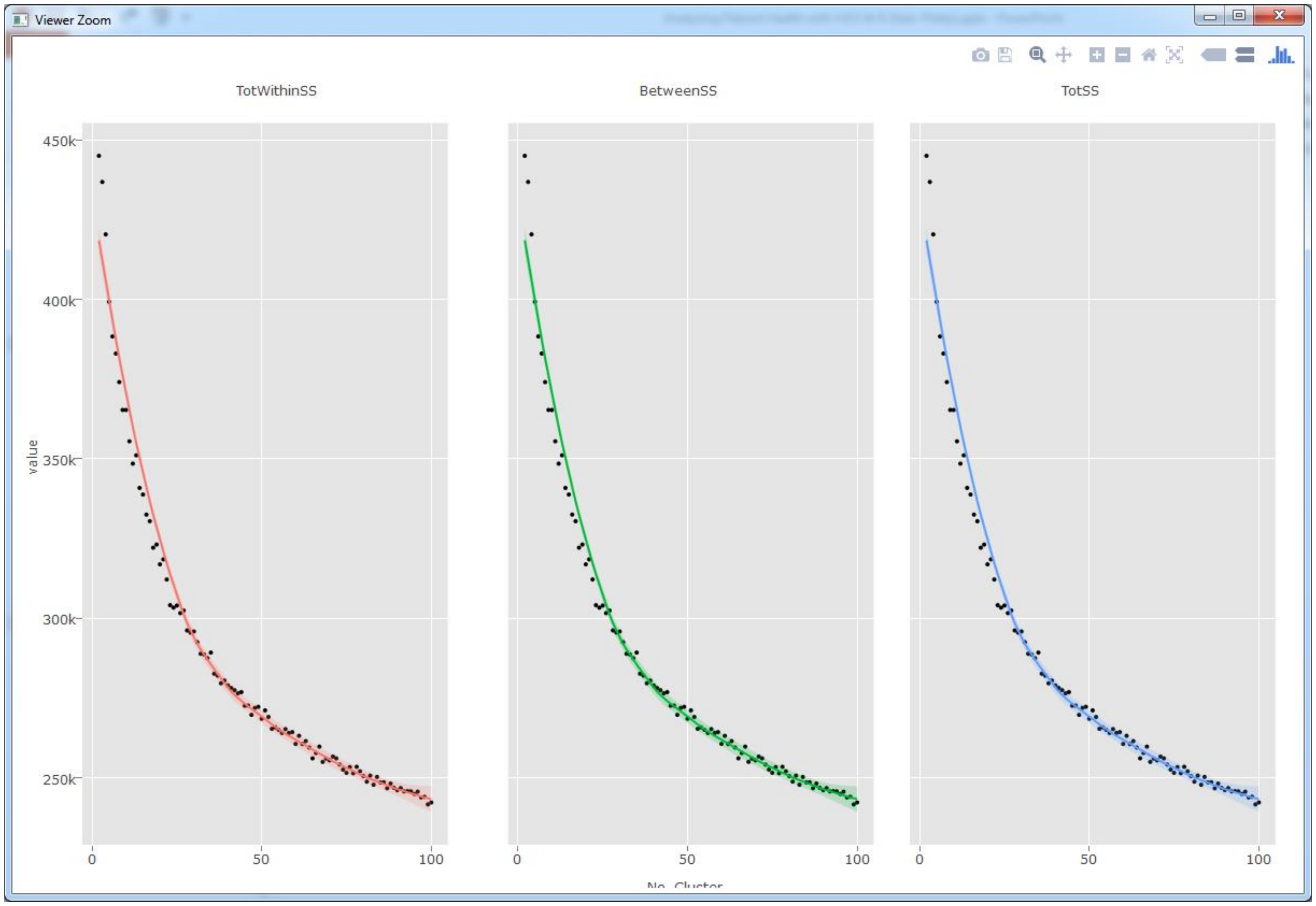
3. Clustering using H2O+R

Determine the appropriate number of clusters (continued…)

The below code charts the three metrics together for a visual presentation by K and the sum of squared distances.

```
library(reshape)
dfK_melt <- melt(dfK, id=c("No_Cluster"))

library(ggplot2)
library(plotly)
p <- ggplot(data = dfK_melt, aes(x = No_Cluster, y = value)) +
  geom_point(aes(text = paste("SS:", variable)), size = 4) +
  geom_smooth(aes(colour = variable, fill = variable)) + facet_wrap(~ variable)

(gg <- ggplotly(p))
```

What we are looking for is the “elbow” of the curve, and therefore will determine the number of clusters we will use. Reason being is that we want to select a K where the sum of squared distances drop is significant but beyond the “elbow” there really isn’t much improvement. We can see in our case we have our “elbow” around K = 40, and therefore that’s what we will use. However, it can be argued that due to the gradual slope of the line after 40 you might be justified in your thinking to keep the K higher. Again, this process of selecting K is more art than science so you need to pick which works best for the context of your project goals.

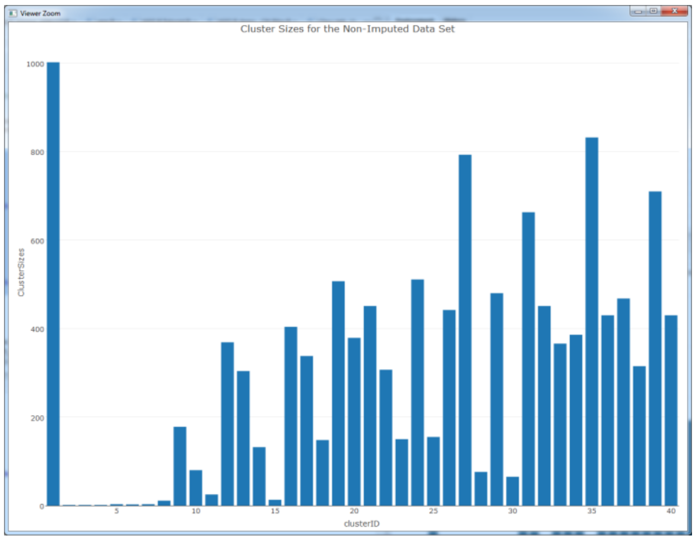


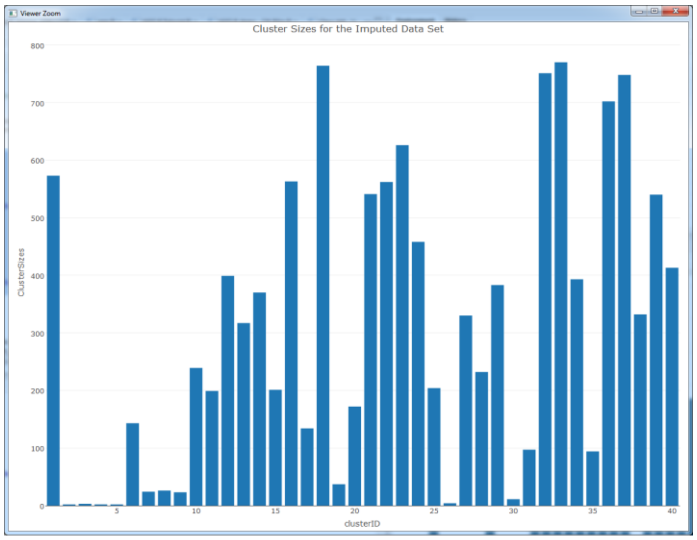
Run kmeans and review cluster sizes

We will not run the version of H2O kmeans with K=40, and review each cluster size.

```
## Run Kmeans
kmeans_hex <- h2o.kmeans(training_frame = nhanes_numDF_hex,
                         k = 40, max_iterations = 1000, standardize = T)
kmeans_imp_hex <- h2o.kmeans(training_frame = nhanes_numDF_Imp_hex,
                         k = 40, max_iterations = 1000, standardize = T)
kmeans_size <- data.frame(ClusterSizes = getClusterSizes(kmeans_hex), clusterID = seq(1:40))
kmeans_imp_size <- data.frame(ClusterSizes = getClusterSizes(kmeans_imp_hex), clusterID = seq(1:40))

library(plotly)
p <- plot_ly(
  data = kmeans_size,
  x = clusterID,
  y = ClusterSizes,
  name = "ClusterSizes non_imp",
  type = "bar"
)%>% layout(title = "Cluster Sizes for the Non-Imputed Data Set", showlegend = FALSE)

p
p <- plot_ly(
  data = kmeans_imp_size,
  x = clusterID,
  y = ClusterSizes,
  name = "ClusterSizes non_imp",
  type = "bar"
)%>% layout(title = "Cluster Sizes for the Imputed Data Set", showlegend = FALSE)
p
```

We can see that we have some cluster with very few observation. Depending on your project, you can determine the best course of action to deal with these cluster with lower levels of observations, but in our case we will leave them to maintain structure (if we got more data to scale for example).





Clustering multidimensional scaling

Next, we wish to see the separation between each cluster considering all the variable centers. To accomplish this, we will use multidimensional scaling (MDS) of the variables combining into a single metric by cluster. The reason for this is that each variable centers are on different scales, and there for will need to be normalized and fitted to a single metric (combining all variable scaled metrics) to determine distance of each cluster.

Notice we are running one dataset that has been imputed and the other has not, which will show how H2O kmeans handles missing values (if you were to run kmeans using R, it would not like the NA's). This is a key advantage of using H2O clustering capabilities.

```
## Get centers
centers_hex <- getCenters(kmeans_hex)
centers_imp_hex <- getCenters(kmeans_imp_hex)

## calc euclidean distance of centers
dist_euclidean <- dist(centers_hex, method = 'euclidean') # Euclidean Distances between the centers
dist_euclidean_imp <- dist(centers_imp_hex, method = 'euclidean') # Euclidean Distances between the centers

## scale distance
fit <- cmdscale(dist_euclidean, eig = TRUE, k = 2) # k is the number of dimensions
fit_imp <- cmdscale(dist_euclidean_imp, eig = TRUE, k = 2) # k is the number of dimensions

# Plot the Clusters in 2D Space
x <- fit$points[ , 1]
y <- fit$points[ , 2]

x_imp <- fit_imp$points[ , 1]
y_imp <- fit_imp$points[ , 2]

par(mfrow=c(1,2))
plot(x, y, xlab = "Coordinate 1", ylab = "Coordinate 2",
    main = "NHANES Clusters - MDS", type = 'n', las = 1)
text(x, y, labels = row.names(centers_hex), cex = .7)

plot(x_imp, y_imp, xlab = "Coordinate 1", ylab = "Coordinate 2",
    main = "NHANES Imputed Clusters - MDS", type = 'n', las = 1)
text(x_imp, y_imp, labels = row.names(centers_hex), cex = .7)
```

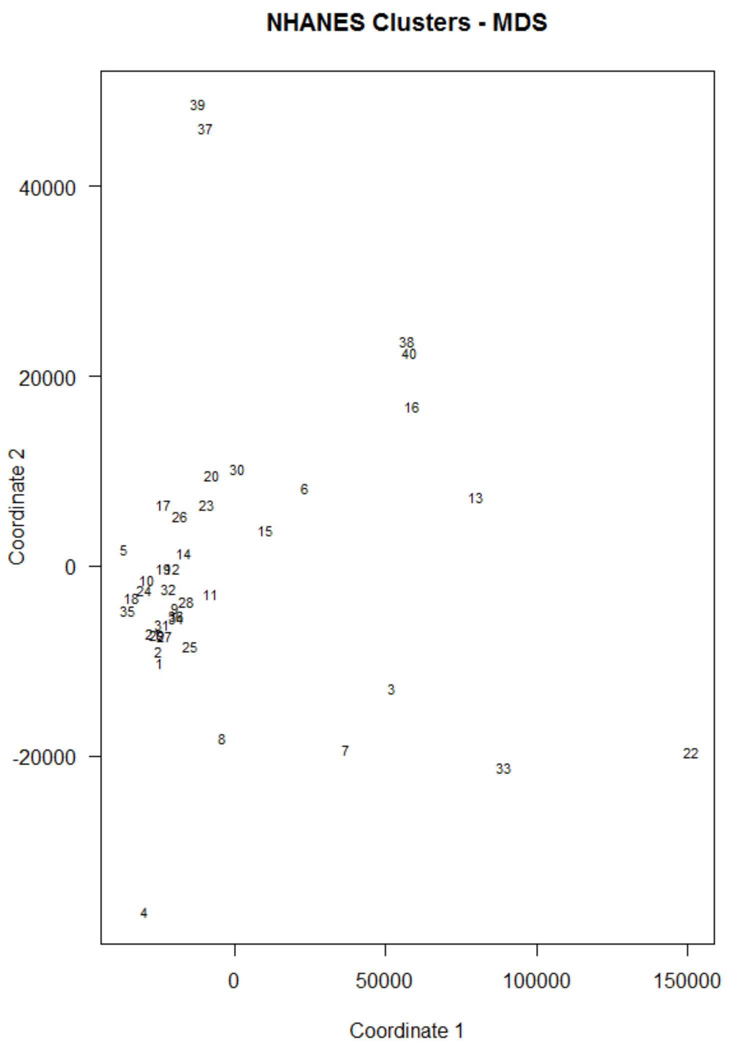
What we did in the above was used the H2O function getCenters() for each data set, then calculated the distance of each using Euclidean distance (you could use others here), scaled each distance by 2 dimensions, then plot the results of each.

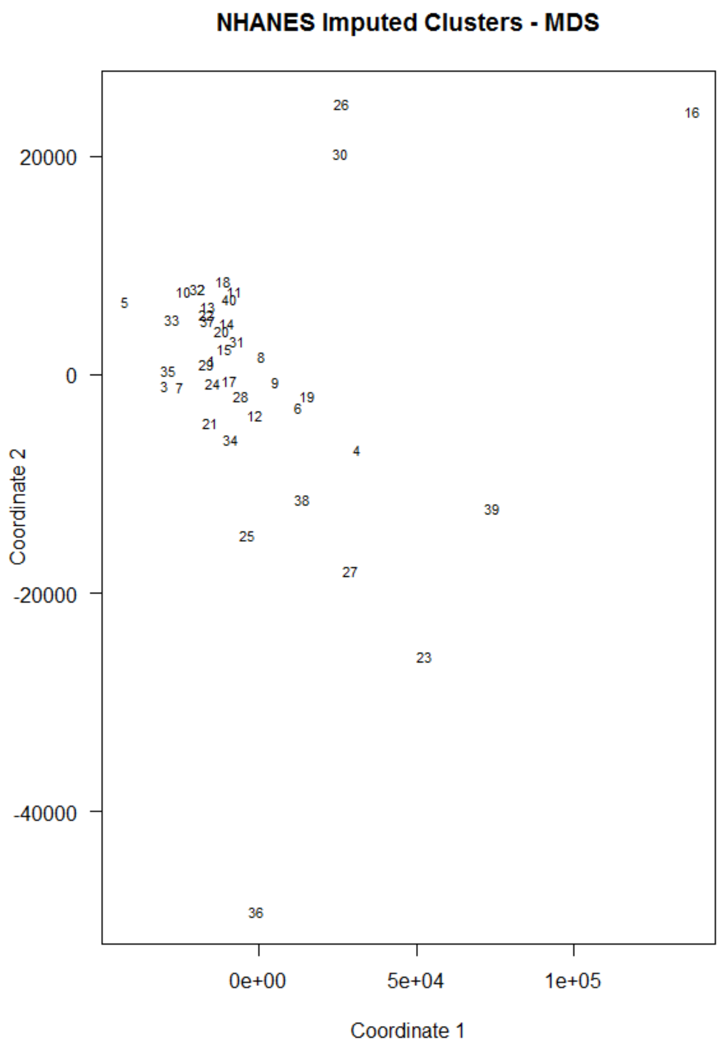
Clustering multidimensional scaling (continued…)

The graph below is the result of our MDS. Don't worry about interpreting the x and y axis (there is no meaning other than for plotting). What this tells us which clusters are different from each other. For example, we can see that on the "NHANES Clusters – MDS" graph that cluster 17 and 26 are very close together. This implies these groups are common to each other with some difference. On the other side, looking again at "NHANES Clusters – MDS" graph, cluster 39 and 4 are very much different groups. We will need to make note of these cluster separations as we attempt to interpret results later.

Another caution is that the cluster number between each graph are not the same, so you cannot say that cluster 4 on the "NHANES Clusters – MDS" graph is the same as cluster 4 on the "NHANES Imputed Clusters – MDS" graph.

Our conclusions in MDS is that we see similar separations and concertation of cluster forms between each data set, which tells us that both clustered data set seem do about the same job (despite having missing values). Although we are not going to do this here, you could go through each observations and determine which cluster group was assign by each data set and calculate a similarity index to show how similar each cluster are to each other (email me and I can share this with you).

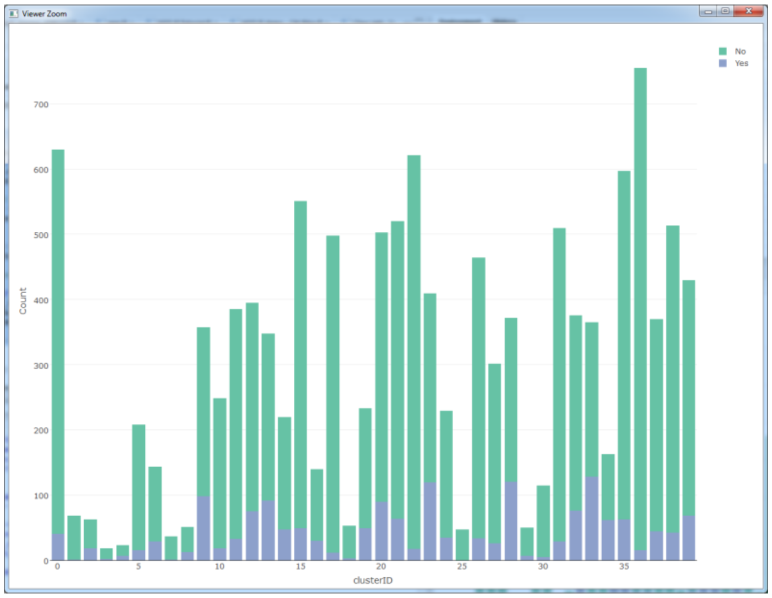




Clustering and diabetes

Now we have our clusters, we want to find the proportions of diabetes within each group. The code below shows graphically the number of diabetes individuals vs. those that do not by cluster.

```
# Combine the incidents with their Clusters
clusters_hex <- h2o.predict(kmeans_hex, nhanes_numDF_hex)
nhanes_numDF_clust_result_hex <- h2o.cbind(nhanes_numDF_hex,clusters_hex)
summary(nhanes_numDF_clust_result_hex)

clusters_imp_hex <- h2o.predict(kmeans_imp_hex, nhanes_numDF_hex)
nhanes_numDF_clust_imp_result_hex <-
h2o.cbind(clusters_imp_hex,nhanes_numDF_Imp_hex,nhanes_numDF_hex$Diabetes)
summary(nhanes_numDF_clust_imp_result_hex)
library(plotly)
library(ggplot2)
nhanes_numDF_clust_result <- as.data.frame(nhanes_numDF_clust_result_hex)
colnames(nhanes_numDF_clust_result)[40] <- "clusterID"
nhanes_numDF_clust_imp_result <- as.data.frame(nhanes_numDF_clust_imp_result_hex)
colnames(nhanes_numDF_clust_imp_result)[1] <- "clusterID"

## set clusterID as factor type (not numeric)
nhanes_numDF_clust_result$clusterID <- as.factor(nhanes_numDF_clust_result$clusterID)
nhanes_numDF_clust_imp_result$clusterID <- as.factor(nhanes_numDF_clust_imp_result$clusterID)

## Add diabetes
nhanes_numDF_clust_result$Diabetes <- nhanesDF_keep$Diabetes
nhanes_numDF_clust_imp_result$Diabetes <- nhanesDF_keep$Diabetes

## Percentage of diabetes per cluster
library(dplyr)
nhanes_numDF_clust_imp_result["Count"] <-1
df2 <- aggregate(nhanes_numDF_clust_imp_result[c("Count")],
            by=list(Diabetes=nhanes_numDF_clust_imp_result$Diabetes,
                clusterID=nhanes_numDF_clust_imp_result$clusterID), FUN=sum, na.rm=TRUE)

p <- plot_ly(df2, x = clusterID, y = Count, type = "bar", color = Diabetes)
layout(p, barmode = "stack")
```



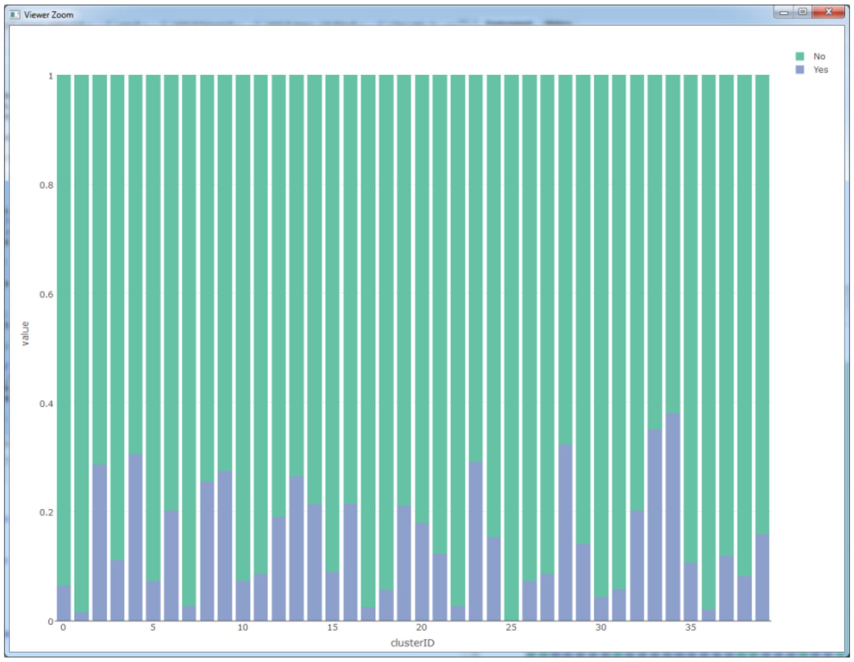
From the graph we see the count of observations by cluster of those with and without diabetes. For example, one of the bigger clusters, cluster 36, shows very little diabetes based on the total number of observations, verses cluster 33 which shows very high amounts of observation of diabetes.

It is difficult to gauge exactly the proportion of diabetes within each cluster, so let's build another graph of the percentage of diabetes within each cluster.

Clustering and diabetes (continued…)

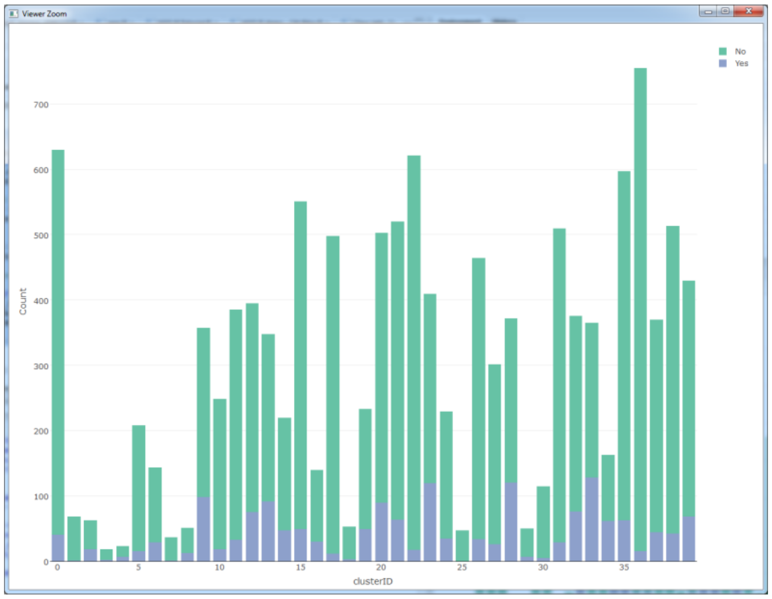
The code below is similar to the previous but shows as percentage vs. counts.

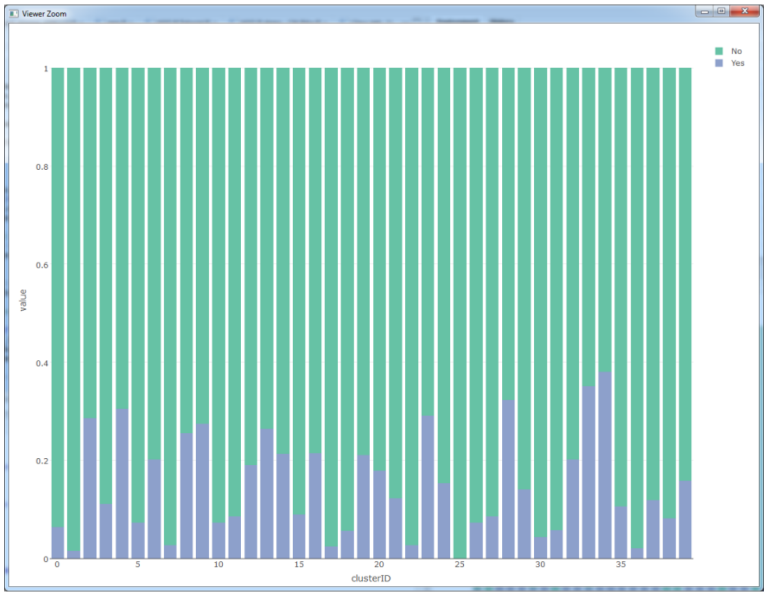
```
df3 <- cast(df2, clusterID~Diabetes)
df3[is.na(df3)] <- 0
df3$NoPCT <- df3$No / (df3$No+df3$Yes)
df3$YesPCT <- 1-df3$NoPCT
df3$No <- NULL
df3$Yes <- NULL
df4 <- melt(df3, id=c("clusterID"))
p <- plot_ly(df4, x = clusterID, y = value, type = "bar", color = Diabetes)
layout(p, barmode = "stack")
```



This graph is much more easily represented for our study to determine the proportion of diabetes found within each cluster. Going back to cluster 33, we see that ~35% of the observations are those with diabetes.

The key question now is what makes each cluster different? When we apply the diabetes proportions we might determine groups of observations and features that are more commonly associated with diabetes.





Clustering summary

In summary, we have used the capabilities of R and H2O to perform clustering by:

1) Briefly reviewing the overall concept of clustering and how H2O capabilities enhances the process of computational grouping observations into homogeneous groups.

2) We then looped through 100 clustering models to then find the appropriate number of cluster, k, by selecting the k where MSE is reduced enough and not really significantly declining after that point. This emphasized the point that using H2O is advantageous when computational intensity raises for looping through multiple interactions of high dimensional data.

3) Next we used multidimensional scaling of the cluster centers by each feature to determine distances from each cluster which indicates just how similar and different each cluster are from one another,

4) Finally, we then added the clustered observations back into the original data set and determined the prevalence of diabetes found within each cluster.

Although its nice that we have these clusters and know what proportions of diabetes are found in each, we need to asses what are the main features that contribute to having diabetes within each of these groups and predicting the likelihood of the diabetes event occurring in each.

We will explore that aspect next using H2O predictive capabilities.

4. Predicting diabetes using H2O+R

Predictive models in H2O

As stated previously, one of our objectives in our analysis is to assess the predictive features that contribute to diabetes. We have built cluster groups, and now want to use the fantastic predictive capabilities H2O has built within R to accomplish this activity.

We will build four types of models used to conduct a predictive model on our target variable diabetes. All four fall within the classification type models as we are attempting to “guess” the class of those with diabetes and those that do not have diabetes.

The four models within H2O we will deploy on our cluster groups are listed below:

1) Distributed Random Forest:
    - An ensemble decision tree method
    - Using predictions from multiple decision trees, the model combines each resulting prediction
    - Each tree gets a vote used in the bagging method
    - It is distributed in terms of the parallelized decision tree process across each H2O clusters

2) Gradient Boosting Machine:
    - Can be either ensemble of either regression or classification tree models, we will use classification tree GBM
    - Is an expansion of the “Multi additive regression tree” of MART combines two methods gradient optimization of loss matrix and boosting for “weak” classifiers producing a committee based approach
    - Uses distributed trees as well

3) General Linear Model:
    - A general linear regression method tool kit for conducting linear models
    - Since our target is binary (diabetes found, not found), we will use a logistic regression approach
    - Uses likelihood optimization in fitting the model parameters

4) Deep Learning:
    - Similar to Neural Networks, deep learner are feedforward Neural Networks with many layers and call fall into other categories such as Deep Belief Network (DBN), Deep Neural Network (DNN), etc.
    - Weights are adapted to minimize the error on the labeled training data

H2O has excellent resources you can read more about these and other models accessing this link: http://www.h2o.ai/resources/

No free lunch approach to modeling means there isn’t just one model that does everything you need, so therefore we will run through all four and compare them on efficacy, accuracy, and parameter setting.

We will supervise all four models as well, setting up training, testing, and a hold out partitioning of each clusters data.

Running Distributed Random Forest by cluster

We will use Distributed Random Forest (DRF) within H2O to predict the class of diabetes by features for each cluster group. Our goal is to extract key features by cluster that contributes to the presence of diabetes. As mentioned before, DRF is an ensemble decision tree method and therefore we the number of trees to use within our model.

One thing we need to be up front and honest to ourselves is that this is a class imbalanced problem (mentioned previously) which will cause us problems in both our assessment of parameter quality and predictive accuracy measures. Let’s dive into the process and study what this means.

First let’s get our data set ready for modeling by only running the model on cluster with greater than 100 observations:

```
library(dplyr)
nhanes_DF_imp_sup <- nhanes_numDF_clust_imp_result
library(sqldf)
cluster_keep <- sqldf("select sum(count) as clust_cnt, clusterID from nhanes_DF_imp_sup group by count, clusterID")
cluster_keep <- cluster_keep[which(cluster_keep$clust_cnt > 100),]
nhanes_DF_imp_sup_keep <- sqldf("select * from nhanes_DF_imp_sup where clusterID in (select clusterID from
cluster_keep)")
nhanes_DF_imp_sup_keep$Diabetes0 <- NULL
nhanes_DF_imp_sup_keep$ID <- NULL
nhanes_DF_imp_sup_keep$Count <- NULL
nhanes_DF_imp_sup_hex <- as.h2o(nhanes_DF_imp_sup_keep)
summary(nhanes_DF_imp_sup_hex)
```

The code above results in a data frame for our supervised (sup) learning models where we have only considered clusters with > 100 observations. We also need a table of all the features listed, so that’s what nhanes_DF_imp_sup_keep_var data frame holds.

Next we create a blank data frame where we insert each model important variables along with key validation statistics by each cluster.

```
model_results_drf <- data.frame(No_Cluster=character(),
                                variable_importances=character(),
                                relative_importance=numeric(),
                                scaled_importance=numeric(),
                                percentage=numeric(),
                                Error_Diabetes=numeric(),
                                stringsAsFactors=FALSE)
```

Running Distributed Random Forest by cluster (continued …)

We will now loop through each cluster (the ones we kept) and produce a random forest model on each cluster, extracting each important feature and saving it to our blank data frame. We also are interested in the confusion matrix error rates on “Yes” to diabetes. Typically, in class imbalance problems, the models produce great results on the training and test data set on the prevalent condition (in our case those without diabetes) but suffer on the end of predicting the lower frequency event (those with diabetes). Our goal is to get the error rate as low as possible on the “Yes” to diabetes, even if that means increasing error on “No”. The reason is that it’s more costly in our case to miss a diabetes diagnoses, vs. miss-diagnosing someone without diabetes, but the models claims they do. This is a common dilemma in healthcare analytics and researchers must always weigh the cost of each study (patient outcomes focused).

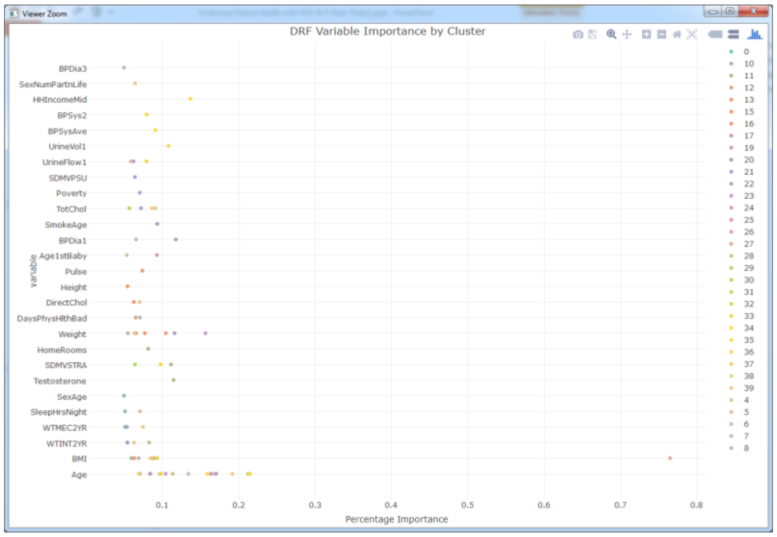
```
t1 <- Sys.time()
for (i in 1:nrow(cluster_keep)){
  clust_ID <- as.character(cluster_keep[i,2])
  clustDF <- nhanes_DF_imp_sup_hex[nhanes_DF_imp_sup_hex$clusterID==clust_ID,]
  clustDF <- h2o.removeVecs(clustDF, c("clusterID","Count"))
  r <- h2o.runif(clustDF)
  train <- clustDF[r < 0.6,]
  test  <- clustDF[(r >= 0.6) & (r < 0.9),]
  hold  <- clustDF[r >= 0.9,]

  response <- "Diabetes"
  predictors <- setdiff(names(clustDF), response)

  try(drf <- h2o.randomForest(x = predictors,
                              y = response,
                              training_frame    = train,
                              validation_frame  = test,
                              ntrees            = 1000,
                              balance_classes   = T), silent = T)

  drf_var_variable_importances <- as.data.frame(drf@model$variable_importances)
  perf_drf <- h2o.performance(drf, clustDF)
  drf_var_variable_importances$Error_Diabetes <- h2o.confusionMatrix(perf_drf)[2,3]
  drf_var_variable_importances$No_Cluster <- clust_ID

  model_results_drf <- rbind(model_results_drf,drf_var_variable_importances)
}
t2 <- Sys.time()
t2-t1
```

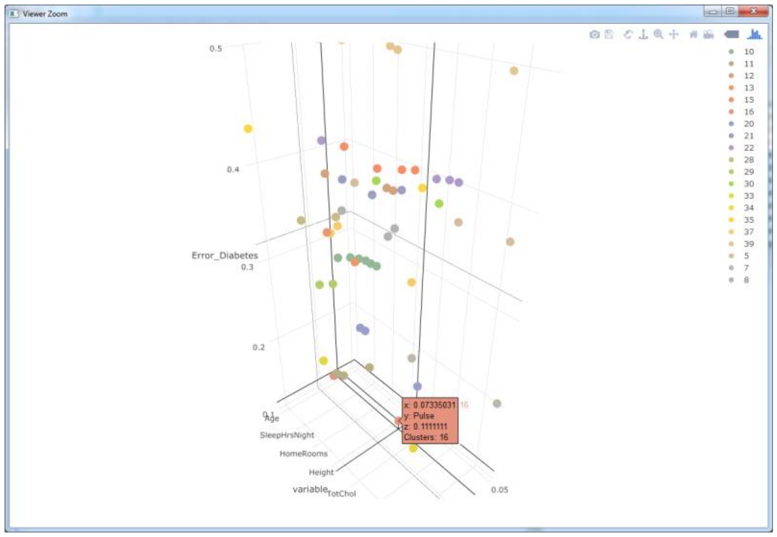
Let’s review the parameters we set here. First, as in all the models we will run, we set the target and predictors, test, and train parameters. Next, ntrees allow us to set the number of trees to cycle through. Remember, RF trees get a vote and each will cycle through for the best result based on reduction of MSE. I selected 1,000 trees which works well with H2O’s paralyzed processing. The balance classes = True lets the DRF know it needs to sample in a certain why due to class imbalance. Finally, We loop through all this and predict based on the Hold data set and grab the diabetes = Yes error rate.

Running Distributed Random Forest by cluster (continued …)

The output of the previous exercise produced a data frame that contains all the variables we have considered and appended the variable importance, error rate on the hold out data fro diabetes = “Yes”, and other statistics by each cluster. We can now take that data frame, filter it down to only include those feature that have high levels of importance as it relates to predicting diabetes for each cluster. I selected 70% and above variable importance just to keep thing simple for graphing the results. Run the code below and lets review the out puts.

```
model_results_drf_filtered <- sqldf("select * from model_results_drf where scaled_importance > .7")
plot_ly(model_results_drf_filtered, x = percentage, y = variable,
        mode = "markers", color = No_Cluster) %>%
  layout( title = "DRF Variable Importance by Cluster",
    xaxis = list(title = "Percentage Importance"),
    margin = list(l = 120))

plot_ly(model_results_drf_filtered, x = percentage, y = variable, z=Error_Diabetes,
        text = paste("Clusters: ", No_Cluster),
        type="scatter3d", mode="markers", color = No_Cluster)%>%
  layout(title = "DRF Variable Importance by Cluster (Imporatnace & Error Rate)",
    xaxis = list(title = "Percentage Importance"),
    margin = list(l = 20))
```





The first graph produces a graph that shows the variables and importance by each cluster. We can see age is a popular feature among the clusters for diabetes which tells us that this is a key feature probably all groups. It appears Cluster 27 puts a lot of focus of importance on BMI (weight and height ratio), which far exceeds the rest of the clusters. If you filter this outlier out by select the rest of the data points, you can see a re-scaled version that easier to read.

The second chart includes the 3rd dimension of diabetes = “Yes” error rate. Since this is 3 dimensional space, you can see variable importance with high levels of importance and low levels of error rate by cluster. From this graph we are able to see for example cluster 5 shows low error rate and high level importance for “Age of 1st Baby”.

Running Gradient Boosting Machine by cluster

As mentioned in the section introduction, Gradient Boosting Machine (GBM) can be used in multiple predictive fashions such as regression or tree based classification. We are interested in the later as we are attempting to classify individuals as either likely to have or contract diabetes or not.

The outputs will be identical to that of what we did for DRF, but there are key difference in the parameters used and the treatment GBM runs through vs. DRF.

First we create again a blank data frame where we will store all our model performance metrics.

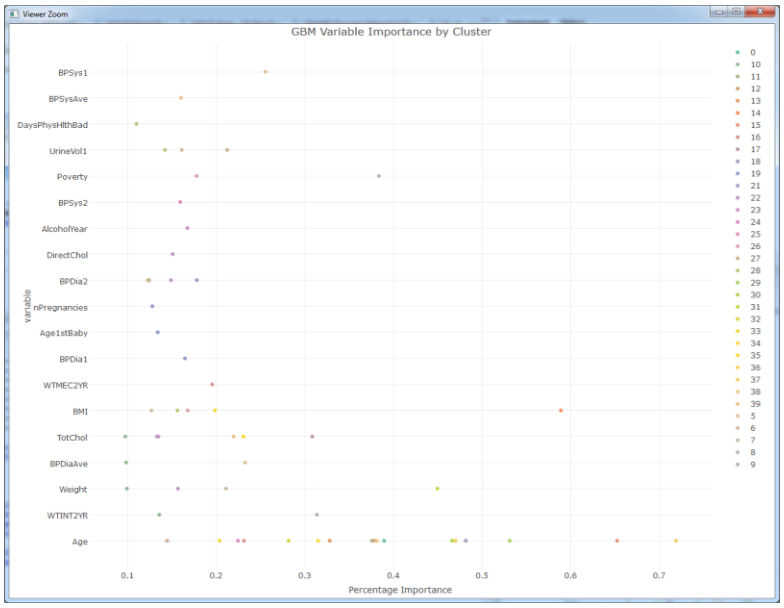
```
model_results_gbm <- data.frame(No_Cluster=character(),
                                variable_importances=character(),
                                relative_importance=numeric(),
                                scaled_importance=numeric(),
                                percentage=numeric(),
                                Error_Diabetes=numeric(),
                                stringsAsFactors=FALSE)
t1 <- Sys.time()
for (i in 1:nrow(cluster_keep)){
  clust_ID <- as.character(cluster_keep[i,2])
  clustDF <- nhanes_DF_imp_sup_hex[nhanes_DF_imp_sup_hex$clusterID==clust_ID,]
  clustDF <- h2o.removeVecs(clustDF, c("clusterID","Count"))
  r <- h2o.runif(clustDF)
  train <- clustDF[r < 0.6,]
  test  <- clustDF[(r >= 0.6) & (r < 0.9),]
  hold  <- clustDF[r >= 0.9,]

  response <- "Diabetes"
  predictors <- setdiff(names(clustDF), response)

  try(gbm <- h2o.gbm(x = predictors,
                     y = response,
                     training_frame     = train,
                     validation_frame   = test,
                     ntrees             = 1000,
                     max_depth          = 6,
                     learn_rate         = 0.1,
                     stopping_rounds    = 1,
                     stopping_tolerance = 0.01,
                     stopping_metric    = "misclassification",
                     balance_classes    = T,
                     seed               = 2000000), silent = T)

  gbm_var_variable_importances <- as.data.frame(gbm@model$variable_importances)
  perf_gbm <- h2o.performance(gbm, clustDF)
  gbm_var_variable_importances$Error_Diabetes <- h2o.confusionMatrix(perf_gbm)[2,3]
  gbm_var_variable_importances$No_Cluster <- clust_ID

  model_results_gbm <- rbind(model_results_gbm,gbm_var_variable_importances)
}
t2 <- Sys.time()
t2-t1
```

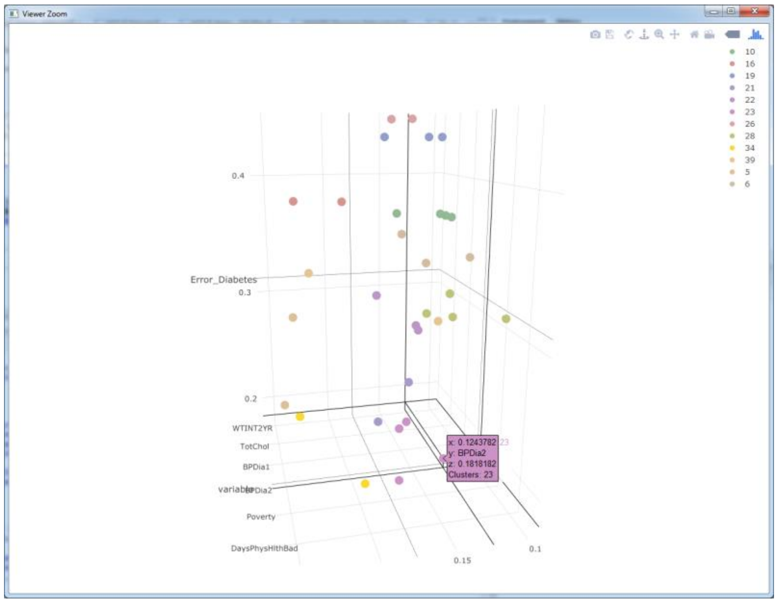
We partition the data into test, train and hold samples for each cluster, same as we did for DRF
ntees is the same, however, in GBM we are penalized for overfitting, so H2O has build what is called grid search models to tune these parameters. We don’t discuss that here, but you can find out more about these capabilities on the H2O resource site I mentioned previously.

Running Gradient Boosting Machine by cluster (continued …)

Again, as we did for the DRF we will look at the features by cluster and value them by importance and error rates.

```
model_results_gbm_filtered <- sqldf("select * from model_results_gbm where scaled_importance > .7")
plot_ly(model_results_gbm_filtered, x = percentage, y = variable,
        mode = "markers", color = No_Cluster) %>%
  layout(
    title = "GBM Variable Importance by Cluster",
    xaxis = list(title = "Percentage Importance"),
    margin = list(l = 120)
  )

plot_ly(model_results_gbm_filtered, x = percentage, y = variable, z=Error_Diabetes,
        text = paste("Clusters: ", No_Cluster),
        type="scatter3d", mode="markers", color = No_Cluster)
```



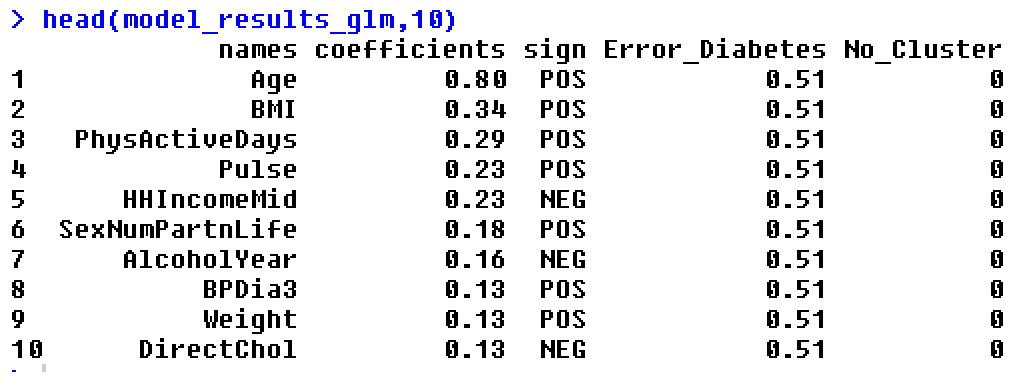


As you can see GBM produces outputs similar to that of DRF except they run much differently. The error rates on average are higher on GBM verses DRF. Running the Grid function on GBM on each cluster might be beneficial to tune each set of models customized for each group.

Running Generalized Linear Model by cluster

Now we will run the Generalized Linear Model (GLM). GLM doesn’t have the same variable importance as with DRF and GBM. What it odes have is a nice feature for standardized coefficient that will help use extract out the important variables by cluster.

```
model_results_glm <- data.frame(No_Cluster=character(),
                                variable_importances=character(),
                                coefficients=numeric(),
                                sign=numeric(),
                                stringsAsFactors=FALSE)

t1 <- Sys.time()
for (i in 1:nrow(cluster_keep)){
  clust_ID <- as.character(cluster_keep[i,2])
  clustDF <- nhanes_DF_imp_sup_hex[nhanes_DF_imp_sup_hex$clusterID==clust_ID,]
  clustDF <- h2o.removeVecs(clustDF, c("clusterID","Count"))
  r <- h2o.runif(clustDF)
  train <- clustDF[r < 0.6,]
  test  <- clustDF[(r >= 0.6) & (r < 0.9),]
  hold  <- clustDF[r >= 0.9,]

  response <- "Diabetes"
  predictors <- setdiff(names(clustDF), response)

  try(glm <- h2o.glm(x = predictors,
                   y = response,
                   training_frame    = train,
                   validation_frame  = test,
                   nfolds            = 5,
                   family            = "binomial"), silent = T)

  glm_var_variable_importances <- as.data.frame(glm@model$standardized_coefficient_magnitudes)
  perf_glm <- h2o.performance(glm, clustDF)
  glm_var_variable_importances$Error_Diabetes <- h2o.confusionMatrix(perf_glm)[2,3]
  glm_var_variable_importances$No_Cluster <- clust_ID

  model_results_glm <- rbind(model_results_glm,glm_var_variable_importances)
}
t2 <- Sys.time()
t2-t1
head(model_results_glm,10)
```

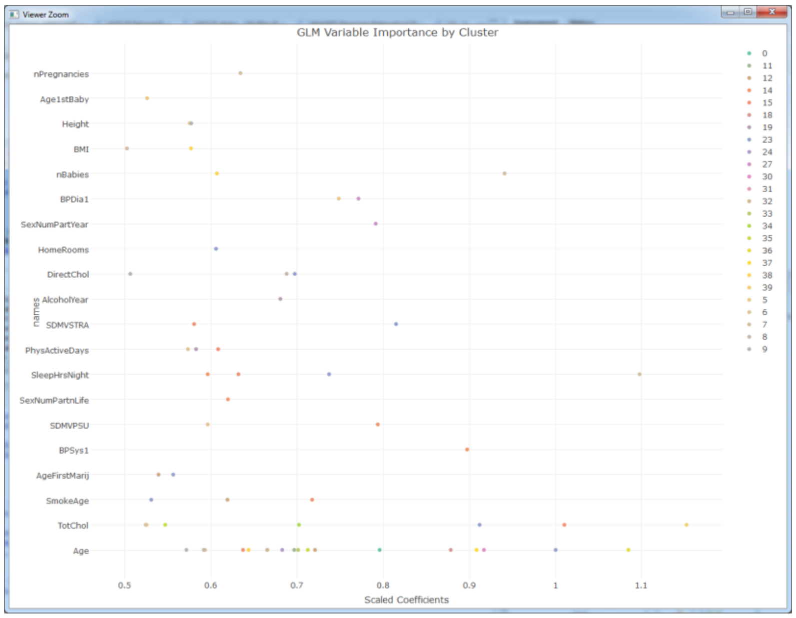


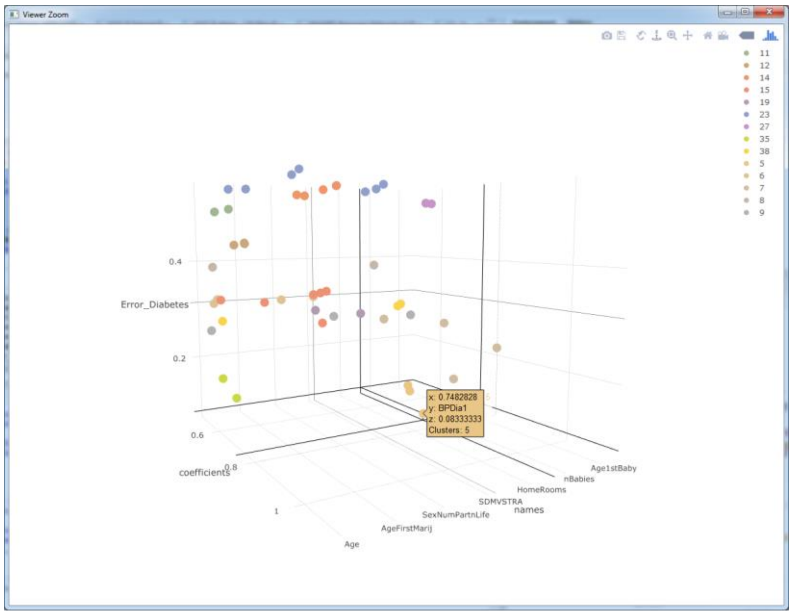
```
> head(model_results_glm,10)
             names coefficients sign Error_Diabetes No_Cluster
1              Age         0.80  POS           0.51          0
2              BMI         0.34  POS           0.51          0
3   PhysActiveDays         0.29  POS           0.51          0
4            Pulse         0.23  POS           0.51          0
5      HHIncomeMid         0.23  NEG           0.51          0
6   SexNumPartnLife        0.18  POS           0.51          0
7      AlcoholYear         0.16  NEG           0.51          0
8           BPDia3         0.13  POS           0.51          0
9           Weight         0.13  POS           0.51          0
10      DirectChol         0.13  NEG           0.51          0
```

Running Generalized Linear Model by cluster (continued …)

The nice thing about the GLM coefficients is that it not only tells the importance, but also the type of relationship the variables has with the target (positive or negative).

```
model_results_glm_filtered <- sqldf("select * from model_results_glm where coefficients > .5")
plot_ly(model_results_glm_filtered, x = coefficients, y = names,
        mode = "markers", color = No_Cluster) %>%
  layout(
    title = "GLM Variable Importance by Cluster",
    xaxis = list(title = "Scaled Coefficients"),
    margin = list(l = 120)
  )

plot_ly(model_results_glm_filtered, x = coefficients, y = names, z=Error_Diabetes,
        text = paste("Clusters: ", No_Cluster),
        type="scatter3d", mode="markers", color = No_Cluster)
```





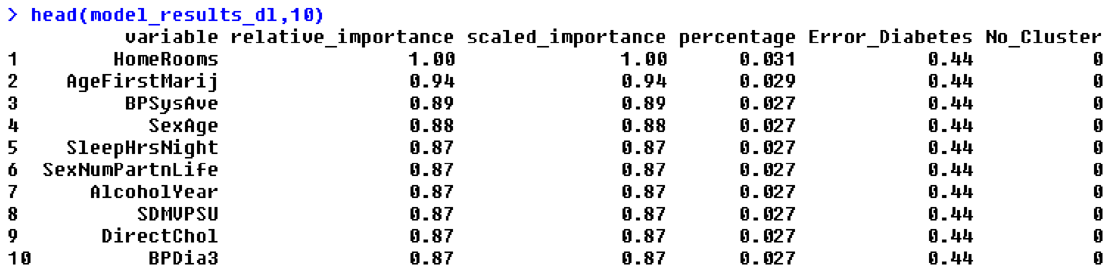
Again, “Age” pops up as the more prevalence variable of importance across most clusters. Interesting that cluster 5 had significantly better error rate on Diabetes = “Yes” than the rest. Number of babies also pops up for cluster 7 which indicates these might be our pregnant mothers with gestational diabetes. Again, not concluded but simply identifying data points of observations. A medical physician would need to diagnose you before any action should be taken.

4. Predicting diabetes using H2O+R

Running Deep Learning by cluster

Now we will run the deep learning model. We specify the number of hidden nodes, tell it’s a imbalanced data set, and tell it we want to get the variable importance from the model outputs.

```
model_results_dl <- data.frame(No_Cluster=character(),
                               variable_importances=character(),
                               relative_importance=numeric(),
                               scaled_importance=numeric(),
                               percentage=numeric(),
                               variable_included=character(),
                               Error_Diabetes=numeric(),
                               stringsAsFactors=FALSE)

t1 <- Sys.time()
for (i in 1:nrow(cluster_keep)){
  clust_ID <- as.character(cluster_keep[i,2])
  clustDF <- nhanes_DF_imp_sup_hex[nhanes_DF_imp_sup_hex$clusterID==clust_ID,]
  clustDF <- h2o.removeVecs(clustDF, c("clusterID","Count"))
  r <- h2o.runif(clustDF)
  train <- clustDF[r < 0.6,]
  test  <- clustDF[(r >= 0.6) & (r < 0.9),]
  hold  <- clustDF[r >= 0.9,]

  response <- "Diabetes"
  predictors <- setdiff(names(clustDF), response)

  try(dl <- h2o.deeplearning(x                  = predictors,
                             y                  = response,
                             training_frame       = train,
                             activation          = "Tanh",
                             balance_classes      = TRUE,
                             input_dropout_ratio  = 0.2,
                             hidden              = c(100, 100, 100, 100, 100),
                             epochs              = 10,
                             variable_importances = T), silent = T)

  dl_var_variable_importances <- as.data.frame(dl@model$variable_importances)
  perf_dl <- h2o.performance(dl, clustDF)
  dl_var_variable_importances$Error_Diabetes <- h2o.confusionMatrix(perf_dl)[2,3]
  dl_var_variable_importances$No_Cluster <- clust_ID

  model_results_dl <- rbind(model_results_dl,dl_var_variable_importances)
}
t2 <- Sys.time()
t2-t1
head(model_results_dl,10)
```



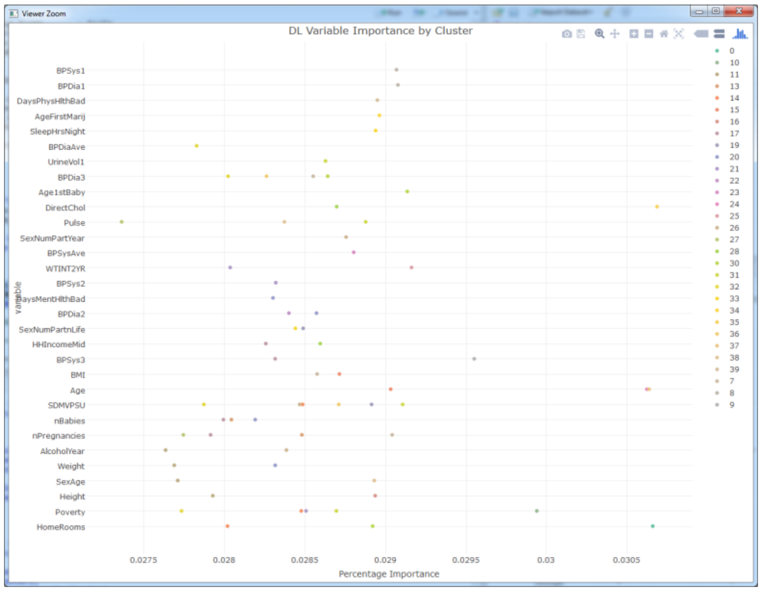
```
> head(model_results_dl,10)
          variable relative_importance scaled_importance percentage Error_Diabetes No_Cluster
1        HomeRooms                1.00              1.00      0.031           0.44          0
2    AgeFirstMarij                0.94              0.94      0.029           0.44          0
3        BPSysAve                0.89              0.89      0.027           0.44          0
4          SexAge                0.88              0.88      0.027           0.44          0
5    SleepHrsNight                0.87              0.87      0.027           0.44          0
6  SexNumPartnLife                0.87              0.87      0.027           0.44          0
7      AlcoholYear                0.87              0.87      0.027           0.44          0
8         SDMVPSU                0.87              0.87      0.027           0.44          0
9       DirectChol                0.87              0.87      0.027           0.44          0
10          BPDia3                0.87              0.87      0.027           0.44          0
```

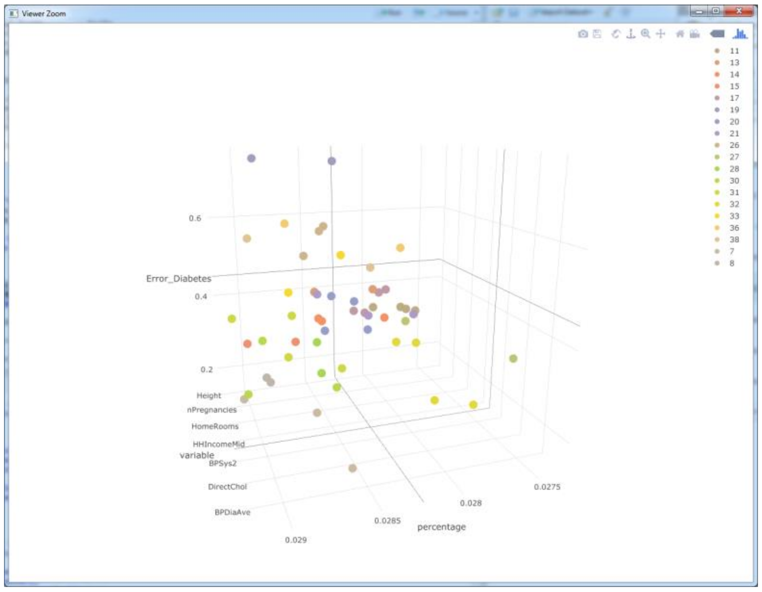
4. Predicting diabetes using H2O+R

Running Deep Learning by cluster (continued…)

Let’s take a look at our variable importance like we have done on the others.

```
model_results_dl_filtered <- sqldf("select * from model_results_dl where scaled_importance > .98")
plot_ly(model_results_dl_filtered, x = percentage, y = variable,
        mode = "markers", color = No_Cluster) %>%
  layout(
    title = "DL Variable Importance by Cluster",
    xaxis = list(title = "Percentage Importance"),
    margin = list(l = 120)
  )

plot_ly(model_results_dl_filtered, x = percentage, y = variable, z=Error_Diabetes,
        text = paste("Clusters: ", No_Cluster),
        type="scatter3d", mode="markers", color = No_Cluster)
```





We see once again the important variables by each cluster in the left graph and the 3D graph that includes error rate of the diabetes = “Yes” category.

Bringing it all together

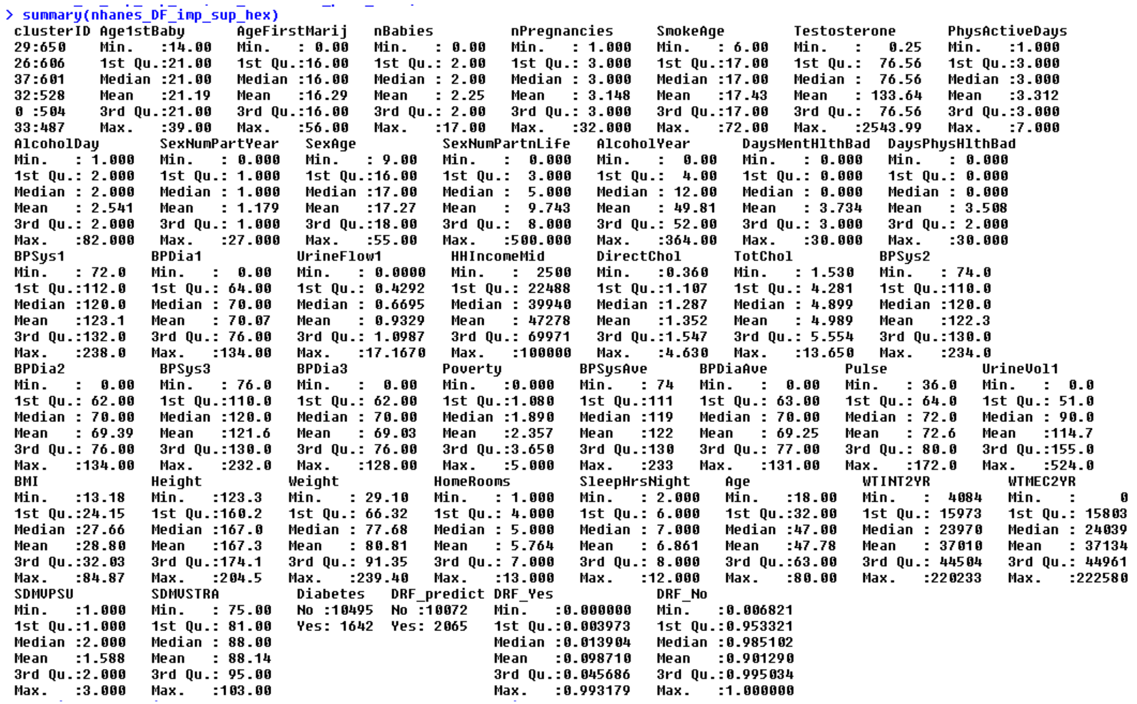
Now that we have an understanding of the features/variables that drive each of the clusters across four different models, we will now run all four models on the entire imputed data set.

This will be similar to the process we did earlier with exception to adding the predictive outputs back into the original data set.

First, we run the Distributed Random Forest:

```
## *************** DRF ****************************************
str(nhanes_DF_imp_sup_hex)
r <- h2o.runif(nhanes_DF_imp_sup_hex)
train <- nhanes_DF_imp_sup_hex[r < 0.6,]
test  <- nhanes_DF_imp_sup_hex[(r >= 0.6) & (r < 0.9),]
hold  <- nhanes_DF_imp_sup_hex[r >= 0.9,]
response <- "Diabetes"
predictors <- setdiff(names(nhanes_DF_imp_sup_hex), response)

drf <- h2o.randomForest(x = predictors, y = response,
                  training_frame     = train,
                  validation_frame  = test,
                  ntrees            = 1000,
                  balance_classes   = T)

drf
drf_pred_hex <- h2o.predict(drf, newdata=hold)
perf_drf <- h2o.performance(drf, nhanes_DF_imp_sup_hex)
h2o.confusionMatrix(perf_drf)

drf_pred_hex2 <- h2o.predict(drf, newdata=nhanes_DF_imp_sup_hex)
nhanes_DF_imp_sup_hex$DRF_predict <- drf_pred_hex2$predict
nhanes_DF_imp_sup_hex$DRF_Yes <- drf_pred_hex2$Yes
nhanes_DF_imp_sup_hex$DRF_No <- drf_pred_hex2$No
```

You can see the new fields added to the data site at the end.



```
> summary(nhanes_DF_imp_sup_hex)
 clusterID Age1stBaby       AgeFirstMarij    nBabies           nPregnancies      SmokeAge         Testosterone       PhysActiveDays
 29:650    Min.   :14.00    Min.   : 0.00    Min.   : 0.00     Min.   : 1.000    Min.   : 6.00    Min.   :    0.25   Min.   :1.000
 26:606    1st Qu.:21.00    1st Qu.:16.00    1st Qu.: 2.00     1st Qu.: 3.000    1st Qu.:17.00    1st Qu.:   76.56   1st Qu.:3.000
 37:601    Median :21.00    Median :16.00    Median : 2.00     Median : 3.000    Median :17.00    Median :   76.56   Median :3.000
 32:528    Mean   :21.19    Mean   :16.29    Mean   : 2.25     Mean   : 3.148    Mean   :17.43    Mean   :  133.64   Mean   :3.312
 0 :504    3rd Qu.:21.00    3rd Qu.:16.00    3rd Qu.: 2.00     3rd Qu.: 3.000    3rd Qu.:17.00    3rd Qu.:   76.56   3rd Qu.:3.000
 33:487    Max.   :39.00    Max.   :56.00    Max.   :17.00     Max.   :32.000    Max.   :72.00    Max.   :2543.99    Max.   :7.000
 AlcoholDay        SexNumPartYear    SexAge          SexNumPartnLife    AlcoholYear      DaysMentHlthBad   DaysPhysHlthBad
 Min.   : 1.000    Min.   : 0.000    Min.   : 9.00   Min.   :  0.000    Min.   :  0.00   Min.   : 0.000    Min.   : 0.000
 1st Qu.: 2.000    1st Qu.: 1.000    1st Qu.:16.00   1st Qu.:  3.000    1st Qu.:  4.00   1st Qu.: 0.000    1st Qu.: 0.000
 Median : 2.000    Median : 1.000    Median :17.00   Median :  5.000    Median : 12.00   Median : 0.000    Median : 0.000
 Mean   : 2.541    Mean   : 1.179    Mean   :17.27   Mean   :  9.743    Mean   : 49.81   Mean   : 3.734    Mean   : 3.508
 3rd Qu.: 2.000    3rd Qu.: 1.000    3rd Qu.:18.00   3rd Qu.:  8.000    3rd Qu.: 52.00   3rd Qu.: 3.000    3rd Qu.: 2.000
 Max.   :82.000    Max.   :27.000    Max.   :55.00   Max.   :500.000    Max.   :364.00   Max.   :30.000    Max.   :30.000
 BPSys1           BPDia1           UrineFlow1        HHIncomeMid       DirectChol      TotChol          BPSys2
 Min.   : 72.0    Min.   :  0.00   Min.   : 0.0000   Min.   :  2500    Min.   :0.360   Min.   : 1.530   Min.   : 74.0
 1st Qu.:112.0    1st Qu.: 64.00   1st Qu.: 0.4292   1st Qu.: 22488    1st Qu.:1.107   1st Qu.: 4.281   1st Qu.:110.0
 Median :120.0    Median : 70.00   Median : 0.6695   Median : 39940    Median :1.287   Median : 4.899   Median :120.0
 Mean   :123.1    Mean   : 70.07   Mean   : 0.9329   Mean   : 47278    Mean   :1.352   Mean   : 4.989   Mean   :122.3
 3rd Qu.:132.0    3rd Qu.: 76.00   3rd Qu.: 1.0987   3rd Qu.: 69971    3rd Qu.:1.547   3rd Qu.: 5.554   3rd Qu.:130.0
 Max.   :238.0    Max.   :134.00   Max.   :17.1670   Max.   :100000    Max.   :4.630   Max.   :13.650   Max.   :234.0
 BPDia2           BPSys3           BPDia3           Poverty          BPSysAve       BPDiaAve          Pulse           UrineVol1
 Min.   :  0.00   Min.   : 76.0    Min.   :  0.00   Min.   :0.000    Min.   : 74    Min.   :  0.00    Min.   : 36.0   Min.   :  0.0
 1st Qu.: 62.00   1st Qu.:110.0    1st Qu.: 62.00   1st Qu.:1.080    1st Qu.:111    1st Qu.: 63.00    1st Qu.: 64.0   1st Qu.: 51.0
 Median : 70.00   Median :120.0    Median : 70.00   Median :1.890    Median :119    Median : 70.00    Median : 72.0   Median : 90.0
 Mean   : 69.39   Mean   :121.6    Mean   : 69.03   Mean   :2.357    Mean   :122    Mean   : 69.25    Mean   : 72.6   Mean   :114.7
 3rd Qu.: 76.00   3rd Qu.:130.0    3rd Qu.: 76.00   3rd Qu.:3.650    3rd Qu.:130    3rd Qu.: 77.00    3rd Qu.: 80.0   3rd Qu.:155.0
 Max.   :134.00   Max.   :232.0    Max.   :128.00   Max.   :5.000    Max.   :233    Max.   :131.00    Max.   :172.0   Max.   :524.0
 BMI              Height           Weight           HomeRooms         SleepHrsNight     Age              WTINT2YR          WTMEC2YR
 Min.   :13.18    Min.   :123.3    Min.   : 29.10   Min.   : 1.000    Min.   : 2.000    Min.   :18.00    Min.   :  4084    Min.   :     0
 1st Qu.:24.15    1st Qu.:160.2    1st Qu.: 66.32   1st Qu.: 4.000    1st Qu.: 6.000    1st Qu.:32.00    1st Qu.: 15973    1st Qu.: 15803
 Median :27.66    Median :167.0    Median : 77.68   Median : 5.000    Median : 7.000    Median :47.00    Median : 23970    Median : 24039
 Mean   :28.80    Mean   :167.3    Mean   : 80.81   Mean   : 5.764    Mean   : 6.861    Mean   :47.78    Mean   : 37010    Mean   : 37134
 3rd Qu.:32.03    3rd Qu.:174.1    3rd Qu.: 91.35   3rd Qu.: 7.000    3rd Qu.: 8.000    3rd Qu.:63.00    3rd Qu.: 44504    3rd Qu.: 44961
 Max.   :84.87    Max.   :204.5    Max.   :239.40   Max.   :13.000    Max.   :12.000    Max.   :80.00    Max.   :220233    Max.   :222580
 SDMVPSU          SDMVSTRA         Diabetes    DRF_predict DRF_Yes              DRF_No
 Min.   :1.000    Min.   : 75.00   No :10495   No :10072   Min.   :0.000000     Min.   :0.006821
 1st Qu.:1.000    1st Qu.: 81.00   Yes: 1642   Yes: 2065   1st Qu.:0.003973     1st Qu.:0.953321
 Median :2.000    Median : 88.00                           Median :0.013904     Median :0.985102
 Mean   :1.588    Mean   : 88.14                           Mean   :0.098710     Mean   :0.901290
 3rd Qu.:2.000    3rd Qu.: 95.00                           3rd Qu.:0.045686     3rd Qu.:0.995034
 Max.   :3.000    Max.   :103.00                           Max.   :0.993179     Max.   :1.000000
```

Bringing it all together (continued…)

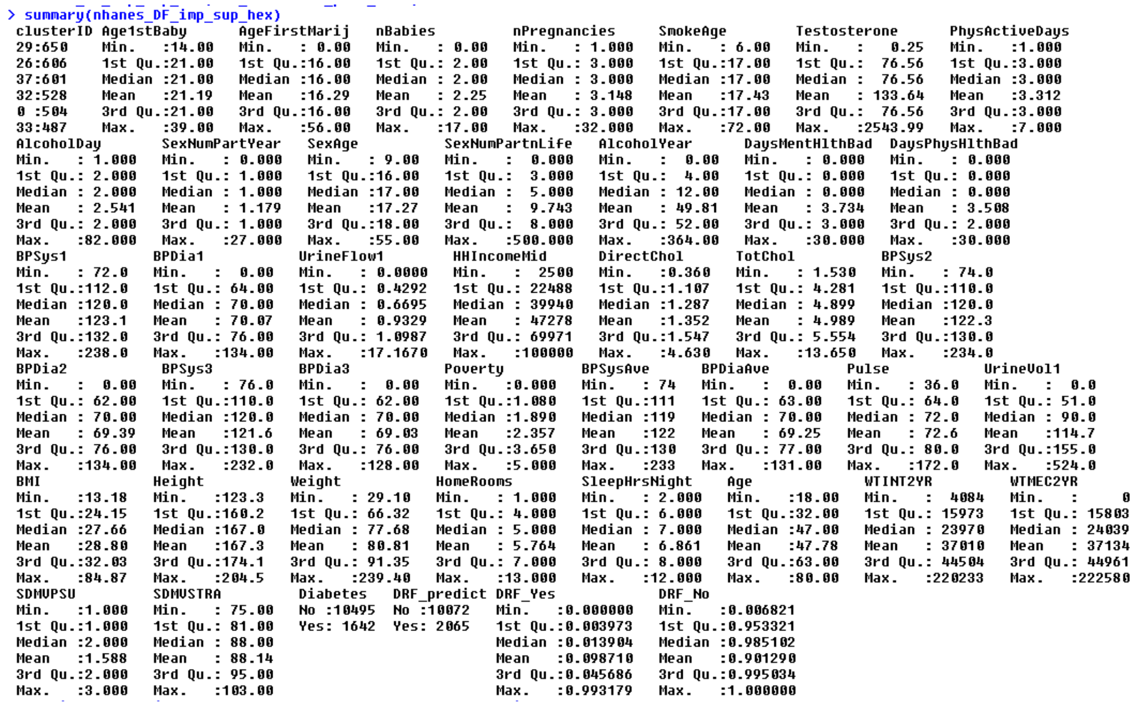
Now that we have an understanding of the features/variables that drive each of the clusters across four different models, we will now run all four models on the entire imputed data set.

This will be similar to the process we did earlier with exception to adding the predictive outputs back into the original data set.

First, we run the Distributed Random Forest:

```
## *************** DRF ****************************************
str(nhanes_DF_imp_sup_hex)
r <- h2o.runif(nhanes_DF_imp_sup_hex)
train <- nhanes_DF_imp_sup_hex[r < 0.6,]
test  <- nhanes_DF_imp_sup_hex[(r >= 0.6) & (r < 0.9),]
hold  <- nhanes_DF_imp_sup_hex[r >= 0.9,]
response <- "Diabetes"
predictors <- setdiff(names(nhanes_DF_imp_sup_hex), response)

drf <- h2o.randomForest(x = predictors, y = response,
                training_frame    = train,
                validation_frame  = test,
                ntrees            = 1000,
                balance_classes   = T)

drf
drf_pred_hex <- h2o.predict(drf, newdata=hold)
perf_drf <- h2o.performance(drf, nhanes_DF_imp_sup_hex)
h2o.confusionMatrix(perf_drf)

drf_pred_hex2 <- h2o.predict(drf, newdata=nhanes_DF_imp_sup_hex)
nhanes_DF_imp_sup_hex$DRF_predict <- drf_pred_hex2$predict
nhanes_DF_imp_sup_hex$DRF_Yes <- drf_pred_hex2$Yes
nhanes_DF_imp_sup_hex$DRF_No <- drf_pred_hex2$No
```

You can see the new fields added to the data site at the end.



```
> summary(nhanes_DF_imp_sup_hex)
 clusterID Age1stBaby      AgeFirstMarij   nBabies         nPregnancies     SmokeAge        Testosterone      PhysActiveDays
 29:650    Min.   :14.00   Min.   : 0.00   Min.   : 0.00   Min.   : 1.000   Min.   : 6.00   Min.   :   0.25   Min.   :1.000
 26:606    1st Qu.:21.00   1st Qu.:16.00   1st Qu.: 2.00   1st Qu.: 3.000   1st Qu.:17.00   1st Qu.:  76.56   1st Qu.:3.000
 37:601    Median :21.00   Median :16.00   Median : 2.00   Median : 3.000   Median :17.00   Median :  76.56   Median :3.000
 32:528    Mean   :21.19   Mean   :16.29   Mean   : 2.25   Mean   : 3.148   Mean   :17.43   Mean   : 133.64   Mean   :3.312
 0 :504    3rd Qu.:21.00   3rd Qu.:16.00   3rd Qu.: 2.00   3rd Qu.: 3.000   3rd Qu.:17.00   3rd Qu.:  76.56   3rd Qu.:3.000
 33:487    Max.   :39.00   Max.   :56.00   Max.   :17.00   Max.   :32.000   Max.   :72.00   Max.   :2543.99   Max.   :7.000
 AlcoholDay        SexNumPartYear    SexAge          SexNumPartnLife    AlcoholYear       DaysMentHlthBad   DaysPhysHlthBad
 Min.   : 1.000   Min.   : 0.000   Min.   : 9.00   Min.   :  0.000   Min.   :  0.00   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.:16.00   1st Qu.:  3.000   1st Qu.:  4.00   1st Qu.: 0.000   1st Qu.: 0.000
 Median : 2.000   Median : 1.000   Median :17.00   Median :  5.000   Median : 12.00   Median : 0.000   Median : 0.000
 Mean   : 2.541   Mean   : 1.179   Mean   :17.27   Mean   :  9.743   Mean   : 49.81   Mean   : 3.734   Mean   : 3.508
 3rd Qu.: 2.000   3rd Qu.: 1.000   3rd Qu.:18.00   3rd Qu.:  8.000   3rd Qu.: 52.00   3rd Qu.: 3.000   3rd Qu.: 2.000
 Max.   :82.000   Max.   :27.000   Max.   :55.00   Max.   :500.000   Max.   :364.00   Max.   :30.000   Max.   :30.000
 BPSys1          BPDia1           UrineFlow1         HHIncomeMid      DirectChol      TotChol          BPSys2
 Min.   : 72.0   Min.   :  0.00   Min.   : 0.0000   Min.   :  2500   Min.   :0.360   Min.   : 1.530   Min.   : 74.0
 1st Qu.:112.0   1st Qu.: 64.00   1st Qu.: 0.4292   1st Qu.: 22488   1st Qu.:1.107   1st Qu.: 4.281   1st Qu.:110.0
 Median :120.0   Median : 70.00   Median : 0.6695   Median : 39940   Median :1.287   Median : 4.899   Median :120.0
 Mean   :123.1   Mean   : 70.07   Mean   : 0.9329   Mean   : 47278   Mean   :1.352   Mean   : 4.989   Mean   :122.3
 3rd Qu.:132.0   3rd Qu.: 76.00   3rd Qu.: 1.0987   3rd Qu.: 69971   3rd Qu.:1.547   3rd Qu.: 5.554   3rd Qu.:130.0
 Max.   :238.0   Max.   :134.00   Max.   :17.1670   Max.   :100000   Max.   :4.630   Max.   :13.650   Max.   :234.0
 BPDia2           BPSys3          BPDia3           Poverty          BPSysAve        BPDiaAve          Pulse           UrineVol1
 Min.   :  0.00   Min.   : 76.0   Min.   :  0.00   Min.   :0.000   Min.   : 74   Min.   :  0.00   Min.   : 36.0   Min.   :  0.0
 1st Qu.: 62.00   1st Qu.:110.0   1st Qu.: 62.00   1st Qu.:1.080   1st Qu.:111   1st Qu.: 63.00   1st Qu.: 64.0   1st Qu.: 51.0
 Median : 70.00   Median :120.0   Median : 70.00   Median :1.890   Median :119   Median : 70.00   Median : 72.0   Median : 90.0
 Mean   : 69.39   Mean   :121.6   Mean   : 69.03   Mean   :2.357   Mean   :122   Mean   : 69.25   Mean   : 72.6   Mean   :114.7
 3rd Qu.: 76.00   3rd Qu.:130.0   3rd Qu.: 76.00   3rd Qu.:3.650   3rd Qu.:130   3rd Qu.: 77.00   3rd Qu.: 80.0   3rd Qu.:155.0
 Max.   :134.00   Max.   :232.0   Max.   :128.00   Max.   :5.000   Max.   :233   Max.   :131.00   Max.   :172.0   Max.   :524.0
 BMI             Height          Weight           HomeRooms        SleepHrsNight    Age             WTINT2YR         WTMEC2YR
 Min.   :13.18   Min.   :123.3   Min.   : 29.10   Min.   : 1.000   Min.   : 2.000   Min.   :18.00   Min.   :  4084   Min.   :     0
 1st Qu.:24.15   1st Qu.:160.2   1st Qu.: 66.32   1st Qu.: 4.000   1st Qu.: 6.000   1st Qu.:32.00   1st Qu.: 15973   1st Qu.: 15803
 Median :27.66   Median :167.0   Median : 77.68   Median : 5.000   Median : 7.000   Median :47.00   Median : 23970   Median : 24039
 Mean   :28.80   Mean   :167.3   Mean   : 80.81   Mean   : 5.764   Mean   : 6.861   Mean   :47.78   Mean   : 37010   Mean   : 37134
 3rd Qu.:32.03   3rd Qu.:174.1   3rd Qu.: 91.35   3rd Qu.: 7.000   3rd Qu.: 8.000   3rd Qu.:63.00   3rd Qu.: 44504   3rd Qu.: 44961
 Max.   :84.87   Max.   :204.5   Max.   :239.40   Max.   :13.000   Max.   :12.000   Max.   :80.00   Max.   :220233   Max.   :222580
 SDMVPSU         SDMVSTRA         Diabetes   DRF_predict DRF_Yes              DRF_No
 Min.   :1.000   Min.   : 75.00   No :10495  No :10072   Min.   :0.000000   Min.   :0.006821
 1st Qu.:1.000   1st Qu.: 81.00   Yes: 1642  Yes: 2065   1st Qu.:0.003973   1st Qu.:0.953321
 Median :2.000   Median : 88.00                          Median :0.013904   Median :0.985102
 Mean   :1.588   Mean   : 88.14                          Mean   :0.098710   Mean   :0.901290
 3rd Qu.:2.000   3rd Qu.: 95.00                          3rd Qu.:0.045686   3rd Qu.:0.995034
 Max.   :3.000   Max.   :103.00                          Max.   :0.993179   Max.   :1.000000
```

4. Predicting diabetes using H2O+R

Bringing it all together (continued …)

Next, we run the Gradient Boosting Machine:

```
## *************** GBM **************************************
gbm <- h2o.gbm(x = predictors,
             y = response,
             training_frame     = train,
             validation_frame   = test,
             ntrees             = 1000,
             max_depth          = 6,
             learn_rate         = 0.1,
             stopping_rounds    = 1,
             stopping_tolerance = 0.01,
             stopping_metric    = "misclassification",
             balance_classes    = T,
             seed               = 2000000)
gbm
gbm_pred_hex <- h2o.predict(gbm, newdata=hold)
perf_gbm <- h2o.performance(gbm, nhanes_DF_imp_sup_hex)
h2o.confusionMatrix(perf_gbm)

gbm_pred_hex2 <- h2o.predict(gbm, newdata=nhanes_DF_imp_sup_hex)
nhanes_DF_imp_sup_hex$GBM_predict <- gbm_pred_hex2$predict
nhanes_DF_imp_sup_hex$GBM_Yes <- gbm_pred_hex2$Yes
nhanes_DF_imp_sup_hex$GBM_No <- gbm_pred_hex2$No
summary(nhanes_DF_imp_sup_hex)
```

Then, we run the Generalized Linear Model:

```
glm <- h2o.glm(x = predictors,
             y = response,
             training_frame    = train,
             validation_frame  = test,
             nfolds            = 5,
             family            = "binomial")
glm
glm_pred_hex <- h2o.predict(glm, newdata=hold)
perf_glm <- h2o.performance(glm, nhanes_DF_imp_sup_hex)
h2o.confusionMatrix(perf_glm)

glm_pred_hex2 <- h2o.predict(glm, newdata=nhanes_DF_imp_sup_hex)
nhanes_DF_imp_sup_hex$GLM_predict <- glm_pred_hex2$predict
nhanes_DF_imp_sup_hex$GLM_Yes <- glm_pred_hex2$Yes
nhanes_DF_imp_sup_hex$GLM_No <- glm_pred_hex2$No
summary(nhanes_DF_imp_sup_hex)
```

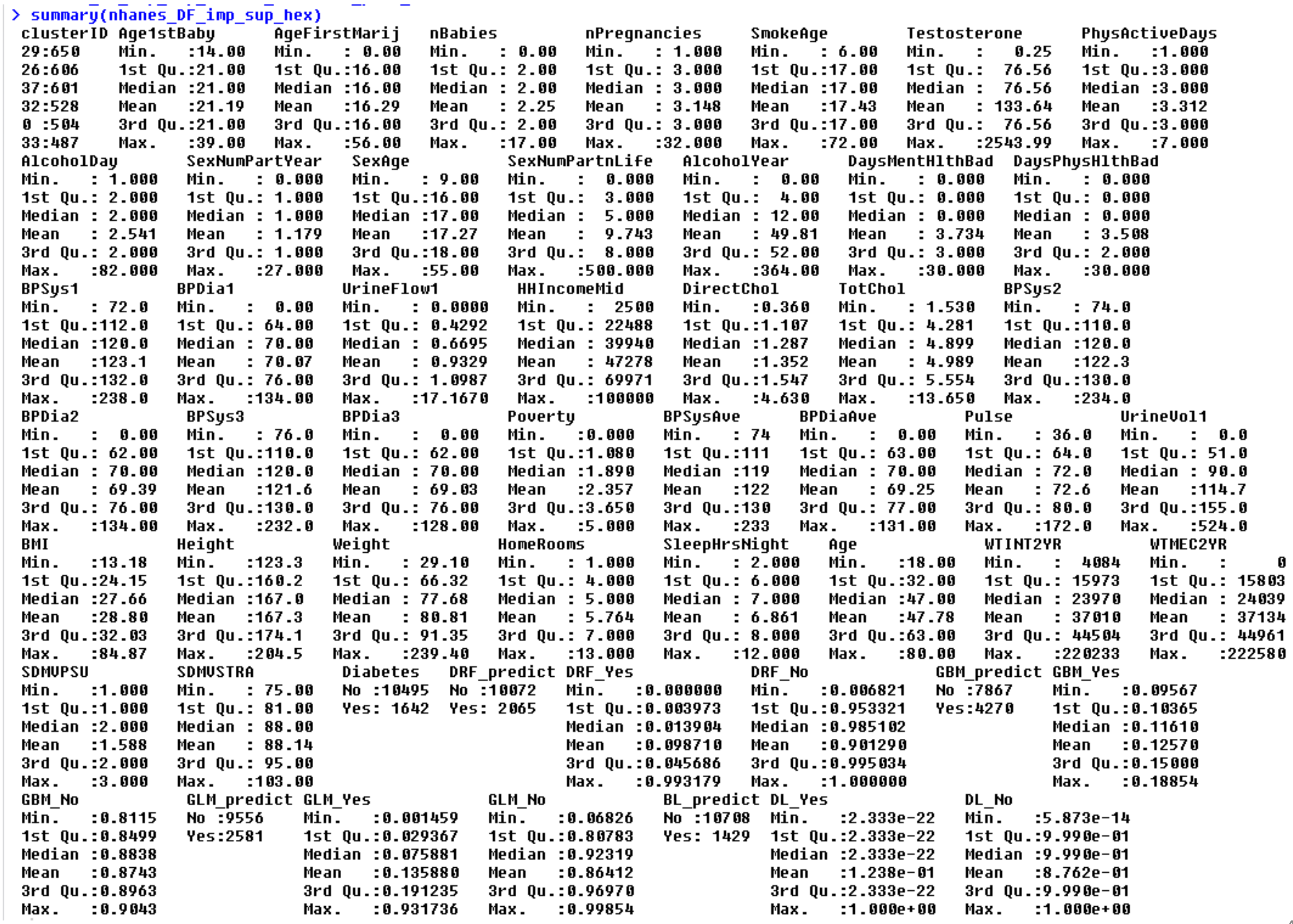
4. Predicting diabetes using H2O+R

Bringing it all together (continued …)

Finally, we run the Deep Learning model:

```
## *************** DL ****************************************
dl <- h2o.deeplearning(x              = predictors,
                       y              = response,
                       training_frame = train,
                       activation     = "Tanh",
                       balance_classes = TRUE,
                       hidden         = c(100, 100, 100, 100, 100),
                       epochs         = 100)
dl
dl_pred_hex <- h2o.predict(dl, newdata=hold)
perf_dl <- h2o.performance(dl, nhanes_DF_imp_sup_hex)
h2o.confusionMatrix(perf_dl)

dl_pred_hex2 <- h2o.predict(dl, newdata=nhanes_DF_imp_sup_hex)
nhanes_DF_imp_sup_hex$BL_predict <- dl_pred_hex2$predict
nhanes_DF_imp_sup_hex$DL_Yes <- dl_pred_hex2$Yes
nhanes_DF_imp_sup_hex$DL_No <- dl_pred_hex2$No
summary(nhanes_DF_imp_sup_hex)
```

We have run all four models, and stored the predictive outputs back within the original imputed data set.

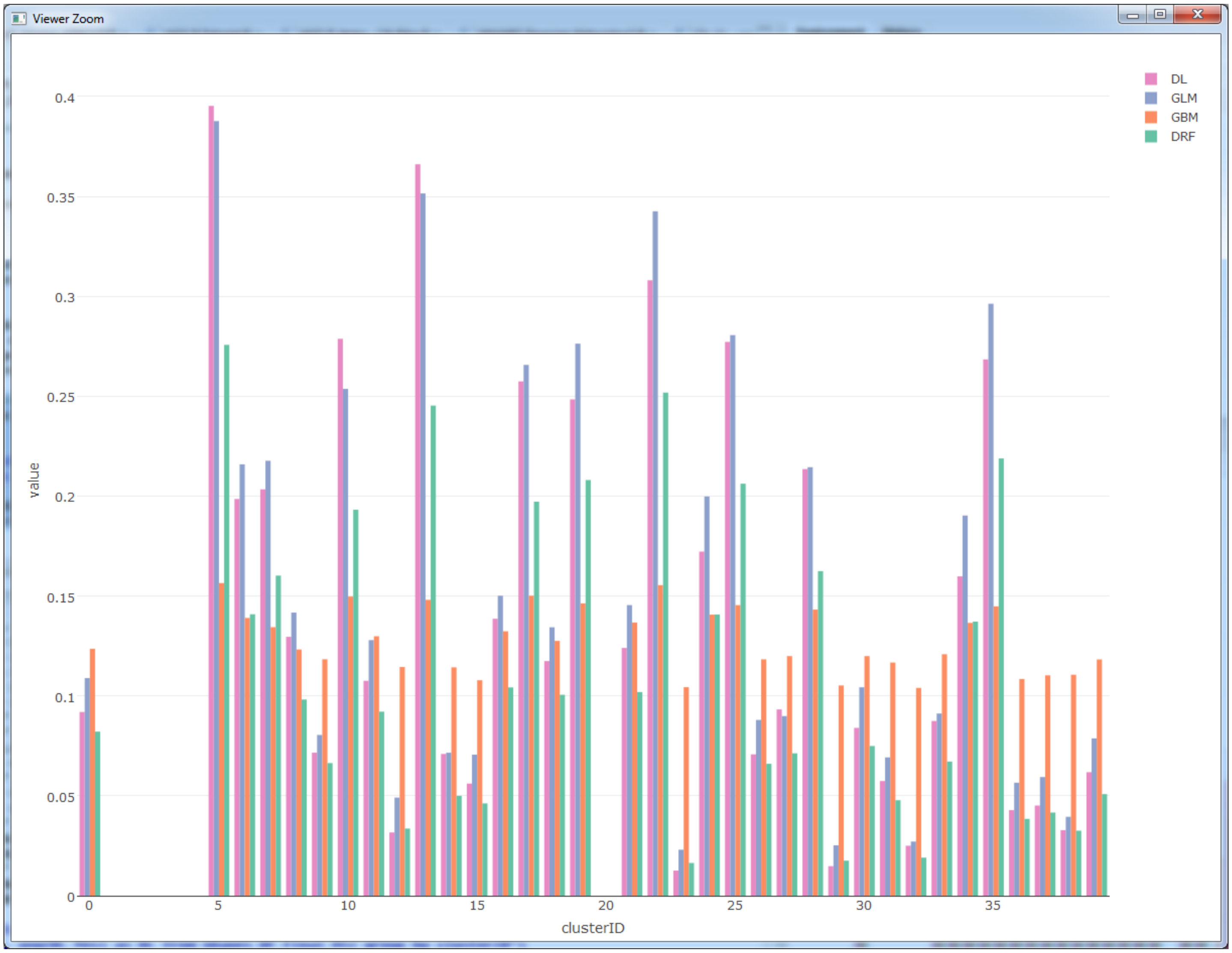


```
> summary(nhanes_DF_imp_sup_hex)
 clusterID Age1stBaby      AgeFirstMarij   nBabies         nPregnancies     SmokeAge        Testosterone     PhysActiveDays
 29:650    Min.   :14.00   Min.   : 0.00   Min.   : 0.00   Min.   : 1.000   Min.   : 6.00   Min.   :   0.25   Min.   :1.000
 26:606    1st Qu.:21.00   1st Qu.:16.00   1st Qu.: 2.00   1st Qu.: 3.000   1st Qu.:17.00   1st Qu.:  76.56   1st Qu.:3.000
 37:601    Median :21.00   Median :16.00   Median : 2.00   Median : 3.000   Median :17.00   Median :  76.56   Median :3.000
 32:528    Mean   :21.19   Mean   :16.29   Mean   : 2.25   Mean   : 3.148   Mean   :17.43   Mean   : 133.64   Mean   :3.312
 0 :504    3rd Qu.:21.00   3rd Qu.:16.00   3rd Qu.: 2.00   3rd Qu.: 3.000   3rd Qu.:17.00   3rd Qu.:  76.56   3rd Qu.:3.000
 33:487    Max.   :39.00   Max.   :56.00   Max.   :17.00   Max.   :32.000   Max.   :72.00   Max.   :2543.99   Max.   :7.000
 AlcoholDay       SexNumPartYear    SexAge          SexNumPartnLife    AlcoholYear      DaysMentHlthBad  DaysPhysHlthBad
 Min.   : 1.000   Min.   : 0.000   Min.   : 9.00   Min.   :  0.000   Min.   :  0.00   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.:16.00   1st Qu.:  3.000   1st Qu.:  4.00   1st Qu.: 0.000   1st Qu.: 0.000
 Median : 2.000   Median : 1.000   Median :17.00   Median :  5.000   Median : 12.00   Median : 0.000   Median : 0.000
 Mean   : 2.541   Mean   : 1.179   Mean   :17.27   Mean   :  9.743   Mean   : 49.81   Mean   : 3.734   Mean   : 3.508
 3rd Qu.: 2.000   3rd Qu.: 1.000   3rd Qu.:18.00   3rd Qu.:  8.000   3rd Qu.: 52.00   3rd Qu.: 3.000   3rd Qu.: 2.000
 Max.   :82.000   Max.   :27.000   Max.   :55.00   Max.   :500.000   Max.   :364.00   Max.   :30.000   Max.   :30.000
 BPSys1          BPDia1           UrineFlow1        HHIncomeMid      DirectChol      TotChol          BPSys2
 Min.   : 72.0   Min.   :  0.00   Min.   : 0.0000   Min.   :  2500   Min.   :0.360   Min.   : 1.530   Min.   : 74.0
 1st Qu.:112.0   1st Qu.: 64.00   1st Qu.: 0.4292   1st Qu.: 22488   1st Qu.:1.107   1st Qu.: 4.281   1st Qu.:110.0
 Median :120.0   Median : 70.00   Median : 0.6695   Median : 39940   Median :1.287   Median : 4.899   Median :120.0
 Mean   :123.1   Mean   : 70.07   Mean   : 0.9329   Mean   : 47278   Mean   :1.352   Mean   : 4.989   Mean   :122.3
 3rd Qu.:132.0   3rd Qu.: 76.00   3rd Qu.: 1.0987   3rd Qu.: 69971   3rd Qu.:1.547   3rd Qu.: 5.554   3rd Qu.:130.0
 Max.   :238.0   Max.   :134.00   Max.   :17.1670   Max.   :100000   Max.   :4.630   Max.   :13.650   Max.   :234.0
 BPDia2           BPSys3          BPDia3           Poverty         BPSysAve      BPDiaAve         Pulse           UrineVol1
 Min.   :  0.00   Min.   : 76.0   Min.   :  0.00   Min.   :0.000   Min.   : 74   Min.   :  0.00   Min.   : 36.0   Min.   :  0.0
 1st Qu.: 62.00   1st Qu.:110.0   1st Qu.: 62.00   1st Qu.:1.080   1st Qu.:111   1st Qu.: 63.00   1st Qu.: 64.0   1st Qu.: 51.0
 Median : 70.00   Median :120.0   Median : 70.00   Median :1.890   Median :119   Median : 70.00   Median : 72.0   Median : 90.0
 Mean   : 69.39   Mean   :121.6   Mean   : 69.03   Mean   :2.357   Mean   :122   Mean   : 69.25   Mean   : 72.6   Mean   :114.7
 3rd Qu.: 76.00   3rd Qu.:130.0   3rd Qu.: 76.00   3rd Qu.:3.650   3rd Qu.:130   3rd Qu.: 77.00   3rd Qu.: 80.0   3rd Qu.:155.0
 Max.   :134.00   Max.   :232.0   Max.   :128.00   Max.   :5.000   Max.   :233   Max.   :131.00   Max.   :172.0   Max.   :524.0
 BMI             Height          Weight           HomeRooms        SleepHrsNight    Age             WTINT2YR         WTMEC2YR
 Min.   :13.18   Min.   :123.3   Min.   : 29.10   Min.   : 1.000   Min.   : 2.000   Min.   :18.00   Min.   :  4084   Min.   :     0
 1st Qu.:24.15   1st Qu.:160.2   1st Qu.: 66.32   1st Qu.: 4.000   1st Qu.: 6.000   1st Qu.:32.00   1st Qu.: 15973   1st Qu.: 15803
 Median :27.66   Median :167.0   Median : 77.68   Median : 5.000   Median : 7.000   Median :47.00   Median : 23970   Median : 24039
 Mean   :28.80   Mean   :167.3   Mean   : 80.81   Mean   : 5.764   Mean   : 6.861   Mean   :47.78   Mean   : 37010   Mean   : 37134
 3rd Qu.:32.03   3rd Qu.:174.1   3rd Qu.: 91.35   3rd Qu.: 7.000   3rd Qu.: 8.000   3rd Qu.:63.00   3rd Qu.: 44504   3rd Qu.: 44961
 Max.   :84.87   Max.   :204.5   Max.   :239.40   Max.   :13.000   Max.   :12.000   Max.   :80.00   Max.   :220233   Max.   :222580
 SDMVPSU         SDMVSTRA         Diabetes    DRF_predict DRF_Yes              DRF_No             GBM_predict GBM_Yes
 Min.   :1.000   Min.   : 75.00   No :10495   No :10072   Min.   :0.000000   Min.   :0.006821   No :7867    Min.   :0.09567
 1st Qu.:1.000   1st Qu.: 81.00   Yes: 1642   Yes: 2065   1st Qu.:0.003973   1st Qu.:0.953321   Yes:4270    1st Qu.:0.10365
 Median :2.000   Median : 88.00                           Median :0.013904   Median :0.985102               Median :0.11610
 Mean   :1.588   Mean   : 88.14                           Mean   :0.098710   Mean   :0.901290               Mean   :0.12570
 3rd Qu.:2.000   3rd Qu.: 95.00                           3rd Qu.:0.045686   3rd Qu.:0.995034               3rd Qu.:0.15000
 Max.   :3.000   Max.   :103.00                           Max.   :0.993179   Max.   :1.000000               Max.   :0.18854
 GBM_No           GLM_predict GLM_Yes             GLM_No            BL_predict DL_Yes               DL_No
 Min.   :0.8115   No :9556    Min.   :0.001459   Min.   :0.06826   No :10708   Min.   :2.333e-22   Min.   :5.873e-14
 1st Qu.:0.8499   Yes:2581    1st Qu.:0.029367   1st Qu.:0.80783   Yes: 1429   1st Qu.:2.333e-22   1st Qu.:9.990e-01
 Median :0.8838               Median :0.075881   Median :0.92319               Median :2.333e-22   Median :9.990e-01
 Mean   :0.8743               Mean   :0.135880   Mean   :0.86412               Mean   :1.238e-01   Mean   :8.762e-01
 3rd Qu.:0.8963               3rd Qu.:0.191235   3rd Qu.:0.96970               3rd Qu.:2.333e-22   3rd Qu.:9.990e-01
 Max.   :0.9043               Max.   :0.931736   Max.   :0.99854               Max.   :1.000e+00   Max.   :1.000e+00
```

Bringing it all together (continued …)

The final thing to do is to visualize how much each cluster is at risk of contracting diabetes.

```
## *************** Viz Clusters ****************************
nhanes_DF_Final_Viz <- as.data.frame(nhanes_DF_imp_sup_hex)
str(nhanes_DF_Final_Viz)

agg_nhanes_Viz <- sqldf("select clusterID, avg(DRF_Yes) as DRF,
avg(GBM_Yes) as GBM, avg(GLM_Yes) as GLM, avg(DL_Yes) as DL from nhanes_DF_Final_Viz group by clusterID")
library(reshape)
agg_nhanes_Viz2 <- melt(agg_nhanes_Viz, id="clusterID")
plot_ly(agg_nhanes_Viz2, x = clusterID, y = value, type = "bar", color = variable).
```



We can see from the graph that clearly cluster 5 is the most at risk indicated by all four models. The risk percentages range from ~15% to 40%. Not surprising we see these ranges so wide as our predicted error rate on each model was ~20-30% depending on the model.

5. Summary / Conclusion

What did we learn/do?

We did several things in this walk through of H2O.ai and R capabilities dealing with the analysis of diabetes. Let's recap each step and what we discovered along the way:

1) Through the EDA phase we learned that number of pregnancies, number of babies, age and weight seem to be areas where large separations in central tendency and variation exists between those with diabetes and those without. Also, the use of plotly for our visualizations allowed us to uniquely drill down and filter within the EDA and other areas as well.
2) We created 40 clusters groups to partition our data set within identifiable homogeneous groups and overlaid the proportion of diabetes found within each.
3) Next we four ran individual models on each cluster group (filtered by at least 100) to determine the main drivers that help predicting diabetes within each group.
4) Finally, we again ran all four models on the entire imputed data set appending the predictive outputs back within the data set to determine overall cluster risk of diabetes.

This approach is useful in that it helps researches develop and validate assumptions about risk of diabetes and how those risk vary by certain groups of individuals. Some of the analysis showed that the number of pregnancies and number of babies as key factors that contribute heavily to diabetes. This might help researches identify other features as well as groups for gestational diabetes.

Review of H2O capabilities

The H2O.ai platform allowed use to iterate through several computational intense process within R such as data imputation, finding the right k size for clusters, converging through thousands of trees for modeling, etc.

This capabilities are value add into the data scientist tool kit and helps with speed of processing.

Future enhancements & recommendations

This work was fun to put together and very insightful for me as well, but with any documentation on approaches in analytics there are flaws and this document is no exception. I've laid out below how this work can be enhanced below:

1) The data imputation methods are very simple within H2O.ai and as a result you get simple basic answers. Our imputation method using H2O.ai could be enhanced by using "by" clauses within our function of features that have more (or fully) presence within the data set. By using features that indicated no presence of NA's such as "Age", "Gender", etc. could be used to help impute missing values. H2O.ai imputation methods can also be enhanced by taking more capabilities on like those found within the R "mice" package (Multivariate Imputation by Chained Equations).
2) Although the grid functions for parameter setting was run for the Gradient Boosting Machine sections, it was not included in this work.
3) The overall error rates of these models are too high for diabetes = "Yes" and therefore no causal conclusion or actions should be taken fully as a result of the models predictive outputs. The H2O.ai models could be enhanced by allowing for weighting the loss matrix. To some degree it does with the balancing parameter, however if you wanted to penalize the error rate on "Yes" more than "No" there is no place to do that as far as I have found.

5. Summary / Conclusion

Future enhancements & recommendations

This may go without saying, but just in case: the results of this analysis demonstrated in this document does not in any way suggestion anyone to self diagnose or take any medical or any other actions as a result of the results produced in this document. Please see a certified and licensed physician for any medical related concerns or questions you might have.