2019

Chimera: A Semiotic Grid

A BINARY LOGISTIC REGRESSION STUDY

JOSHUA LAFERRIERE

SHUBHAM BHOWATE

PRERANAREDDY ANANTH

Contents

[Executive Summary 3](#_Toc8016512)

[Introduction 4](#_Toc8016516)

[Project page: 4](#_Toc8016518)

[Data 4](#_Toc8016519)

[The steps required to run the code are as follows: 5](#_Toc8016520)

[System Files 5](#_Toc8016521)

[Partitioning & Sampling Files 5](#_Toc8016522)

[Libraries 6](#_Toc8016523)

[Error logging 6](#_Toc8016524)

[Data Loading (Extract) 6](#_Toc8016525)

[Data Cleaning 7](#_Toc8016526)

[Conversion Profiles 8](#_Toc8016527)

[Convert1Index 8](#_Toc8016528)

[Rules 9](#_Toc8016529)

[Convert2Index 9](#_Toc8016530)

[Rules 9](#_Toc8016531)

[Convert3Index 10](#_Toc8016532)

[Rules 10](#_Toc8016533)

[Post Conversion 10](#_Toc8016534)

[Hypothesis 11](#_Toc8016535)

[Oversampling 11](#_Toc8016536)

[Correlation Matrix (Factor Grouping) 13](#_Toc8016537)

[Model Assumptions 13](#_Toc8016538)

[Regression using StepAIC (early method) 14](#_Toc8016539)

[sub\_returnCVnames (Modified BestGLM) 15](#_Toc8016540)

[Factor Reduction (cleandatacode.R) 16](#_Toc8016541)

[Data Partitioning: Loops & Resampling 17](#_Toc8016542)

[SaveCSVs.R 18](#_Toc8016543)

[Model Comparison with Population fitted model 18](#_Toc8016544)

[Conclusion 19](#_Toc8016545)

[High Level Inferences 19](#_Toc8016546)

[Further considerations 19](#_Toc8016547)

[Logistic Regression Equations 21](#_Toc8016548)

[References 25](#_Toc8016549)

[Appendix 25](#_Toc8016550)

[V7221 Median Conversion 29](#_Toc8016551)

[Median Confidence 30](#_Toc8016552)

[Oversampling Code 31](#_Toc8016553)

[Modified BestGLM Code 32](#_Toc8016554)

[bestGLM errors 33](#_Toc8016555)

[Nullpairs 34](#_Toc8016556)

# 

# Executive Summary

Our research consists of poll data from the ICPSR: Inter-University Consortium for Political and Social Research. The vision of the project is to identify response factors that can be influenced to achieve a better quality of life for students.

We will be utilizing binary class differentiation around a median (>=), stratified subsampling techniques of bootstrapping a Monte Carlo resampling and binary logistic regression (see figure 1) to find factors of interest from poll research data to find factors of influence for the following 3 response terms.

# V7118 #X PSYD/LIFETIME

* + Lifetime presence of Psychedelic substance use

# V8517 FRQ GANG FIGHT

* + Lifetime presence of gang fight

# V7221 R HS GRADE/D=1

* + GPA above or equal to B+ (Median response)

The three questions we identified are:

* What factors influence being included in “above or equal” to median value GPA? (B+)
* What factors influence having the presence of a history of Gang Violence?
* What factors influence lifetime presence of psychedelic drug use?

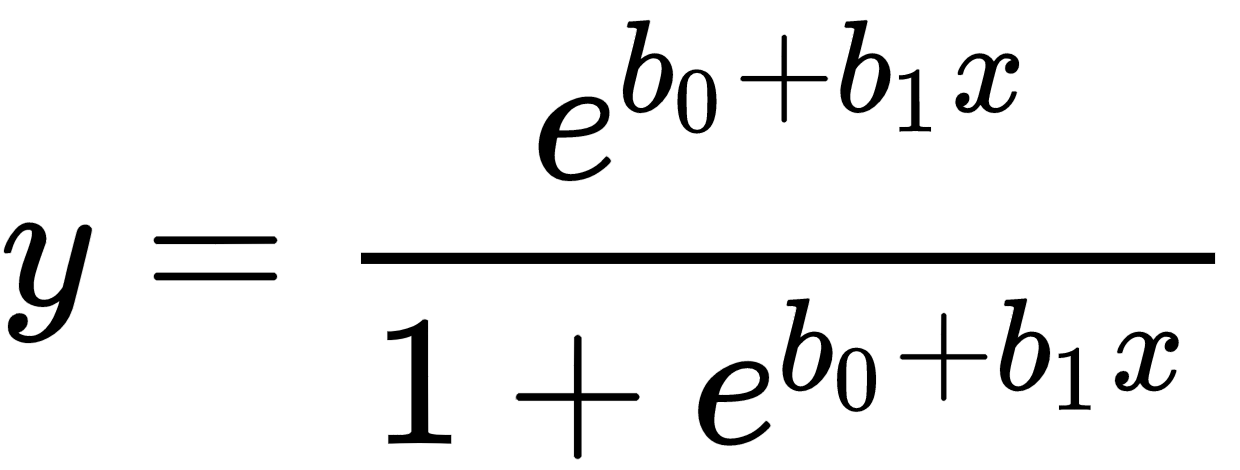


Figure 1

The following set of 21 factors serve as a template and are shared across each response term

V7202,0,"R'S SEX",2

V7206,0,"R'S HSHLD FATHER",2

V7215,2,"FATHR EDUC LEVEL",2

V7551,2,"#HR/W INTERNET S",3

V7552,2,"DALY WEB FACEBK",3

V7553,2,"#HR GAMING",3

V7562,2,"#HR TEXT",7

V7563,2,"#HR TALK CELL",7

V8526,3,"OFTN EAT BRKFST",4

V8527,3,"OFTN EAT GN VEG",4

V8528,3,"OFTN EAT FRUIT",4

V8529,3,"OFTN EXERCISE",4

V8530,3,"OFTN 7HRS SLEEP",4

V8531,3,"OFTN SLEEP <SHLD",4

V8502,3,"LIFE MEANINGLESS",5

V8505,3,"I ENJOY LIFE",5

V8509,3,"FUTURE HOPELESS",5

V8512,3,"SATISFD W MYSELF",5

V8514,3,"GOOD TO BE ALIVE",6

V8536,3,"FUTR R LIFE WRSE",6

V8565,3,"I AM OFTEN BORED",6

To achieve model robustness in our final data frame used for modeling we utilize stratified subsampling, a Monte Carlo resampling technique, to balance the response term to achieve a mean close to .5; otherwise known as oversampling. We exploit measures of central tendency to extract the means from these balanced datasets to derive predictor effect plots (Pardoe 141, 258, 266). These effect plots show what the presence of a class term does in terms of propensity to be in the response class holding other class terms constant at their respective means (in our case stratified means).

# Introduction

The data provided covers the years 2012 to 2017. The respondents in the poll study consisted of 8th to 10th graders. The name of the study is titled “**Monitoring the Future: A Continuing Study of American Youth (8th- and 10th-Grade Surveys),”.**

The poll data provided gave 6 years with 3 files per year. These 3 files were an SPSS .sav file (which we exported to .csv for subsequent merging in R), a codebook (which provided distribution, categorical breakdown, as well as descriptions per variable), and a user guide pdf which we didn’t really utilize but provided breakdowns of how factors changed between years.

The factors were already divided into the following categories

1. Drugs

B. Education.

C. Work and Leisure.

D. Gender/Sex Roles and Family.

E. Population Concerns.

F. Conservation, Materialism, Equity, etc.

G. Religion.

I. Social Change.

J. Social Problems.

L. Military.

M. Interpersonal Relationships.

O. Concern for Others.

P. Happiness.

Q. Other Personality Variables.

R. Background and School.

S. Deviant Behavior and Victimization.

T. Health habits and symptoms.

# Our strategy was to dive into this large dataset (i.e. the various columns provided in each provided year’s codebook) collectively and identify potential would be factors and pool them for further regression analysis. We focussed on habitual, environmental, health, and psychological related factors for our independent terms.

# Project page:

<https://github.com/thistleknot/Capstone-577>

For version control as well as accessibility, the project was tracked using GitHub. The project is currently over 500 commits. The readme.

<https://github.com/thistleknot/Capstone-577/blob/master/readme.txt>

The files we based our final analysis were based on commit **256c42b**.

## Data

* Console Output Report:
  + <https://github.com/thistleknot/Capstone-577/tree/master/output>/[output.txt](https://github.com/thistleknot/Capstone-577/blob/256c42b1fc9b63fe2579d8446400e6945727e713/output/Milestone%2018%20report%20output.txt)
  + To match the report, we had to stop the results once during saveCSVs.R, and rerun, then the results showed the same.
* Commit Output: <https://github.com/thistleknot/Capstone-577/tree/master/output>
* Predictor Effect Plots
  + <https://docs.google.com/spreadsheets/d/1AyRXf5kk0MUCzyFW2AwmZPd__bEsh5VhBwo9uBq92C8/>

## The steps required to run the code are as follows:

* In cleandatacode.R modify
  + Set Linux to 0 or 1
  + sourceDir to point to where you downloaded and extracted the source folder to. This should be the same folder that output sits in.
* Run cleandatacode.R which starts the simulation up until the point where it starts to output tabulation results.
  + Then stop the simulation
    - You can let it run all the way through, but the files necessary to run the rest of the simulation (\*final.csv) are already in the output folder.
* Run saveCSVs.R and allow it to finish
  + You may be prompted to install necessary libraries. If so, please use tools-install packages to do so.

## System Files

The system consists of 2 different main files.

* Cleandatacode.R
  + Derives final.csv which stores cross-validated thresholds,
  + does factor reduction.
* saveCSVs.R
  + Best subset
  + Class balancing
  + ROC
    - Population model to population data
    - Monte Carlo model to population data
    - Monte Carlo model to Monte Carlo data
  + Check VIF for collinearity
  + Confusion Matrix
  + Correlation Plots

## Partitioning & Sampling Files

Other files related to cleaning, transforming, indexing, class balancing, and aggregating the data into data frames (and in their proper order of being called) are.

* newDF.R
* reseedboth.R
* reseedTest.R
* reseedTrain.R
* MCResampleTest.R
* MCResampleTrain.R
* redrawTrain.R
* redrawTest.R
* pairedLists.R

Note: All these files are assumed to be sourced from cleandatacode.R

# Libraries

* dplyr
* plyr
* anchors
* caret
* corrplot
* MASS
* car
* leaps
* bestglm
* compare
* R.utils
* tidyr
* stringr
* ModelMetrics
* "ROCR"
* outliers
* factoextra
* Rfast
* cutpointr
* InformationValue
* tibble
* mctest
* rcompanion
* MLmetrics
* anchors

# Error logging

We ran into two errors during our simulation. One was logical which was caused by an empty data frame resulting from no records after dropping NAs which we logged as “nullpairs” and the other was an error thrown by bestGLM “Error in qk[, 4]: an incorrect number of dimensions” which we not so eloquently logged as separate empty timestamped csvs. We suspect this latter error is caused by the size of the data frame. We logged the error for manual review to see if the occurrence of the errors would have affected the final tabulated results when applying a cutoff, we determined they did not.

# Data Loading (Extract)

We received a fold for each of the 6 years, each with SPSS .sav file, a PDF codebook, and a PDF user guide (which detailed the variable coding and distribution breakdown) within. We exported the SPSS .save files to CSV and reintegrated them in R into a single data frame called **d\_combined** (which later is transformed into **NewDF**).

* Dataset for 2012 - 31106 records across 552 Variables.

#sourceDir="/home/rstudio/577/Capstone-577/"

sourceDir="C:/Users/user/Documents/School/CSUF/ISDS577/projects/Capstone-577/"

source(paste0(sourceDir,"bestglm.R"))

source(paste0(sourceDir,"pairedLists.R"))

# Read CSV into R

d\_2012 <- read.csv(paste0(sourceDir,"34574-0001-Data.csv"), header=TRUE, sep=",")

d\_2013 <- read.csv(paste0(sourceDir,"34574-0001-Data.csv"), header=TRUE, sep=",")

d\_2014 <- read.csv(paste0(sourceDir,"36149-0001-Data.csv"), header=TRUE, sep=",")

d\_2015 <- read.csv(paste0(sourceDir,"36407-0001-Data.csv"), header=TRUE, sep=",")

d\_2016 <- read.csv(paste0(sourceDir,"36799-0001-Data.csv"), header=TRUE, sep=",")

d\_2017 <- read.csv(paste0(sourceDir,"37183-0001-Data.csv"), header=TRUE, sep=",")

d\_combined <- rbind.fill(d\_2012,d\_2013,d\_2014,d\_2015,d\_2016,d\_2017)

* Dataset for 2013 - 28495 records across 550 variables.
* Dataset for 2014 - 28536 records across 563 variables
* Dataset for 2015 - 31162 records across 577 variables
* Dataset for 2016 - 32873 records across 571 variables
* Dataset for 2017 - 30181 records across 581 variables

# Data Cleaning

During data exploration phase. We discovered a lot of NA’s coded as -8 and -9’s.

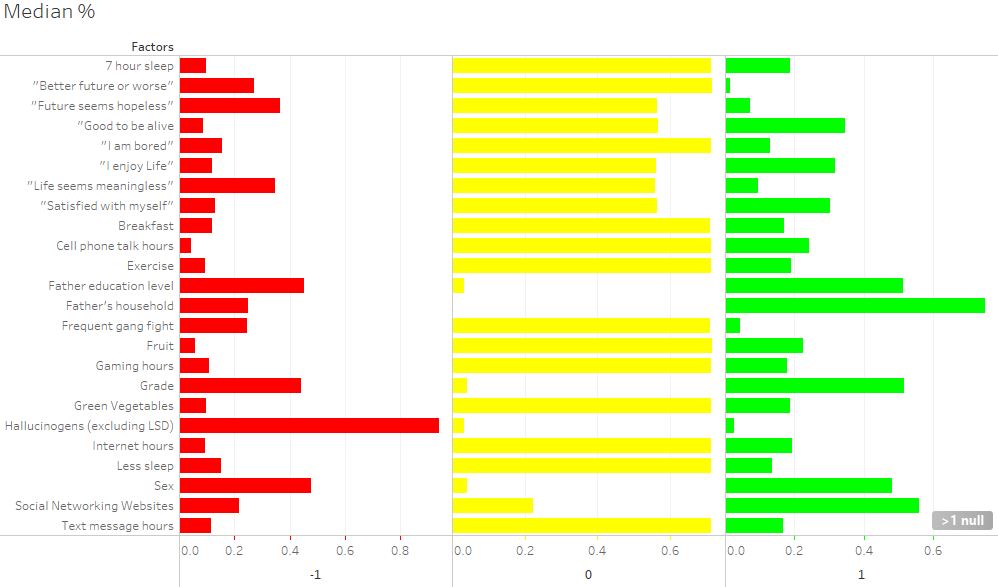
There are 619 factors left after combining the datasets.

We have 3 files we use which prep the system for which factors to focus on. These files are gpalist.txt, psyDFilterList.txt, and gangfight.txt. Sample input line from gangfight.txt

V8517,1,"FRQ GANG FIGHT",0

The columns represented in these input files represent

* factor id
* conversion profile
* description
* category flag (no longer used aside from excluding 8’s as non-response terms and 0 as response and all else as x’s)



To address the issue with NA’s we took two approaches. Rather than applying it to the entire data frame d\_combined (which results in no records!) or excluding columns with excessive NAs. Our approach was to deal with NA’s at the last possible moment at the smallest possible level which was when we aggregated the columns together. After generating pairedLists.R we modified the algorithm to work with only 2 columns at a time. This was the juncture at when the records are dropped. Which required us to rewrite the entire indexing structure of the data frames. In our case where we resample from NewDF “reseed, MCResample, and reDraw” R files within cleandatacode.R.

# Conversion Profiles

The transformation of the data occurs in NewDF.R where we transform according to 3 conversion profiles.

* convert1Index
  + Simple yes/no
* convert2Index
  + Median based
  + One deciding constant set at loop iteration
    - medianDirection
    - controls operant condition: >= median or > median
* convert3Index
  + split around a positive response (i.e. response was classed as 4+: “mostly” or higher).

It should be noted that these profiles convert negative responses in NewDF to -1 for what will become 0 eventually in data.train and data.test at the time they are called. This is because we use 0 at this juncture to hold NAs and then drop NAs at the last possible level (after generating index’s for which columns we are going to include in a partition which is what pairedLists.R’s output is for, column indexes), this is because it was easier to work with data frames with 0’s rather than NA’.s

Within NewDF.R the basic logic for the conversion is as follows

Pre conversion

Anything below 0 is considered NA which we code as 0.

NewDF <- replace.value( NewDF, colnames(NewDF), from=as.integer(-9), to=as.double(0), verbose = FALSE)

NewDF <- replace.value( NewDF, colnames(NewDF), from=as.integer(-8), to=as.double(0), verbose = FALSE)

Gender Conversion

NewDF <- replace.value( NewDF, "V7202", from=as.integer(1), to=as.double(-1), verbose = FALSE)

#father household status

NewDF <- replace.value( NewDF, "V7206", from=as.integer(0), to=as.double(-1), verbose = FALSE)

NewDF <- replace.value( NewDF, "V7202", from=as.integer(2), to=as.double(1), verbose = FALSE)

## Convert1Index

Simple binary presence (yes/no) (i.e. any amount is converted to a yes, or 1)

V1 V2 V3 V4

1 V7118 1 #X PSYD/LIFETIME 0

2 V8517 1 FRQ GANG FIGHT 8

### Rules

NewDF <- replace.value( NewDF, as.character(list[,1][convert1Index]), from=as.integer(1), to=as.double(-1), verbose = FALSE)

NewDF <- replace.value( NewDF, as.character(list[,1][convert1Index]), from=as.integer(2), to=as.double(1), verbose = FALSE)

NewDF <- replace.value( NewDF, as.character(list[,1][convert1Index]), from=as.integer(3), to=as.double(1), verbose = FALSE)

NewDF <- replace.value( NewDF, as.character(list[,1][convert1Index]), from=as.integer(4), to=as.double(1), verbose = FALSE)

NewDF <- replace.value( NewDF, as.character(list[,1][convert1Index]), from=as.integer(5), to=as.double(1), verbose = FALSE)

NewDF <- replace.value( NewDF, as.character(list[,1][convert1Index]), from=as.integer(6), to=as.double(1), verbose = FALSE)

NewDF <- replace.value( NewDF, as.character(list[,1][convert1Index]), from=as.integer(7), to=as.double(1), verbose = FALSE)

## Convert2Index

V1 V2 V3 V4

3 V7221 2 R HS GRADE/D=1 8

6 V7215 2 FATHR EDUC LEVEL 2

7 V7551 2 #HR/W INTERNET S 3

8 V7552 2 DALY WEB FACEBK 3

9 V7553 2 #HR GAMING 3

10 V7562 2 #HR TEXT 7

11 V7563 2 #HR TALK CELL 7

Converted to 1 if >= Median.

### Rules

Our concern/hope was that there was no increasing or decreasing trend so we could state that the binary logistic regression equation could be inferred to apply to years beyond the dataset, but to be able to make that statement, we needed to check each year and derive confidence intervals for the median. See the **Appendix** for **Median Confidence** for the code. We utilized the median, which is a common bifurcator in classification, as the measure of central tendency to split by. Also in the Appendix we have an example of how we transform V7221 around the median under V7221 Median Conversion. We utilized an algorithm provided by University College London.

The medians were confirmed the same for each year. For V7563 in 2012, the median’s 95% upper confidence interval was 3 vs 2. But for our purposes, we assume the median is accurate for each year and remains as such and a population resamples every 5 years would be in order to confirm the median remained the same. This means when we derive our binary logistic regression equations. These medians represent thresholds of inclusiveness to be considered in the 1 category. Often these are verifiable objective points to be aimed for when seeking to modify an expected outcome.

* V7221,2,"R HS GRADE/D=1",0
  + Median: #7: B+ 95% conf interval confirmed
* V7215,2,"FATHR EDUC LEVEL",0
  + #5: for college grad father, 95% conf confirmed
* V7551,2,"#HR/W INTERNET S",0
  + #4: 3-5 Hours Internet #95% conf confirmed
* V7552,2,"DALY WEB FACEBK",0
  + #5: 6-9 Hours Facebook # 95% conf confirmed
* V7553,2,"#HR GAMING",0
  + #4 3-5 Hours Gaming # 95% conf confirmed
* V7562,2,"#HR TEXT",0
  + #4 3-5 Hours Texting # 95% conf confirmed
* V7563,2,"#HR TALK CELL",0
  + #2: <1 Hour talking on cell phone # 95% conf confirmed

### Convert3Index

**

V1 V2 V3 V4

12 V8526 3 OFTN EAT BRKFST 4

13 V8527 3 OFTN EAT GN VEG 4

14 V8528 3 OFTN EAT FRUIT 4

15 V8529 3 OFTN EXERCISE 4

16 V8530 3 OFTN 7HRS SLEEP 4

17 V8531 3 OFTN SLEEP <SHLD 4

18 V8502 3 LIFE MEANINGLESS 5

19 V8505 3 I ENJOY LIFE 5

20 V8509 3 FUTURE HOPELESS 5

21 V8512 3 SATISFD W MYSELF 5

22 V8514 3 GOOD TO BE ALIVE 6

23 V8536 3 FUTR R LIFE WRSE 6

24 V8565 3 I AM OFTEN BORED 6

### Rules

### Post Conversion

NewDF <- replace.value( NewDF, as.character(list[,1][convert3Index]), from=as.integer(1), to=as.double(-1), verbose = FALSE)

NewDF <- replace.value( NewDF, as.character(list[,1][convert3Index]), from=as.integer(2), to=as.double(-1), verbose = FALSE)

NewDF <- replace.value( NewDF, as.character(list[,1][convert3Index]), from=as.integer(3), to=as.double(-1), verbose = FALSE)

NewDF <- replace.value( NewDF, as.character(list[,1][convert3Index]), from=as.integer(4), to=as.double(1), verbose = FALSE)

NewDF <- replace.value( NewDF, as.character(list[,1][convert3Index]), from=as.integer(5), to=as.double(1), verbose = FALSE)

NewDF <- replace.value( NewDF, as.character(list[,1][convert3Index]), from=as.integer(6), to=as.double(1), verbose = FALSE)

0’s are stored as -8 (NA’s) and then a final pass is done to convert trailing -8 and -9’s to 0’s. This is likely redundant having been done at the beginning

NewDF[NewDF == 0] <- -8

NewDF <- replace.value( NewDF, colnames(NewDF), from=as.integer(-9), to=as.double(0), verbose = FALSE)

NewDF <- replace.value( NewDF, colnames(NewDF), from=as.integer(-8), to=as.double(0), verbose = FALSE)

# Hypothesis

We will establish that the factors chosen have a non zero summed coefficient relationship in a binary logistic regression equation with the response term. We will do this using scores of significance as outputted by R. We will also compare our models against an overfitted population model to see how our coefficients were sized.

* **H0 sum(Coefficients) = 0**
* **H1 sum(Coefficients) > 0**
  + **Find robust subset**
  + **We set to show that H1 is true using cross-validation and simulation to converge on significant terms using Monte Carlo class balanced resampling**

# Oversampling

To achieve equal training of response classes of 1’s and 0’s. We use stratified oversampling when constructing our data frame (Shmueli, 130). We initially followed a procedure outlined by Selva Prabhakaran to size the set of 0’s to be equal to the set of 1’s. This is over/undersampling to fit a specific class. We initially tried this method then retooled our algorithm to treat both classes equally when it comes to which class to sample from. In essence, we sample from records where Y is of a given response value (either 1 or 0) until our data frame is constructed properly.

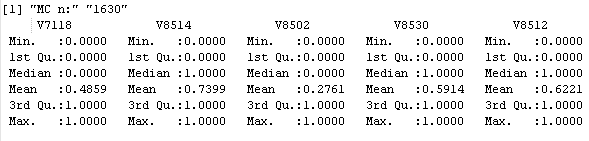
The main logic behind the resampling technique is through deriving the minimum between a % (25 in this example) of the size of available ones and again of zeros, this gives us our **sample size** we call *minFactor*. We then divide this minimum *minFactor* by the average between that same % in size of available ones and zeros [in this case halved, which merely doubles the iterations by halving the sample size]. This gives us *reloopFactor*. The logic looks like this.

* avgCountHalved <- round(mean(length(input\_ones[,1]),length(input\_zeros[,1]))/2)
* minFactor <- min(round(.25\*length(input\_ones[,1])),round(.25\*length(input\_zeros[,1])))
* reloopFactor <- min(round(.25\*length(input\_ones[,1])),round(.25\*length(input\_zeros[,1])))/round(.25\*avgCountHalved)

The symmetrical proportional class balancing code we use in saveCSVs.R can be found in the **Appendix** under the subsection titled **Oversampling Code**. The code we based ours on, which is not symmetrical, is given below, PhD author Selva Prabhakaran states they specifically want better sensitivity which is a concept they termed Concordance. The reason we deviate from this setup is that despite having implemented a medianDirection flag in cleandataCode.R that flips the conditional operant for transforming 1’s by a greater than rather than greater than or equal to median condition, we deviated from the example code in an attempt to achieve maximum specificity and sensitivity and forcing a cutoff threshold of .5 so we can equally test for 1’s and 0’s. Alternatively. We were using this code as intended by sampling from the zeros at the size of the sample of ones and deriving new transformed variables to work with. However, we desired to simplify the code and developed a symmetrical balancing approach which does indeed have higher sensitivity than a population fitted model dominated by 0’s which elsewise performs on par with the population model in terms of overall accuracy.



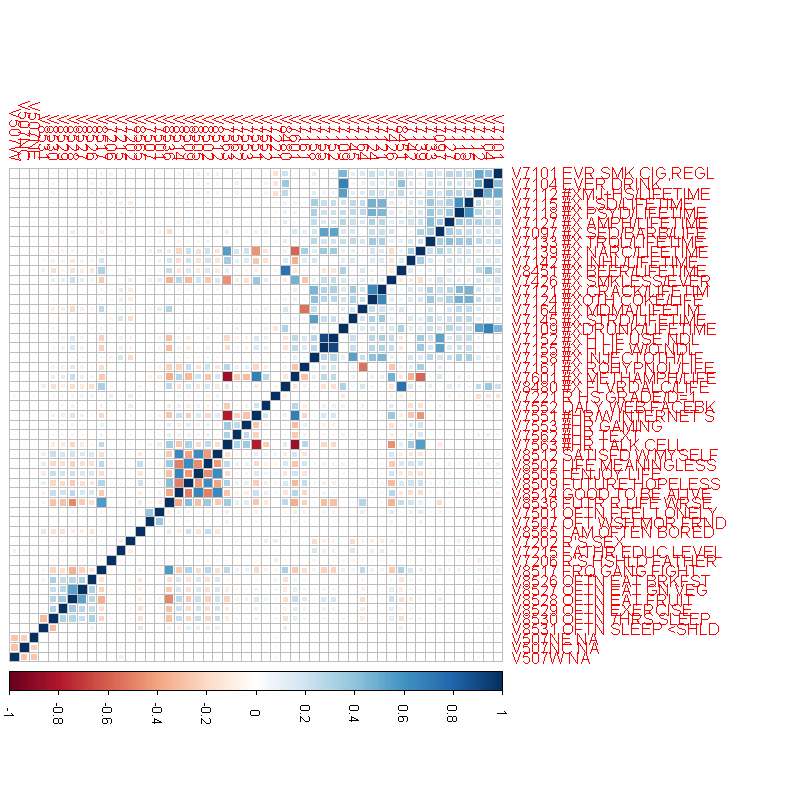
Summary dataframe output.



We use stratified sampling, or proportional resampling of specific datasets to achieve a desired distribution. We then use the means from the monte carlo resampled dataframes for our predictor effect plots. The reason this works is because measures of central tendency dictate that population mean will surface over repeated sampling. Which is exactly what we are doing. Resampling from the subset of Y’s that are 1’s and then 0’s at a specific size each pass.

The difference from Selva’s code is we randomly sample from the non-desired class at an equal amount as the desired class but we base this amount not on the size of the ones, but on a percentage based on the minimum of both classes and then derive a reloop factor to address being undersized to any of the particular classes. These constructed dataframes are the “gauntlet” we run our regression equations through to be able to properly predict a 1 or a 0 based on how 1’s and 0’s appear within the population. This “Monte Carlo” model is then compared to a model which has been overfitted to the population data.

# Correlation Matrix (Factor Grouping)



This is one of our earliest plots where our na’s were merely coded as 0. Here, we group similar terms together visually based on their correlations. This plot was made to see what factors were relevant and how we should split up our search space. For example, V8512 through V8546 were seen to be highly correlated with each other and therefore were treated as a group to be included or excluded or pruned in pairs (as the psychology questions seemed to alternate between red and blue). V8517 through V8531 were grouped as Health, V7551 through V7563 were grouped as habits, and V7501 through V7206 were grouped as one-off’s. The substances (V7101 to V8480) all clearly were correlated with each other (upper right blue squarish area).

We referred to this chart to consider which variables or class thereof to drop once we did decide on which to include. We dropped all but psychedelic substances and later during development when we ran into an issue with bestglm and na’s, we dropped V7501 and V7507 for having more than 75% NA’s.

## Model Assumptions

Quoting Statistic Solutions for Binary Logistic Model Assumptions.

First, binary logistic regression requires the dependent variable to be binary and ordinal logistic regression requires the dependent variable to be ordinal.

Second, logistic regression requires the observations to be independent of each other. In other words, the observations should not come from repeated measurements or matched data.

Third, logistic regression requires there to be little or no multicollinearity among the independent variables. This means that the independent variables should not be too highly correlated with each other.

Fourth, logistic regression assumes linearity of independent variables and log odds. although this analysis does not require the dependent and independent variables to be related linearly, it requires that the independent variables are linearly related to the log odds.

In our check for multicollinearity, we initially checked all pairwise x combinations to see if they had a greater correlation with each other than with the response term (Shmueli, 206) during 4thpass.R. However, we check our binary logistic regression model for multicollinearity at the end checking VIF’s (which was stated as usable for binary logistic regression by UCLA as well as page 269) using the mcstat library and was preferable to coding it.

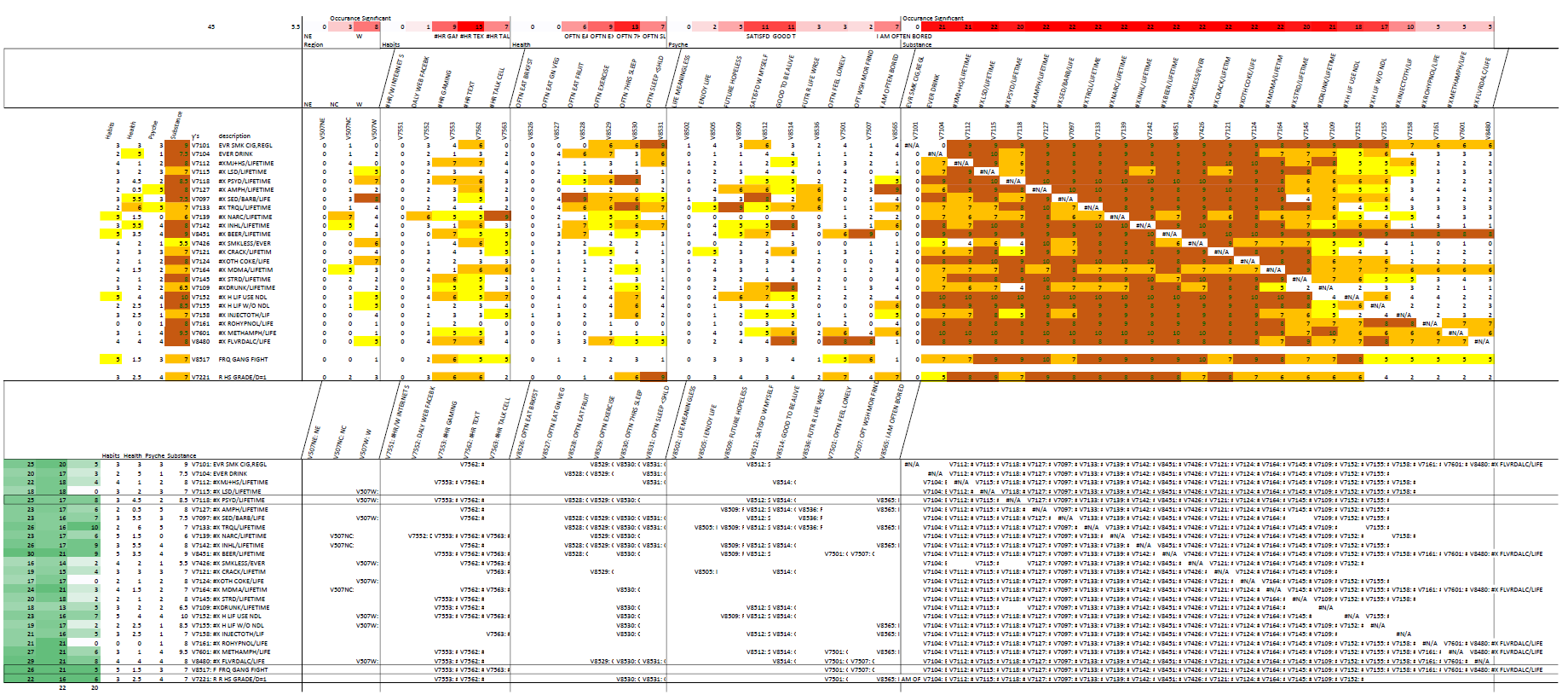
# Regression using StepAIC (early method)

During initial discovery phase. We attempted to use StepAIC to find the best regression equation as well as attempt to find the best equation for each substance. However we realized we were doing two things. One, the cardinal sin of overfitting our model to training data, and two. Two many y responses. We needed 3 questions. So we used StepAIC to filter down our substances to one category.

We structured a type of data partitioning that made use of a *holdout* and *non holdout set* for us to do stepAIC factor pruning from. We gave the holdout analysis an initial set and iterated over each substance as a response, feeding it a giant list of factors and watching what survived stepAIC and tabulating the results. What you see here is the results of 10 runs and we highlighted results that appeared 5 or more times.

We coded categorical variables (V507NE, V507NC, V507W) which were subsequently dropped as they were not found relevant under any cross tabulation results. Based on the large number of factors tabulated, Psychedelic drug use was the substance we decided to focus on for further development.

It is interesting to note that the areas highlighted in red were the tabulations of factors which met the threshold of 5 or more. And “texting”, “sleep”, “satisfaction with oneself”, and “feelings of good to be alive” were associated the most with drug use when using stepAIC.



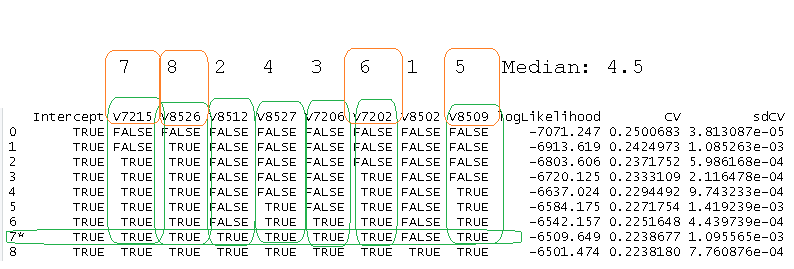
Despite not going with StepAIC. Our plan began to focus around finding a filtering algorithm, whatever it may be (stepAIC or bestGLM) and let the tabulation do the filtering and view the results after a few runs at various settings. We initially tried stepAIC but realized it wasn’t using exhaustive search spaces and would often result in nothing to report, so we switched to a modified bestglm.

# sub\_returnCVnames (Modified BestGLM)

We make use of a modified version of BestGLM called *sub\_returnCVNames*. BestGLM itself is a best subsets approach that makes use of binary logistic regression as well as cross validation. The picture below showcases how our modification deviates from the standard output of bestglm. The example happens to be a fluke from the normal behaviour of bestglm (e.g. the majority of the number of factors given as inputs returned as outputs). We have provided the code in the **Appendix**, under the section titled **Modified bestGLM**.

We use bestglm modified to further reduce the list in half regardless of minimum error and base the results on >= median column tabulation of the factors present in the [best subset] list. These tabulations are highlighted in orange. The best subset is highlighted in green (as well as an \*asterisk) which is the best subset with the smallest number of factors within 1 standard error of minimum k. In spite of this example, the *modified* subset algorithm generally preserves *more* than what bestglm would normally output and it is this feature that makes it useful for our filtering purposes.

With the way sub\_returnCVNames returns “above or equal to” median values, both values would have to be present in the minimum model to be returned, meaning rarely would both terms be returned, generally just one is returned.



# Factor Reduction (cleandatacode.R)

Our system is not parallel, meaning it processes commands one at a time. Using this as an analogy, the system can be explained as a type of moving football in which there is only one focus at any given moment. This “football” is an ever evolving set of terms as it shuffles between NamesTV training partition data.train and its counterpart NamesH which is the holdout (data.test) version of itself.

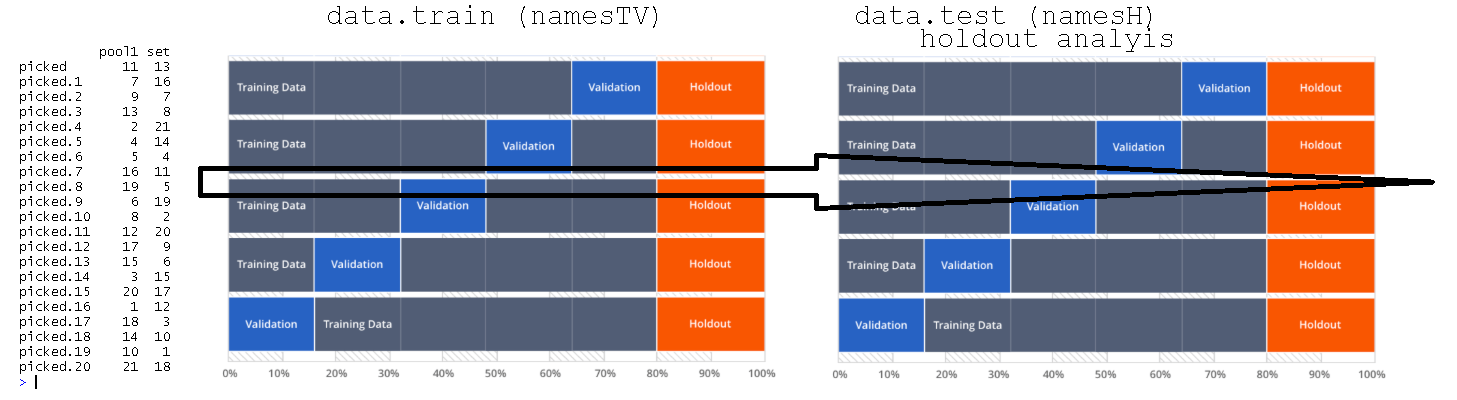
The way these terms are tabulated is through the pairing of two factors at a time. We do this by calling pairedLists.R which takes a number (up to 21 in our case for 21 factors) and generates two randomized lists used for factor pairing. The code had to be written in such a way so as to know not to assign the same number to the same element between the two lists (so a factor isn’t paired with itself) .These pairs of numbers are used to generate *xpair*s in the code.

Because we are only working with pairs of factors and (modified) bestGLM, this means that only occasionally will both factors be picked, what is the normative case; however, is that one of the factors will be returned from the pair. The goal we are trying to achieve by pairing is to minimize the complexity of the overwhelming problem of dealing with 21 factors and dropped rows from NA’s as well as reduce the information cost of finding best subsets of 21 factors and working with subsamples by simply bifurcating at the smallest level. This, not that. Between two factors, pick one and repeat that process using randomization. This has the benefit of reducing the problem to 21 individual elements that can be iterated as opposed to trying to find the best combination between 21 elements.

During NamesH list generation is where we apply holdout analysis. We don’t use traditional concepts like using RMSE and trying to find the best model. Rather we use the concept of having a separate holdout partition to do more testing from.

Each pass, using the paired lists we mentioned earlier, we generate two lists of up to 21 index pairs for indexing factor column names and passing these names along with a data partition to sub\_returnCVNames. NamesTV for data.train and NamesH for data.test. If the terms match a given row element between both lists (i.e. meaning they passed a given iteration of a list of xpairs), the resultant factor is tabulated.

An array *crossValidated* holds these successes. The factor may show up to twice each pass (each factor is paired off twice between two lists each run). So we divide the # of successes each pass by 2.

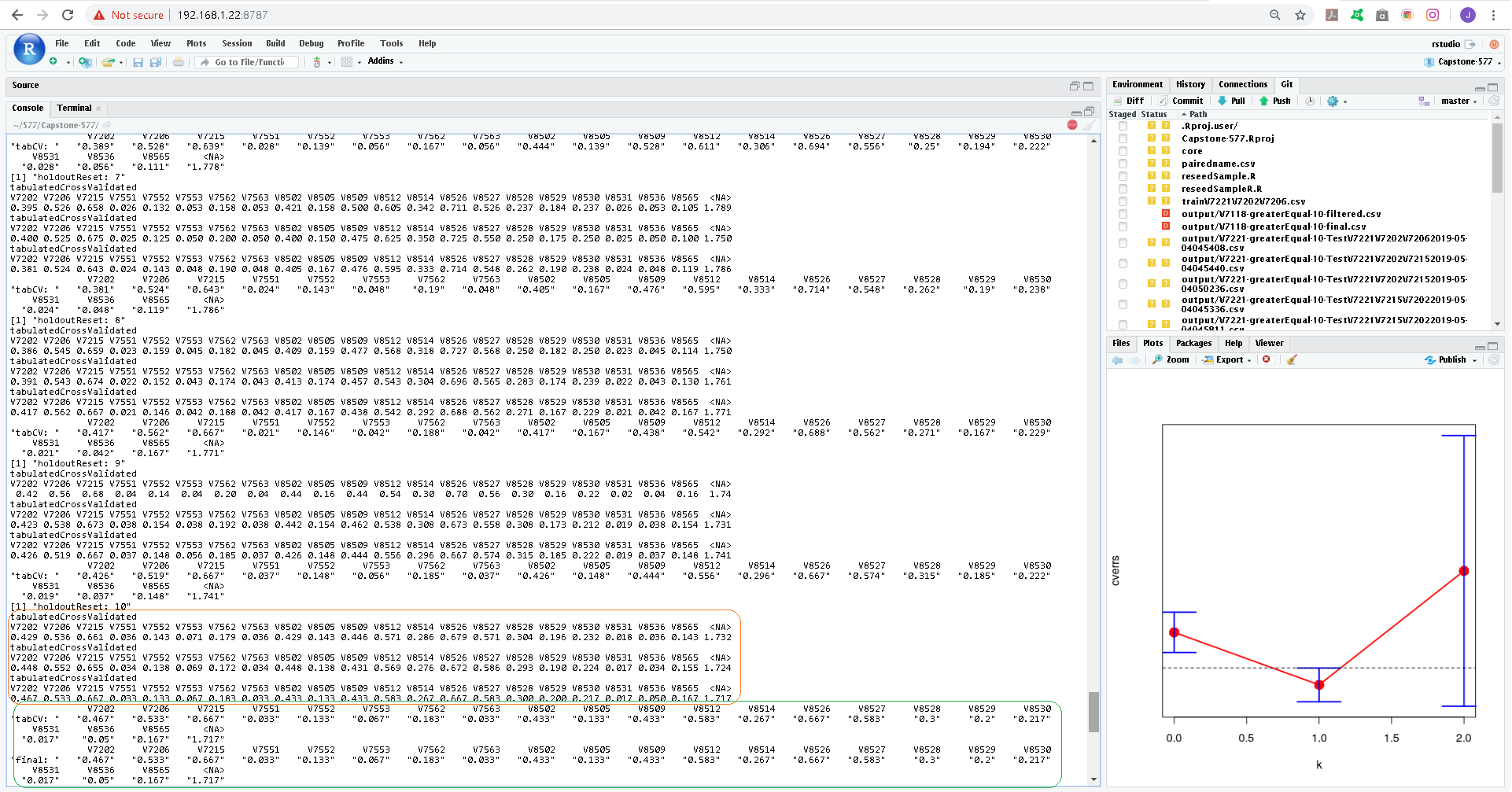


## Data Partitioning: Loops & Resampling

To simplify the subsampling and randomization. Data.train and data.test being resampled each [resample] inner loop (x3) from a holdout and pretrain (non holdout) set (partitions) which are resampled 10% each from the population at large each [holdoutresetloop] outer loop (x10). The outer loops purpose is to unscramble these outer subsample locations through Monte Carlo resampling of their indexes.

The inner resample loops are to “exhaust” these external macro set locations by monte carlo resampling (i.e. with replacement) from their indexes over 3 passes (at 33% each) each pass. This results in 10 (holdoutreset loops) x 2 samples (one for holdout set and one for preTrain set [drawn from prenonholdoutSet] ) x3 resample loops.

Each holdout and pretrain set are sampled 10 times at 10% holdoutreset loop for two sets. Each of these sets is tested 3 times at 33% sampling. At 10 holdoutreset loops with 3 resample loops equals 30 number of runs (loops) total.



The figure above details in orange each resample’s 3 tabulated results. There are 10 sets of these 3 resamples. I.e 10 holdout x 3 resample loops makes for 30 number of runs (numRuns) by which the tabulated results are divided against.

NA’s are tabulated as well. The way the lists can only ever allow at most 2 unique names or a pair of NA’s (2 NA’s vs 2 unique names) results, which means NA’s are reported at a greater quantity than the factors of interest. This results in a threshold of 1 as the max a factor can show as (but NA’s are up to 4). With .5 representing the value reported if a factor shows up at least once every run.

At the end of all the holdout passes crossValidated is translated to tabulatedCrossValidated (encircled in green) and a final.csv is printed with the factors occurence stored as “Freq”. Output from V7221’s final.csv is provided.

"","tabulatedCrossValidated","Freq"

"16",NA,2.889

"3","V7215",0.611

"11","V8527",0.556

"1","V7202",0.444

"8","V8512",0.389

"10","V8526",0.389

"12","V8528",0.389

"2","V7206",0.333

"7","V8509",0.333

"14","V8530",0.333

"6","V8505",0.278

"9","V8514",0.222

"13","V8529",0.222

"15","V8565",0.222

"4","V7562",0.167

"5","V8502",0.111

We set our final threshold for >.35. Which means the factor shows up somewhere between a minimum of more than every other run (> .25) and less than every run (.5).

# SaveCSVs.R

The terms coming out of cleandatacode.R (i.e. the \*final.csv) are not checked for how well they work with each other, and hence are likely to be collinear. To find always terms that will always converge as significant at the population level, we rely on the filtering power of a class balanced monte carlo constructed dataframe to give us our final sub\_returnCVNames subset from this potentially collinear list.

## Model Comparison with Population fitted model

We then compare model parameter’s derived from the bestglm output from the class balanced monte carlo partition we created against the population’s best fitted model. We then compare ROC plots (Shmueli 120) and Confusion Matrix accordingly (Shmueli 115)

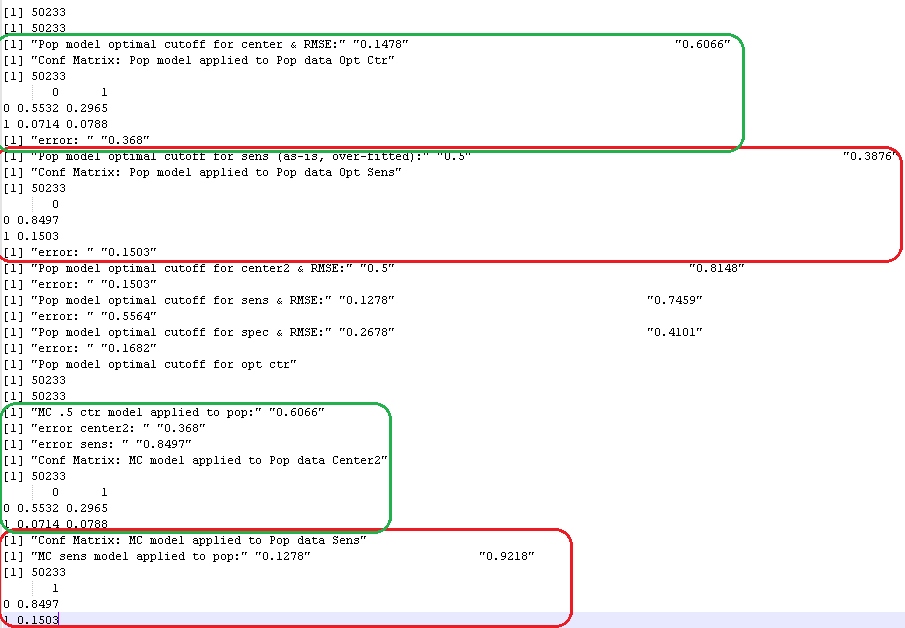
Our model has a higher sensitivity yet performed worse in the 1st and 3rd population fitted comparisons in overall accuracy. This is because the population fitted model failed to detect any 1’s where-as ours did. This is because the predicted values of the logos of the populated fitted model were all below .5 (Shmueli, 270). Further analysis in trying various cut-off thresholds might actually return some 1’s, or maybe not depending on how the model was trained. Where-as our monte carlo based models performed near the accuracy levels they stated they would for the monte carlo dataframe.

The purpose of saveCSVs.R is merely to show how our model does in comparison against population data. When the class is balanced, such as with the 2nd model V7221, our robust model performs nearly as well as the population fitted model (.6222 RMSE for population vs .6215 for ours). This is actually pretty decent considering that our monte carlo simulated model is based on 20405 records and the population fitted model is based on 48693.

When adjusting cutoff threshold for optimal threshold at the population, our model compared the same as the population overfitted model. Telling us our model is optimized at .5 threshold.

### Milestone 18

With a model overfitted to the population, we derive confusion matrix when the cutoff is @ .5 (normal output). Then based on the plogis output, derive optimal thresholds for both sensitivity and the "both" option (i.e. both sensitivity and specificity) which will be different from .5 for this model (top), where we compare results to the monte carlo models (bottom). Two different plogis with two different models trained on two different constructed datasets converge on the same error rating.



# Conclusion

Our system converged on significant formulas establishing response term relationships. These formulas can be relied on to find response positive factors of interest at .5 cutoff threshold when a population overfitted model cannot. When we adjusted the population overfitted model’s plogis output for optimal sensitivity and “both” thresholds respectively, our monte carlo class balanced model converged on the same confusion matrix error rates as the population overfitted model. Because our model is trained on significantly less data than the population set and is specifically noisy (resampled from) yet converges on the same results as the overfitted model when adjusting the population plogis for optimized thresholds, this tells us we can using the monte carlo model at the derived thresholds in the future to achieve similar accuracy results.

## High Level Inferences

Holding all other factors at their population means, increase chance of presence of

**V7118 Psychedelic Drug Use, presence of**:

Given a 50% chance of psychedelic drug use these are the factors of influence. (Note, only 2.7% of the population has answered positive).

* V8502: “Life is meaningless response” +~21%.
* V8512: “Satisfied with myself” -~19%

Error Rate: 38.03%

**V7221 GPA (B+)**

Given a 50% chance of having a B+ of not. These are the influencers. Note: 56% of the population have a B+ or lower.

* V8502: Life Meaningless: ~14% increase
* V8509: Future Hopeless: ~ 14% increase
* V8527: Often Eat Green Vegetables: ~2.5% Decrease
* V8526: Often Eat Breakfast: 7.5% Increase

Error Rate: 38.63%

**V8517 Gang Fights**

Given a 50% propensity to gang violence. Gang violence exists in the population at 15%.

* V8502: Life Meaningless: ~14% increase
* V8509: Future Hopeless: ~ 14% increase
* V8527: Often Eat Green Vegetables: ~2.5% Decrease
* V8526: Often Eat Breakfast: 7.5% Increase

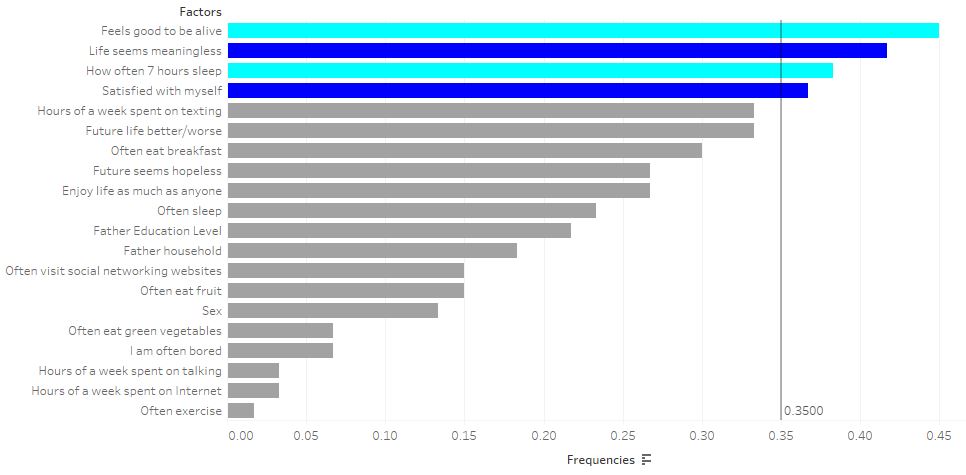
# Further considerations

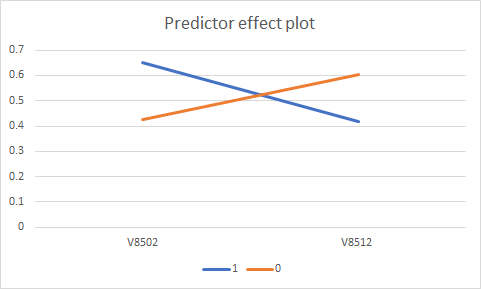
Use one of the years as a holdout set to construct and then use as test data to compare the final models against. For example, splitting d\_2017 from d\_combined early on during cleandatacode.R.

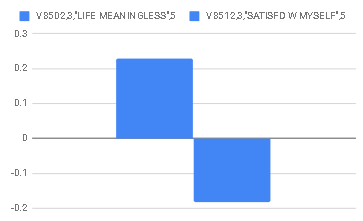
Using these y terms as a base for questions. One could construct a questionnaire and use the results to predict the likelihood someone is or will become one of these response categories. However, the intended goal of the study was to find influencers in hopes of reducing the likely outcome of a response.

# Logistic Regression Equations

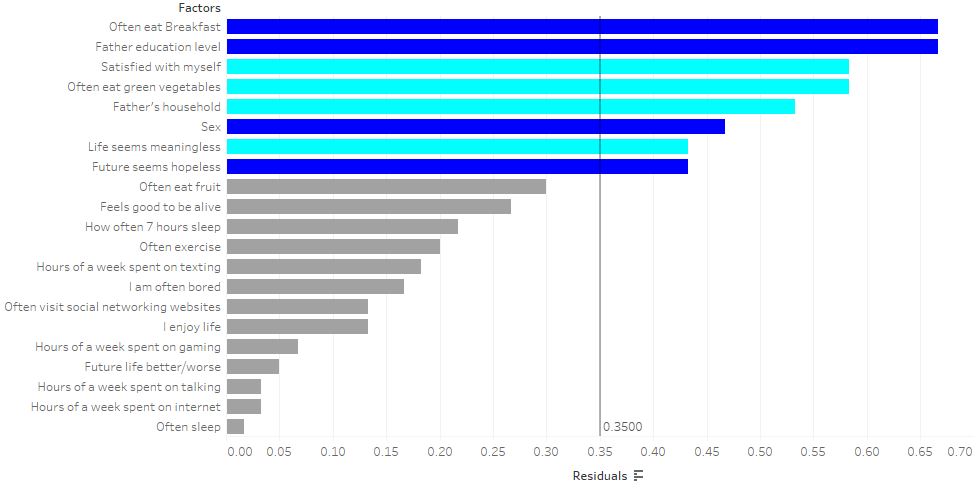
**V7118**=exp(0.1551+0.9366\*V8502-0.7405\*V8512)/1+exp(0.1551+0.9366\*V8502-0.7405\*V8512)

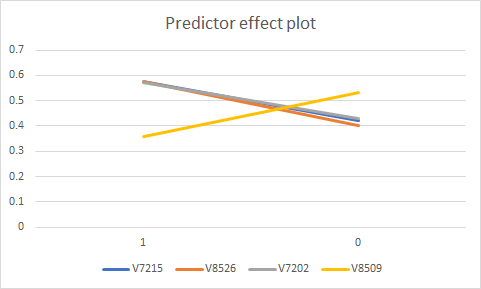


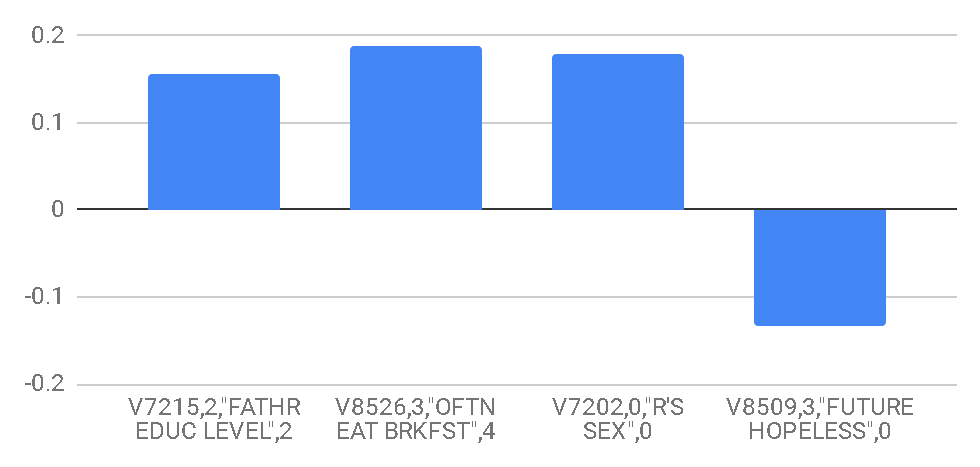




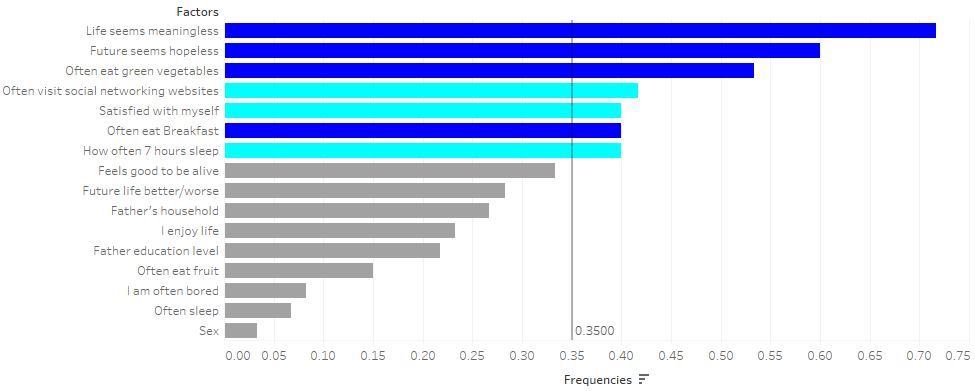
**V7221**=exp(-0.8971+0.6308\*V7215+0.6977\*V8526+0.5753\*V7202-0.7231\*V8509)/1+exp(-0.8971+0.6308\*V7215+0.6977\*V8526+0.5753\*V7202-0.7231\*V8509)

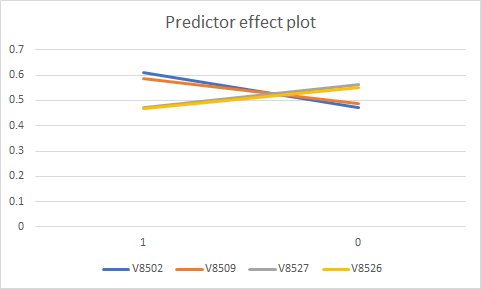


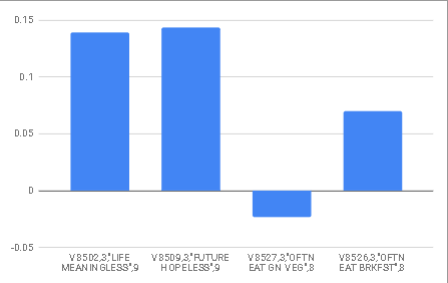




**V8517**=exp(0.2047+0.5651\*V8502+0.4090\*V8509-0.3567\*V8527-0.3257\*V8526)/1+exp(0.2047+0.5651\*V8502+0.4090\*V8509-0.3567\*V8527-0.3257\*V8526)







# 

# References

**“LESSON 3 LOGISTIC REGRESSION DIAGNOSTICS.”** *IDRE Stats*, **UCLA, stats.idre.ucla.edu/stata/webbooks/logistic/chapter3/lesson-3-logistic-regression-diagnostics/.**

**“Assumptions of Logistic Regression.” Statistics Solutions**, ***Statistics Solutions*,** [**www.statisticssolutions.com/assumptions-of-logistic-regression/**](http://www.statisticssolutions.com/assumptions-of-logistic-regression/)**.**

Ucl. “Confidence Intervals for a Median.” *UCL Great Ormond Street Institute of Child Health*, 2 Nov. 2018, [www.ucl.ac.uk/child-health/short-courses-events/about-statistical-courses/research-methods-and-statistics/chapter-8-content-8](http://www.ucl.ac.uk/child-health/short-courses-events/about-statistical-courses/research-methods-and-statistics/chapter-8-content-8).

**McLeod, A I, and C Xu. “Bestglm: Best Subset GLM.” *Universidad Del Pais Vasco*, CRAN, www.et.bs.ehu.es/cran/web/packages/bestglm/vignettes/bestglm.pdf.**

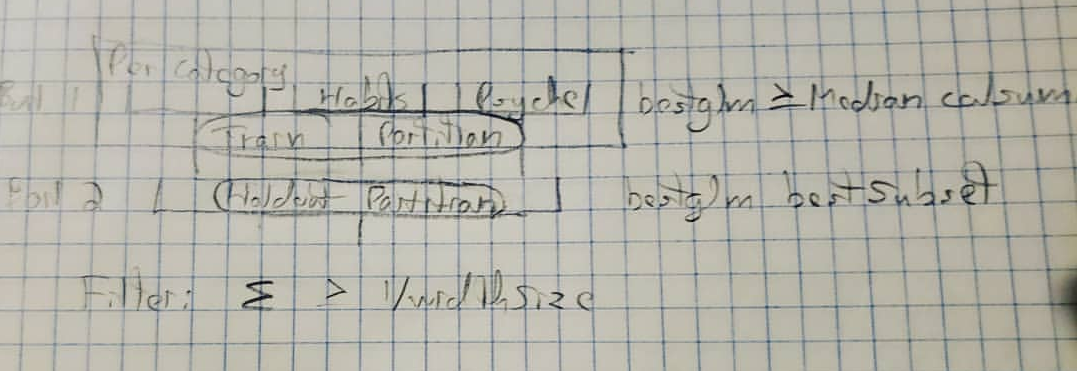
Pardoe, Iain. *Applied regression modeling*. Hoboken, NJ: Wiley, 2012. Print

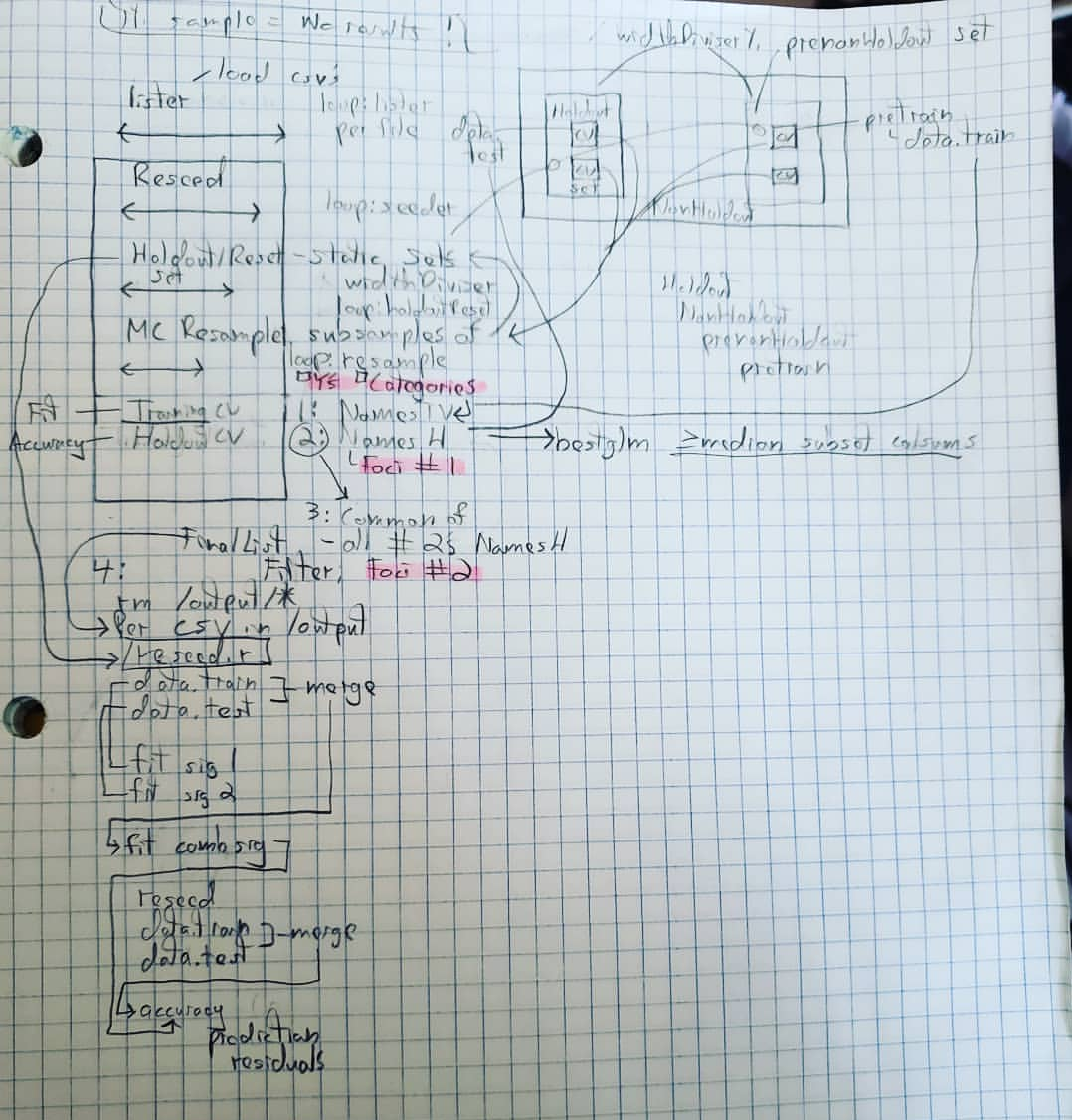
Prabhakaran, Selva. **“Logistic Regression with R.”** *Logistic Regression With R*, **r-statistics.co/Logistic-Regression-With-R.html.**

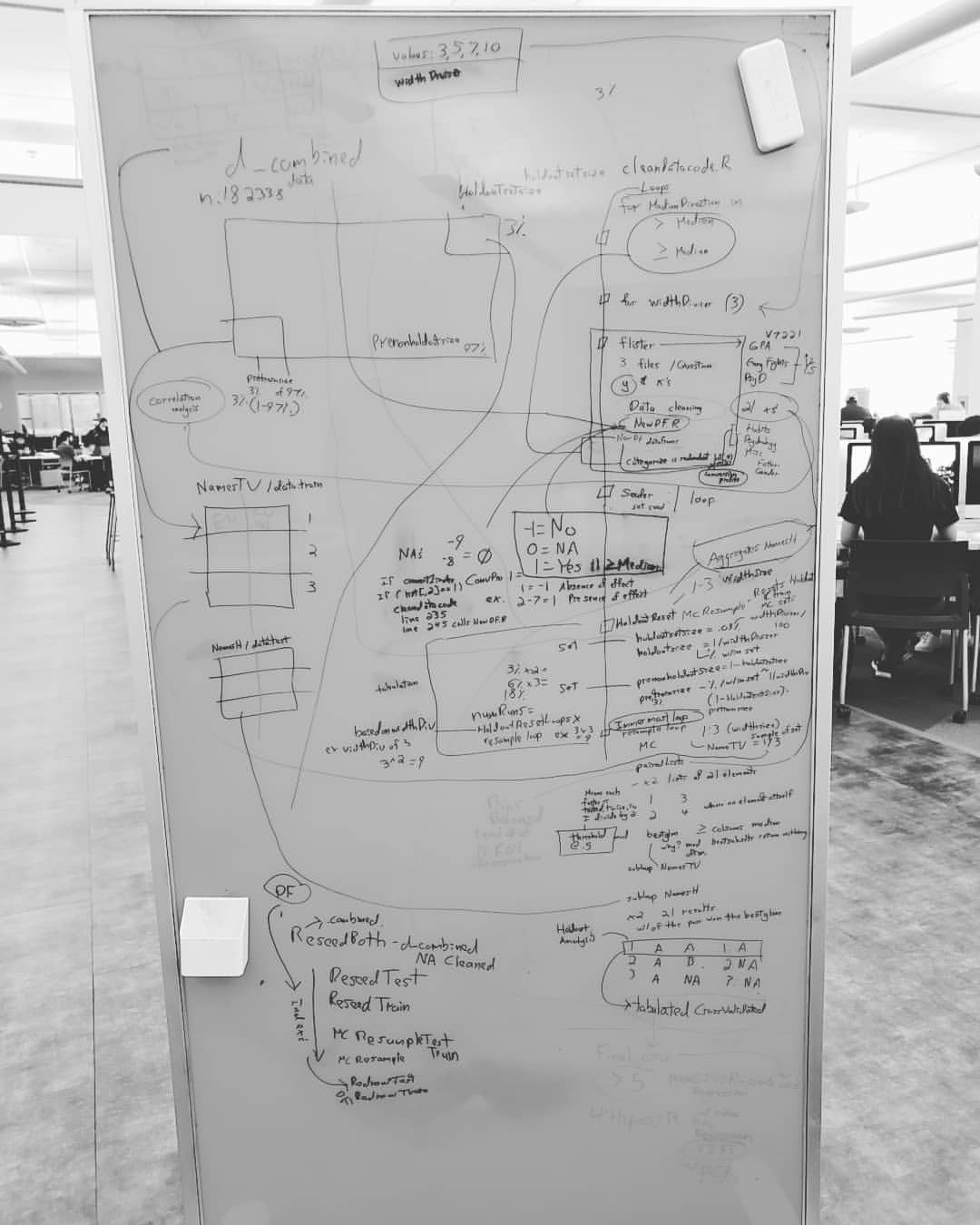
Shmueli, Galit, Peter C. Bruce, and Nitin R. Patel. *Data mining for business analytics : concepts, techniques, and applications with XLMiner*. Hoboken, New Jersey: John Wiley & Sons, 2016. Print.

# Appendix

## Early Diagrams







## V7221 Median Conversion

#https://www.ucl.ac.uk/child-health/short-courses-events/about-statistical-courses/research-methods-and-statistics/chapter-8-content-8

#95% confidence

#7: B+

#95% conf confirmed

#if flister==1(gpa)

#NewDF is referenced by saveCSV's, so by disabling this, I prevent referencing it

#if(flister==1)

#NewDF[,"V7221"]

V7221\_Index <- c()

if (medianDirection=="greaterEqual") V7221\_Index <- NewDF[,"V7221"] >= median(NewDF[,"V7221"][NewDF[,"V7221"]>0])

if (medianDirection=="greater") V7221\_Index <- NewDF[,"V7221"] > median(NewDF[,"V7221"][NewDF[,"V7221"]>0])

centerpoint = (length(NewDF[,"V7221"][NewDF[,"V7221"]>0]))/2

width = round(1.96\*sqrt((length(NewDF[,"V7221"][NewDF[,"V7221"]>0])))/2)

lower = (length(NewDF[,"V7221"][NewDF[,"V7221"]>0]))/2 - width

upper = (length(NewDF[,"V7221"][NewDF[,"V7221"]>0]))/2 + width

sort(((NewDF[,"V7221"][NewDF[,"V7221"]>0])))[lower]

sort(((NewDF[,"V7221"][NewDF[,"V7221"]>0])))[upper]

NewDF[V7221\_Index,"V7221"] <- 21

V7221\_Index <- c()

V7221\_IndexNotAbove <- NewDF[,"V7221"] != 21

V7221\_IndexNotAboveNotZero <- c()

V7221\_IndexNotAboveNotZero <- NewDF[V7221\_IndexNotAbove,"V7221"] != 0

NewDF[V7221\_IndexNotAbove,"V7221"][V7221\_IndexNotAboveNotZero] <- -1

tempIndex <- c()

tempIndex <- NewDF["V7221"]==21

NewDF[tempIndex,"V7221"] <- 1

summary(NewDF[,"V7221"])

## Median Confidence

for (interests in c("V7221","V7215","V7551","V7552","V7553","V7562","V7563"))

{

#median information

print(paste("interest:",interests))

for(year in c("d\_2012","d\_2013","d\_2014","d\_2015","d\_2016","d\_2017","d\_combined"))

{

#https://stackoverflow.com/questions/28802652/access-variable-dataframe-in-r-loop

df <- (get(year)[,interests])

print(paste("year:",year))

print(paste("count:",sum (count(df[df>0]))))

centerpoint = (length(df[df>0]))/2

#print(centerpoint)

width = round(1.96\*sqrt((length(df[df>0])))/2)

lower = (length(df[df>0]))/2 - width

upper = (length(df[df>0]))/2 + width

print(paste("lower:", sort(((df[df>0])))[lower]))

print(paste("median:",median(df[df>0])))

print(paste("upper:",sort(((df[df>0])))[upper]))

print(round(table ( df[(df>0)] ) / sum (count(df[df>0])) ,4))

#https://stackoverflow.com/questions/9317830/r-do-i-need-to-add-explicit-new-line-character-with-print

writeLines("\n")

}

writeLines("\n")

}

## Oversampling Code

avgCountHalved <- c()

avgCountHalved <- round(mean(length(input\_ones[,1]),length(input\_zeros[,1]))/2)

trainingData <- c()

ones.index <- c()

zeros.index <- c()

reloopFactor <- c()

minFactor <- c()

minFactor <- min(round(.25\*length(input\_ones[,1])),round(.25\*length(input\_zeros[,1])))

reloopFactor <- min(round(.25\*length(input\_ones[,1])),round(.25\*length(input\_zeros[,1])))/round(.25\*avgCountHalved)

remainder <- c()

remainder = reloopFactor-floor(reloopFactor)

#i=1

#this might break depending on the size of 1's or 0's, but I hope

for(i in 1:4)

{

if(floor(reloopFactor)>0)

{

for (loops in 1:floor(reloopFactor))

{

#generates index and samples in place. I have to do this, else repeat index's get stored as .1's and .2' respectively

ones.index <- rbind(ones.index,input\_ones[sample(c(rownames(input\_ones)), minFactor),]) # 1's for training

zeros.index <- rbind(zeros.index,input\_zeros[sample(c(rownames(input\_zeros)), minFactor),]) # 0's for training. Pick as many 0's as 1's

}

}

ones.index <- rbind(ones.index,input\_ones[sample(c(rownames(input\_ones)), minFactor\*remainder),]) # 1's for training

zeros.index <- rbind(zeros.index,input\_zeros[sample(c(rownames(input\_zeros)), minFactor\*remainder),]) # 0's for training. Pick as many 0's as 1's

both <- c()

both <- rbind(ones.index, zeros.index)

#summary(both)

#https://stackoverflow.com/questions/2370515/how-to-get-row-index-number-in-r

mix <- c()

mix <- sample(c(rownames(both)),round(nrow(both)/2) )

#mix <- sample(both,length(both)/2)

#colnames(mix) <- colnames(trainingData)

trainingData <- rbind(trainingData, both[mix,])

}

#not reduced column data

finalTraining <- c()

finalTrainingI <- c()

size <- c()

size <- round(nrow(trainingData))

finalTrainingI <- sample(c(rownames(trainingData)),size/4)

finalTraining <- trainingData[finalTrainingI,]

print(c("MC n:",nrow(finalTraining)))

summary(finalTraining)

## Modified BestGLM Code

sub\_returnCVNames <- function(data\_sent){

#data\_sent=data.train

holderOfData <- cbind(Filter(var,data.frame(data\_sent[,-1 , drop = FALSE])),data.frame(data\_sent[,1 , drop = FALSE]))

#table(NewDF[,"V7202"])

info <- which(colSums(holderOfData)==nrow(holderOfData))

name <- rownames(data.frame(info))

if(!length(info)==0) holderOfData <- holderOfData[, -which(names(holderOfData) == name)]

if ( widthDiviser == 1 ) B <- suppressMessages(bestglm(Xy = holderOfData, IC="CV", CVArgs=list(Method="HTF", K=2, REP=widthDiviser, TopModels=widthDiviser, BestModels = widthDiviser), family=binomial,method = "exhaustive"))

if (!(widthDiviser == 1 )) B <- suppressMessages(bestglm(Xy = holderOfData, IC="CV", CVArgs=list(Method="HTF", K=5, REP=3, TopModels=widthDiviser, BestModels = widthDiviser), family=binomial,method = "exhaustive"))

set<-round(colSums(B$Subsets))[-1]

if(!is.null(B$Subsets))

{

cverrs = B$Subsets[, "CV"]

sdCV = B$Subsets[, "sdCV"]

CVLo = cverrs - sdCV

CVHi = cverrs + sdCV

ymax = max(CVHi)

ymin = min(CVLo)

k = 0:(length(cverrs) - 1)

if(!(ymax=="Inf" || ymax=="-Inf")) plot(k, cverrs, ylim = c(ymin, ymax), type = "n", yaxt = "n")

points(k,cverrs,cex = 2,col="red",pch=16)

lines(k, cverrs, col = "red", lwd = 2)

axis(2, yaxp = c(0.6, 1.8, 6))

segments(k, CVLo, k, CVHi,col="blue", lwd = 2)

eps = 0.15

segments(k-eps, CVLo, k+eps, CVLo, col = "blue", lwd = 2)

segments(k-eps, CVHi, k+eps, CVHi, col = "blue", lwd = 2)

indMin = which.min(cverrs)

fmin = sdCV[indMin]

cutOff = fmin + cverrs[indMin]

abline(h = cutOff, lty = 2)

indMin = which.min(cverrs)

fmin = sdCV[indMin]

cutOff = fmin + cverrs[indMin]

min(which(cverrs < cutOff))

}

left=length(set)-3

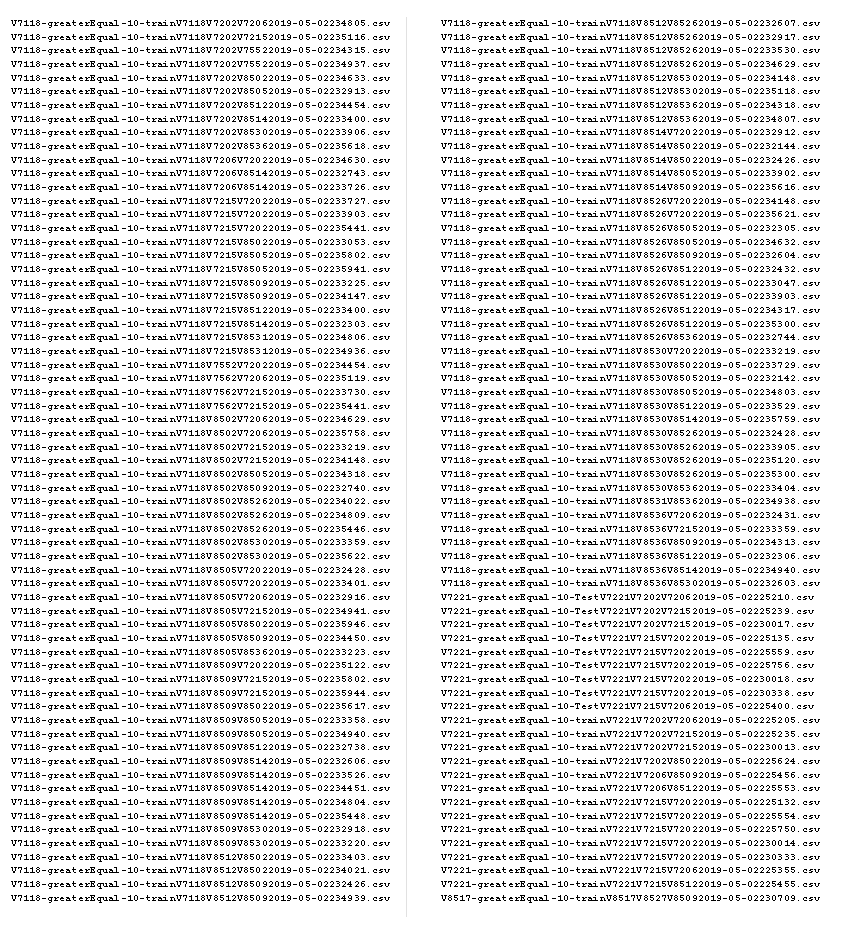
result <- set[1:left]

#aboveMedianCV <- as.character(rownames(data.frame(which(result >= median(result)))))

return(as.character(rownames(data.frame(which(result >= median(result))))))

}

## bestGLM errors



## Nullpairs

