Economic Impacts Assessment

Scott Coffin

12/23/2020

Table of Contents

# Survey Completeness

Since data are missing for some categories within responses, we have three choices: 1) listwise-deletion: remove rows that contain missing data. This will, of course, reduce the strength of the dataset. 2) mean/median substitution: another quick fix that takes the mean/median of the existing data points and substitutes the missing data points. This would obviously bias the analysis since it decreases variance. 3) Multiple imputation: With this approach, rather than replacing missing values with a single value, we use the distribution of the observed data/variables to estimate multiple possible values for the data points. This allows us to account for the uncertainty around the true value, and obtain approximately unbiased estimates (under certain conditions). Moreover, accounting for uncertainty allows us to calculate standard errors around estimations, which in turn leads to a better sense of uncertainty for the analysis.

To decide which option to use, let’s first examine how much data is missing from our survey.

#make subset of data showing complete cases (rows that have full coverage of variables) for the systems that COMPLETED the survey and were asked to participate (i.e. no volunteers).  
sub <- allSmalls.requested.responded %>% select(cash\_reserve\_unrestricted, cash\_reserve\_restricted, cash\_reserve\_total, months\_before\_assist, delinquent\_num\_acc, delinquent\_amount\_dollars)  
#how many NA's are present total  
#sum(is.na(sub))  
#how many complete cases total  
#sum(complete.cases(sub))  
sum(complete.cases(sub)) / nrow(sub)

## [1] 0.4137931

We can see that a rather large proportion (41%!) of survey data is missing just for the selected few variables, which are considered to be the primary responses of interest in this survey. If we were to simply exclude incomplete cases, we would trim a substantive poriton of the dataset, potentially reducing the power of the study to unnacceptable levels. The table below takes a deeper look at the number of missing values in each column for the probabilistic and non-probabilistic samples.

rm(sub) #keep it clean  
  
#create dataframe with completeness  
completenessSummary <- allSmalls %>%  
 filter(voluntary == "y" |voluntary == "n") %>%  
 select(voluntary, cash\_reserve\_unrestricted, cash\_reserve\_restricted, cash\_reserve\_total, months\_before\_assist, delinquent\_num\_acc, delinquent\_amount\_dollars) %>%  
 group\_by(voluntary) %>%  
 summarise\_all((name = ~sum(is.na(.))/length(.)))   
#convert to matrix and transpose  
transposed <- as.data.frame(t(as.matrix(completenessSummary[2:7])))  
#reassign column names  
colnames(transposed) <- c("Probabilistic", "Non-Probabilistic")  
transposed

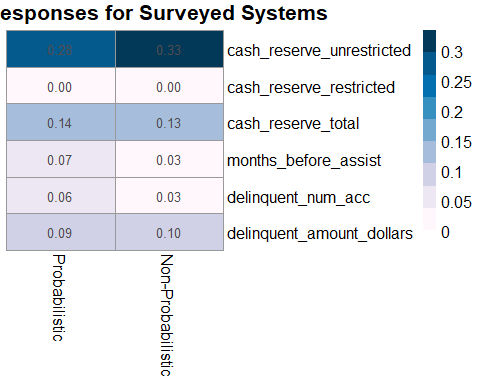
## Probabilistic Non-Probabilistic  
## cash\_reserve\_unrestricted 0.27851459 0.33333333  
## cash\_reserve\_restricted 0.52785146 0.51282051  
## cash\_reserve\_total 0.13793103 0.12820513  
## months\_before\_assist 0.06631300 0.02564103  
## delinquent\_num\_acc 0.06366048 0.02564103  
## delinquent\_amount\_dollars 0.08753316 0.10256410

There are no obvious differences in response completeness between the probabilistic and non-probabilistic samples, as demonstrated in the above table. This provides reassurance that blending these samples is viable. Further, District Staff report that small systems often did not report whether cash was restricted or unrestricted due to the lack of such designations or restrictions in their accounting systems. Therefore, it is a reasonable assumption that missing values for restricted cash reserve may be recorded as zero. This correction takes place in the below code chunk.

#recode restricted cash reserves to 0  
allSmalls$cash\_reserve\_restricted[is.na(allSmalls$cash\_reserve\_restricted)] <- 0  
  
#create dataframe with completeness with restricted cash reserve question as 0  
completenessSummary <- allSmalls %>%  
 filter(voluntary == "y" |voluntary == "n") %>%  
 select(voluntary, cash\_reserve\_unrestricted, cash\_reserve\_restricted, cash\_reserve\_total, months\_before\_assist, delinquent\_num\_acc, delinquent\_amount\_dollars) %>%  
 group\_by(voluntary) %>%  
 summarise\_all((name = ~sum(is.na(.))/length(.)))   
#convert to matrix and transpose  
transposed <- as.data.frame(t(as.matrix(completenessSummary[2:7])))  
#reassign column names  
colnames(transposed) <- c("Probabilistic", "Non-Probabilistic")  
transposed

## Probabilistic Non-Probabilistic  
## cash\_reserve\_unrestricted 0.27851459 0.33333333  
## cash\_reserve\_restricted 0.00000000 0.00000000  
## cash\_reserve\_total 0.13793103 0.12820513  
## months\_before\_assist 0.06631300 0.02564103  
## delinquent\_num\_acc 0.06366048 0.02564103  
## delinquent\_amount\_dollars 0.08753316 0.10256410

#format as matrix  
MissingMatrix <- data.matrix(transposed)  
#build heatmap  
pheatmap(MissingMatrix,  
 main = "Missing Responses for Surveyed Systems", #title  
 fontsize = 12,  
 cluster\_rows = FALSE, cluster\_cols = FALSE,#disable dendrograms  
 display\_numbers = TRUE,  
 treeheight\_row = 0, treeheight\_col = 0, #keeps clustering after dropping dendrograms  
 col = brewer.pal(n = 9, name = "PuBu")) #blue color scheme with 9 colors)

 It’s clear that both the probabilistic and non-probabilistic surveyed water systems provided similiar levels of information in this survey. Let’s see if there were differences in response completeness by sampling bin.

tagCompletenessSummary <- allSmalls %>%  
 filter(voluntary == "y" |voluntary == "n") %>%  
 select(tag, cash\_reserve\_unrestricted, cash\_reserve\_restricted, cash\_reserve\_restricted, cash\_reserve\_total, months\_before\_assist, delinquent\_num\_acc, delinquent\_amount\_dollars) %>%  
 group\_by(tag) %>%  
 summarise\_all((name = ~sum(is.na(.))/length(.))) %>%   
 mutate(across(is.numeric,~round(., 2)))

## Warning: Problem with `mutate()` input `..1`.  
## i Predicate functions must be wrapped in `where()`.  
##   
## # Bad  
## data %>% select(is.numeric)  
##   
## # Good  
## data %>% select(where(is.numeric))  
##   
## i Please update your code.  
## This message is displayed once per session.  
## i Input `..1` is `across(is.numeric, ~round(., 2))`.

## Warning: Predicate functions must be wrapped in `where()`.  
##   
## # Bad  
## data %>% select(is.numeric)  
##   
## # Good  
## data %>% select(where(is.numeric))  
##   
## i Please update your code.  
## This message is displayed once per session.

tagCompletenessSummary

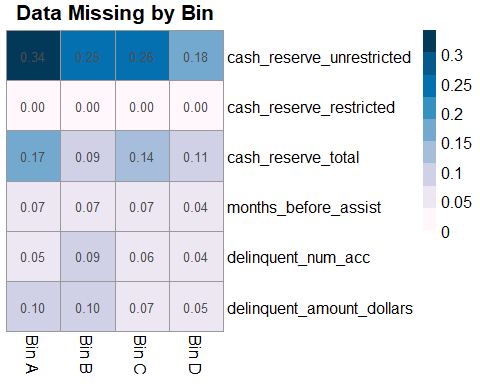
## # A tibble: 4 x 7  
## tag cash\_reserve\_unrestricted cash\_reserve\_restricted cash\_reserve\_total  
## <fct> <dbl> <dbl> <dbl>  
## 1 Bin A 0.34 0 0.17  
## 2 Bin B 0.25 0 0.09  
## 3 Bin C 0.26 0 0.14  
## 4 Bin D 0.18 0 0.11  
## months\_before\_assist delinquent\_num\_acc delinquent\_amount\_dollars  
## <dbl> <dbl> <dbl>  
## 1 0.07 0.05 0.1   
## 2 0.07 0.09 0.1   
## 3 0.07 0.06 0.07  
## 4 0.04 0.04 0.05

Again, these data may be easier to view as a heatmap.

#convert to matrix and transpose  
transposedTag <- as.data.frame(t(as.matrix(tagCompletenessSummary[2:7]))) #1 is category, 2-6 are variables  
#reassign column names  
colnames(transposedTag) <- c("Bin A", "Bin B", "Bin C", "Bin D")  
transposedTag

## Bin A Bin B Bin C Bin D  
## cash\_reserve\_unrestricted 0.34 0.25 0.26 0.18  
## cash\_reserve\_restricted 0.00 0.00 0.00 0.00  
## cash\_reserve\_total 0.17 0.09 0.14 0.11  
## months\_before\_assist 0.07 0.07 0.07 0.04  
## delinquent\_num\_acc 0.05 0.09 0.06 0.04  
## delinquent\_amount\_dollars 0.10 0.10 0.07 0.05

#format as matrix  
MissingMatrixTag <- data.matrix(transposedTag)  
#build heatmap  
pheatmap(MissingMatrixTag,  
 main = "Data Missing by Bin", #title  
 fontsize = 12,  
 cluster\_rows = FALSE, cluster\_cols = FALSE,#disable dendrograms  
 display\_numbers = TRUE,  
 treeheight\_row = 0, treeheight\_col = 0, #keeps clustering after dropping dendrograms  
 col = brewer.pal(n = 9, name = "PuBu")) #blue color scheme with 9 colors)

 There seems to be a clear step-wise trend for the willingness to reportunresicted cash reserve by size (Bin A are smallest, Bin D are largest). This tracks with the above statement regarding District Staff reponses from surveys. Furthermore, unrestricted cash reserves are not applicable to small systems that do not have restrictions, so these missing values are not an issue. Luckily, no other obvious trends for missing response values appear to be present. Let’s see if there’s a relationship between reporting of this value and service connections.

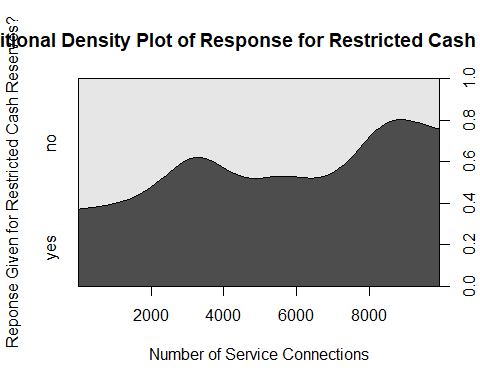
rm(MissingMatrixTag, tagCompletenessSummary, completenessSummary, transposed, transposedTag)  
#convert numeric to 1  
allSmalls.responded <- allSmalls.responded %>%   
 mutate(cash\_reserve\_restricted\_NA = case\_when(cash\_reserve\_restricted >= 0 ~ 1))  
  
#Convert NA to 0  
allSmalls.responded$cash\_reserve\_restricted\_NA <- allSmalls.responded$cash\_reserve\_restricted\_NA %>% replace\_na(0) %>% as.factor()  
#convert to factor  
allSmalls.responded <- allSmalls.responded %>%   
 mutate(cash\_reserve\_restricted\_NA\_yn = case\_when(cash\_reserve\_restricted\_NA == "0" ~ "no",  
 cash\_reserve\_restricted\_NA == "1" ~ "yes"))   
allSmalls.responded$cash\_reserve\_restricted\_NA\_yn <- as.factor(allSmalls.responded$cash\_reserve\_restricted\_NA\_yn)  
  
#examine relationship of response to restricted reseved cash question with number of service connections  
  
lgit<- glm(cash\_reserve\_restricted\_NA ~ Service\_Connections + Fee\_Code,   
 data = allSmalls.responded, family = "binomial")  
summary(lgit)

##   
## Call:  
## glm(formula = cash\_reserve\_restricted\_NA ~ Service\_Connections +   
## Fee\_Code, family = "binomial", data = allSmalls.responded)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.8538 -0.9708 -0.8431 1.2038 1.6242   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -0.90728038 0.30915646 -2.935 0.003339 \*\*   
## Service\_Connections 0.00018978 0.00005792 3.277 0.001051 \*\*   
## Fee\_CodeDAVCL 1.28236170 0.33537707 3.824 0.000131 \*\*\*  
## Fee\_CodeDAVCS -0.12001029 0.60124208 -0.200 0.841790   
## Fee\_CodeSC 0.37875059 0.34164433 1.109 0.267598   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 575.53 on 415 degrees of freedom  
## Residual deviance: 537.17 on 411 degrees of freedom  
## AIC: 547.17  
##   
## Number of Fisher Scoring iterations: 4

Here we can see that a significant relationship exists between number of service connections and the response level for the question regarding restricted cash reserves. There also seems to be a significant relationship between the disadvantaged community larges, which is related to service connections.

This may also be visualized in a conditional density plot.

cdplot(cash\_reserve\_restricted\_NA\_yn ~ Service\_Connections, data = allSmalls.responded,  
 main = "Conditional Density Plot of Response for Restricted Cash Reserves", xlab = "Number of Service Connections", ylab = "Reponse Given for Restricted Cash Reserves?")



Let’s further test these grouping differences in response with a chi-squared test.

wald.test(b = coef(lgit), #coefficients from logit model   
 Sigma = vcov(lgit), #variance covariance matrix of the error terms  
 Terms = 3:5) #categorical terms (fee codes)

## Wald test:  
## ----------  
##   
## Chi-squared test:  
## X2 = 15.9, df = 3, P(> X2) = 0.0012

The p-value of 0.0035 again suggests that fee code is a significant predictor for willingness to respond to this question. An easier way to conceptualize this may be through odds-ratios.

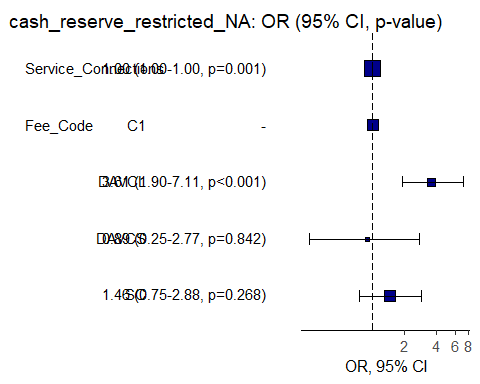
#exponentiate the coefficients (i.e. create odds-ratios)  
round(exp(cbind(OddsRatio = coef(lgit), confint(lgit))),2)

## OddsRatio 2.5 % 97.5 %  
## (Intercept) 0.40 0.22 0.73  
## Service\_Connections 1.00 1.00 1.00  
## Fee\_CodeDAVCL 3.61 1.90 7.11  
## Fee\_CodeDAVCS 0.89 0.25 2.77  
## Fee\_CodeSC 1.46 0.75 2.88

Now we can clearly see that disadvantaged community Larges are 3.47 times more likely to report their unrestricted cash reserves (97.5% CI: 1.83 - 6.84). Odds ratios are plotted below:

#remove model   
rm(lgit)  
#define explanatory variables  
explanatory = c("Service\_Connections", "Fee\_Code")  
#plot ORs  
allSmalls.responded %>%   
 or\_plot("cash\_reserve\_restricted\_NA",  
 explanatory, table\_text\_size=4, title\_text\_size=14,   
 plot\_opts=list(xlab("OR, 95% CI"), theme(axis.title = element\_text(size=12))))

## Warning: Removed 1 rows containing missing values (geom\_errorbarh).



Since we’ve determined (both statistically, and qualitatively from District Staff responses) that the reporting of restricted/unrestricted cash reserves is dependent on accounting systems that are reflective of the system’s relative size, a reliable assumption may be made that total cash reserves can be expressed as unrestricted cash reserves for systems that only report totals and no restriction status. This simple conversion will allow for meaningful comparisons and reduce the amount of missing data while retaining high confidence by avoiding the need for sophisticated approaches (e.g. imputation) that would introduce additional uncertainty.

A simple logical test is used to create a new variable called “cash\_reserve\_aggregate”, which is equivalent to total cash reserves if no information is provided for restricted cash reserves, or unrestricted cash reserves if information is provided for restricted cash reserves.

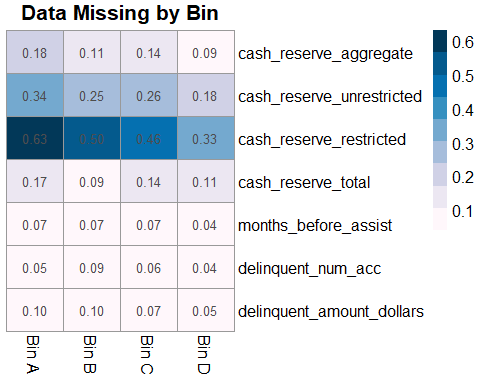
#recode with new variable  
allSmalls.responded %<>%   
 mutate(cash\_reserve\_aggregate = ifelse(cash\_reserve\_restricted\_NA\_yn == "no", cash\_reserve\_total, cash\_reserve\_unrestricted))

The filling in of missing data from the above conversion may be visualized in the heatmap below.

#create table  
tagCompletenessSummary <- allSmalls.responded %>%   
 select(tag, cash\_reserve\_aggregate, cash\_reserve\_unrestricted, cash\_reserve\_restricted, cash\_reserve\_restricted, cash\_reserve\_total, months\_before\_assist, delinquent\_num\_acc, delinquent\_amount\_dollars) %>%  
 group\_by(tag) %>%  
 summarise\_all((name = ~sum(is.na(.))/length(.))) %>%   
 mutate(across(is.numeric,~round(., 2)))   
  
#convert to matrix and transpose  
transposedTag <- as.data.frame(t(as.matrix(tagCompletenessSummary[2:8]))) #1 is category, 2-6 are variables  
#reassign column names  
colnames(transposedTag) <- c("Bin A", "Bin B", "Bin C", "Bin D")  
transposedTag

## Bin A Bin B Bin C Bin D  
## cash\_reserve\_aggregate 0.18 0.11 0.14 0.09  
## cash\_reserve\_unrestricted 0.34 0.25 0.26 0.18  
## cash\_reserve\_restricted 0.63 0.50 0.46 0.33  
## cash\_reserve\_total 0.17 0.09 0.14 0.11  
## months\_before\_assist 0.07 0.07 0.07 0.04  
## delinquent\_num\_acc 0.05 0.09 0.06 0.04  
## delinquent\_amount\_dollars 0.10 0.10 0.07 0.05

#format as matrix  
MissingMatrixTag <- data.matrix(transposedTag)  
#build heatmap  
pheatmap(MissingMatrixTag,  
 main = "Data Missing by Bin", #title  
 fontsize = 12,  
 cluster\_rows = FALSE, cluster\_cols = FALSE,#disable dendrograms  
 display\_numbers = TRUE,  
 treeheight\_row = 0, treeheight\_col = 0, #keeps clustering after dropping dendrograms  
 col = brewer.pal(n = 9, name = "PuBu")) #blue color scheme with 9 colors)



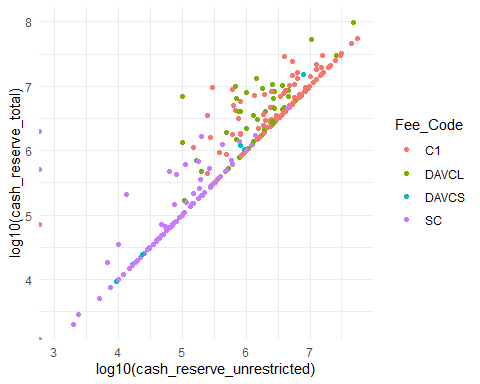
In the above heatmap, it can be seen that the the newly created variable “cash\_reserve\_aggregate” has virtually the same amount of missing data as the total cash reserve variable, thus confirming that the conversion process was successful.

While data is missing for unrestricted cash reserves, it may be possible to estimate this from other data. This may be easily visualized in a dot plot.

allSmalls.requested.responded %>%   
 ggplot(aes(x = log10(cash\_reserve\_unrestricted), y = log10(cash\_reserve\_total), color = Fee\_Code)) +   
 geom\_point() +  
 theme\_minimal()

## Warning in FUN(X[[i]], ...): NaNs produced  
  
## Warning in FUN(X[[i]], ...): NaNs produced  
  
## Warning in FUN(X[[i]], ...): NaNs produced  
  
## Warning in FUN(X[[i]], ...): NaNs produced

## Warning: Removed 117 rows containing missing values (geom\_point).



We can see discrete grouping of cash reserve (total and unrestricted) by fee code with community larges (C1) and disadvantaged community larges have higher cash reserves, while small community (SC) and disadvantaged small communities have lower cash reserves. There also seems to be a clear relationship between these variables, with some highly significant outliers. Independent linear regression models may be used to predict missing data, however more sophisticated methods are likely necessary to reliably predict outliers.

Since there are many missing data, we cannot simply rely on single models for each variable. It is therefore necessary to perform multiple imputation to preserve our dataset. I will use the mice (which stands for Multivariate Imputation by Chained Equations) package which is developed by Stef van Buuren. The basic steps of multiple imputation are described by Rubin (1976). *Citation: Rubin, Donald B. 1976. “Inference and missing data.” Biometrika 63, no. 3: 581-592.*

# Multiple Imputation

*The following text is copied, sometimes verbatim, from the [University of Virgina Library] (*[*https://data.library.virginia.edu/getting-started-with-multiple-imputation-in-r/*](https://data.library.virginia.edu/getting-started-with-multiple-imputation-in-r/)*)*

1. impute the missing values by using an appropriate model which incorporates random variation.
2. repeat the first step 3-5 times.
3. perform the desired analysis on each data set by using standard, complete data methods.
4. average the values of the parameter estimates across the missing value samples in order to obtain a single point estimate.
5. calculate the standard errors by averaging the squared standard errors of the missing value estimates. After this, calculate the variance of the missing value parameter across the samples. Finally, combine the two quantities in multiple imputation for missing data to calculate the standard errors.

Put in a simpler way, we a) choose values that keep the relationship in the dataset intact in place of missing values b) create independently drawn imputed (usually 5) datasets c) calculate new standard errors using variation across datasets to take into account the uncertainty created by these imputed datasets (Kropko et al. 2014). *Citation: Kropko, Jonathan, Ben Goodrich, Andrew Gelman, and Jennifer Hill. 2014. “Multiple imputation for continuous and categorical data: Comparing joint multivariate normal and conditional approaches.” Political Analysis 22, no. 4.*

## Missing Data Assumptions

Rubin (1976) classified types of missing data in three categories: MCAR, MAR, MNAR

MCAR: Missing Completely at Random – the reason for the missingness of data points are at random, meaning that the pattern of missing values is uncorrelated with the structure of the data. An example would be a random sample taken from the population: data on some people will be missing, but it will be at random since everyone had the same chance of being included in the sample.

MAR: Missing at Random – the missingness is not completely random, but the propensity of missingness depends on the observed data, not the missing data. An example would be a survey respondent choosing not to answer a question on **income** because they believe the privacy of personal information. As seen in this case, the missing value for income can be predicted by looking at the answers for the personal information question.

MNAR: Missing Not at Random – the missing is not random, it correlates with unobservable characteristics unknown to a researcher. An example would be social desirability bias in survey – where respondents with certain characteristics we can’t observe systematically shy away from answering questions on racial issues.

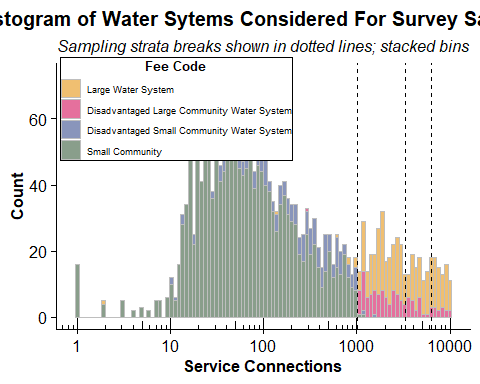
All multiple imputation techniques start with the MAR assumption. While MCAR is desirable, in general it is unrealistic for the data. Thus, researchers make the assumption that missing values can be replaced by predictions derived by the observable portion of the dataset. This is a fundamental assumption to make, otherwise we wouldn’t be able to predict plausible values of missing data points from the observed data.

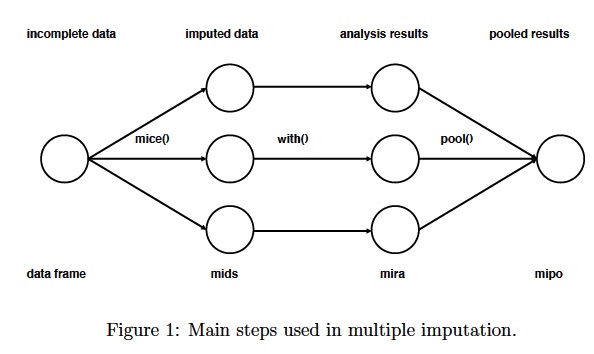
There are two approaches to multiple imputation, implemented by different packages in R:

* Joint Multivariate Normal Distribution Multiple Imputation: The main assumption in this technique is that the observed data follows a multivariate normal distribution. Therefore, the algorithm that R packages use to impute the missing values draws values from this assumed distribution. Amelia and norm packages use this technique. The biggest problem with this technique is that the imputed values are incorrect if the data doesn’t follow a multivariate normal distribution.
* Conditional Multiple Imputation: Conditional MI, as indicated in its name, follows an iterative procedure, modeling the conditional distribution of a certain variable given the other variables. This technique allows users to be more flexible as a distribution is assumed for each variable rather than the whole dataset.

If our data appears highly normal, we may use the first option. Let’s examine service connections for small systems surveyed (<10,000).

allSmalls %>%   
ggplot(aes(x = Service\_Connections, fill = Fee\_Code)) +   
 geom\_histogram(position = "stack",alpha=0.6, bins = 100, color = "gray") +  
 scale\_x\_log10() +  
 annotation\_logticks(base = 10, sides = "b")+ #only bottom  
 labs(x='Service Connections',y = "Count",  
 title = "Histogram of Water Sytems Considered For Survey Sampling", subtitle = "Sampling strata breaks shown in dotted lines; stacked bins")+  
 scale\_fill\_manual(name = "Fee Code",   
 labels = c("Large Water System", "Disadvantaged Large Community Water System", "Disadvantaged Small Community Water System", "Small Community"),  
 values = cal\_palette("superbloom3"))+  
 geom\_vline(xintercept = c(1009, 3315, 6360), linetype ='dashed') +  
 theme\_half\_open() +  
 theme(legend.position = c(0.01, 0.83),  
 legend.box.background = element\_rect(color = "black"),  
 legend.title = element\_text(size = 10, face = "bold", hjust = 0.5),  
 legend.text = element\_text(size = 7),  
 plot.title = element\_text(size = 15, face = "bold", hjust = 0.5),  
 plot.subtitle = element\_text(size = 12, hjust = 0.5, face = "italic"),  
 axis.title = element\_text(size = 12, face = "bold",),  
 axis.text = element\_text(size = 12))

 It’s highly unlikely that our data are very non-normal, as implied by the distributions of service connections. Therefore, we will use Conditional MI to impute values.



Three main steps to impute data. Source: University of Virgina Library

## Mice Overview

As the first step, the *mice* command creates several complete datasets (in the figure above, n=3). It considers each missing value to follow a specific distribution, and draws from this distribution a plausible value to replace the missing value.

These complete datasets are stored in an object class called *mids*, short for multiply imputed dataset. These datasets are copies of the original dataframe except that missing values are now replaced with values generated by mice. Since these values are generated, they create additional uncertainty about what the real values of these missing data points are. We will need to factor in this uncertainty in the future as we are estimating the regression coefficients from these datasets.

Now that we have 3 complete datasets, the next step is to run an ols regression on all these 3 datasets with 549 observations each (the dataset without incompletes has 182 observations). With *with\_mids* command, we run the ols regression and obtain a different regression coefficient for each dataset, reflecting the effect of service connection size on delinquent number of accounts. These 3 coefficients are different from each other because each dataset contains different imputed values, and we are uncertain about which imputed values are the correct ones. The analysis results are stored in a *mira* object class, short for multiply imputed repeated analysis.

Finally, we pool together the 3 coefficients estimated by the imputed dataset into 1 final regression coefficient, and estimate the variance using the *pool* command. With the assumption that regression coefficients are obtained from a multivariate normal distribution, in order to obtain the final coefficient we just take the mean of 3 values. We calculate the variance of the estimated coefficient by factoring in the within (accounting for differences in predicted values from the dataset regarding each observation) and between (accounting for differences between 3 datasets) imputation variance.

## Smalls Imputation

First let’s start my building an Ordinary Least Squares (OLS) linear regression to predict cash in reserve (total) based on relevant predictors for all systems (e.g. service connections, median 12 month household income, delinquent number of accounts, delinquent amout in dollars, CalEnvrioScreen 3 scores). We will ignore the volunteer data for now.

## Estimate an OLS Regression  
fitols <- lm(cash\_reserve\_total ~ Service\_Connections + Population + Median\_12month\_HH\_income +  
 Median\_rent\_pct\_income + CES\_3.0\_Score + delinquent\_num\_acc +  
 delinquent\_amount\_dollars, na.action = na.omit,   
 data = allSmalls.requested.responded)  
summary(fitols)

##   
## Call:  
## lm(formula = cash\_reserve\_total ~ Service\_Connections + Population +   
## Median\_12month\_HH\_income + Median\_rent\_pct\_income + CES\_3.0\_Score +   
## delinquent\_num\_acc + delinquent\_amount\_dollars, data = allSmalls.requested.responded,   
## na.action = na.omit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -16103839 -1900102 -843153 528138 94820450   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 970863.053 3833793.073 0.253 0.800265   
## Service\_Connections 1020.848 454.017 2.248 0.025304 \*   
## Population 169.968 110.159 1.543 0.123950   
## Median\_12month\_HH\_income -3.920 19.573 -0.200 0.841385   
## Median\_rent\_pct\_income 43364.437 89849.833 0.483 0.629725   
## CES\_3.0\_Score -64819.856 39300.586 -1.649 0.100173   
## delinquent\_num\_acc -5896.743 1683.818 -3.502 0.000535 \*\*\*  
## delinquent\_amount\_dollars 12.890 4.716 2.733 0.006658 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 7879000 on 287 degrees of freedom  
## (82 observations deleted due to missingness)  
## Multiple R-squared: 0.2641, Adjusted R-squared: 0.2461   
## F-statistic: 14.71 on 7 and 287 DF, p-value: 0.0000000000000002246

As we can see in the table above, a significant number of observations were deleted due to missingness (76 out of 371). Since this is greater than 14% of the whole dataset, our survey would be limited in power.

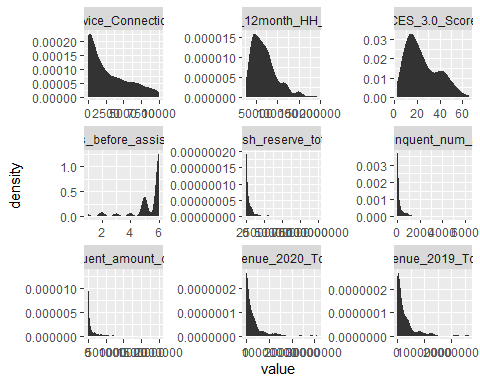
Time to impute. First we need to prepare the dataset for imputation. It’s best to keep it in it’s rawest form, so any categorical factors should be left as so, instead of using the ordinal transformed variable from above. Further, we want to remove variables that have haver than 25% missing values because they may mess up the imputation. It’s also important to remove variables that are highly correlated with others so as to stop the imputation working otherwise. Additionally, any extreme outliers should be removed, as they may dramatically impact results.

require(reshape2)

## Warning: package 'reshape2' was built under R version 4.0.3

allSmalls.requested.responded %>%   
 select(PWSID, Service\_Connections, Median\_12month\_HH\_income, CES\_3.0\_Score,  
 months\_before\_assist\_num, cash\_reserve\_total, delinquent\_num\_acc, delinquent\_amount\_dollars,  
 revenue\_2020\_Total, revenue\_2019\_Total) %>%   
 melt() %>% #convert wide to long  
 ggplot(aes(x = value)) +   
 stat\_density() +   
 facet\_wrap(~variable, scales = "free")

## Warning: Removed 295 rows containing non-finite values (stat\_density).

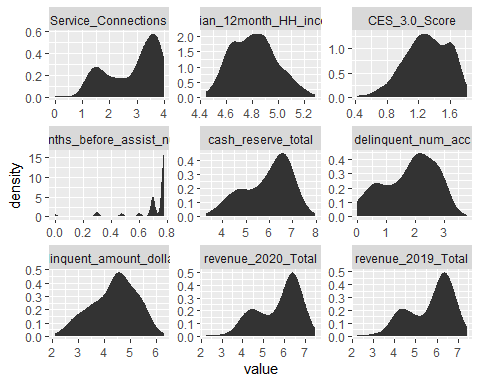
 Everything looks more or less realistic, however would benefit from log10 transformation for visualizing and possibly for statistics.

require(reshape2)  
allSmalls.requested.responded %>%   
 select(PWSID, Service\_Connections, Median\_12month\_HH\_income, CES\_3.0\_Score,  
 months\_before\_assist\_num, cash\_reserve\_total, delinquent\_num\_acc, delinquent\_amount\_dollars,  
 revenue\_2020\_Total, revenue\_2019\_Total) %>%   
 mutate\_if(~is.numeric(.) && (.) > 0, log10) %>%   
 melt() %>% #convert wide to long  
 ggplot(aes(x = value)) +   
 stat\_density() +   
 facet\_wrap(~variable, scales = "free")

## Warning: Problem with `mutate()` input `cash\_reserve\_total`.  
## i NaNs produced  
## i Input `cash\_reserve\_total` is `.Primitive("log10")(cash\_reserve\_total)`.

## Warning in mask$eval\_all\_mutate(dots[[i]]): NaNs produced

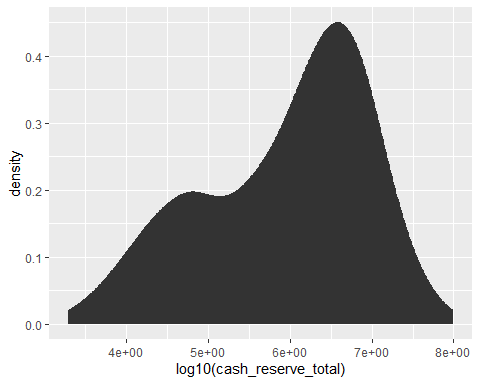
## Warning: Removed 417 rows containing non-finite values (stat\_density).

 Now distributions look more or less “normal.”

allSmalls.requested.responded %>%   
 filter(cash\_reserve\_total < 1000000000) %>%   
 ggplot(aes(x = log10(cash\_reserve\_total ))) +   
 scale\_x\_continuous(labels = scales::scientific) +  
 stat\_density()

## Warning in FUN(X[[i]], ...): NaNs produced

## Warning: Removed 11 rows containing non-finite values (stat\_density).

 We can see that there is a very wide gap between cash reserves of some systems (nearly 100 million dollars versus majority having less than 1 million). When plotted on a log10 scale, it’s easy to see this discrepancy. To ensure these aren’t typos, let’s take a closer look at these systems.

allSmalls.requested.responded %>%   
 #filter(cash\_reserve\_total > 100000) %>%   
 select(PWSID, cash\_reserve\_total, Service\_Connections, Population, Fee\_Code, Regulating\_Agency, Water\_System\_Name, comments\_cash\_reserves, comments\_exp\_rev, months\_before\_assist, delinquent\_num\_acc) %>%   
 arrange(desc(cash\_reserve\_total)) %>%   
 head()

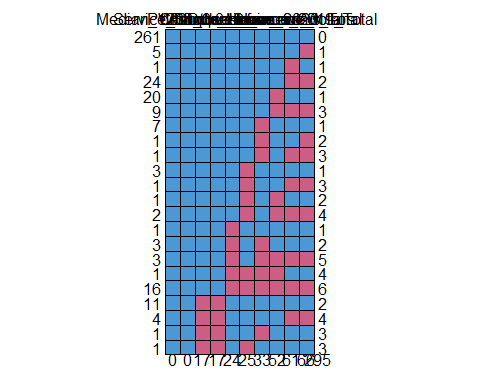
## PWSID cash\_reserve\_total Service\_Connections Population  
## 1 CA3310048 98514858 1586 4300  
## 2 CA1910204 56365000 7498 31204  
## 3 CA2410001 53592144 8419 29479  
## 4 CA3710018,CA3710044 46379028 8030 29955  
## 5 CA3710016 32517738 8254 44726  
## 6 CA3710023 30700000 6461 24509  
## Fee\_Code Regulating\_Agency Water\_System\_Name  
## 1 DAVCL DISTRICT 20 - RIVERSIDE COACHELLA VWD: I.D. NO. 8  
## 2 C1 DISTRICT 16 - CENTRAL LOS ANGELES CWWD 29 & 80-MALIBU  
## 3 DAVCL DISTRICT 11 - MERCED CITY OF ATWATER  
## 4 C1 DISTRICT 14 - SAN DIEGO RINCON DEL DIABLO MWD (ID-1 & ID-A)  
## 5 C1 DISTRICT 14 - SAN DIEGO RAINBOW MUNICIPAL WD  
## 6 C1 DISTRICT 14 - SAN DIEGO SANTA FE I.D.  
## comments\_cash\_reserves  
## 1 Reserves: these are the total restricted and unrestricted reserves for the Domestic water fund as of 6/30/20.  
## 2 5. Beginning fund balance for the district. Fund restricted for the operation, maintenance, repair, and construction of waterworks systems. There haven’t been any funds taken out of the reserve fund.  
## 3 Most delinquent accounts were delinquent prior to COVID-19. No shutoffs since November 2019. The 153 accounts include sewer, water and garbage services.  
## 4 Rincond Del Diablo has waived late fees for delinquent accounts since March 2020. Rincon ran query of their billing information to determine how many delinquent accounts the system has as of 11/12/2020.  
## 5 Rainbow has 827 delinquent accounts as of 10/31/2020 compared to 821 as of 1/31/2020. Tracey Largent, Rainbow Accountant, stated this is not a significant increase.  
## 6 9. The District has seen minimal increases in accounts receivable and operating impacts from the pandemic.  
## comments\_exp\_rev  
## 1 Information is an estimate; as CVWD bills all customers same rate as its larger water system. For this calculation; CVWD divided revenue and expenses by dividing the total of number of service connections in all of its three water systems and multiplying by the number of service connections for this particular water system.  
## 2 Billing is spread over on a monlthy basis. Water System is not submetered and the revenues are total.  
## 3 Aug revenue includes TCP settlement check: $9,229,462.00  
## 4 Includes all expenses and revenues for debt service, potable water, and recycled water funds. CIP expenses included. Water rates were increased on September 1, 2019. Some months show a significant increase in expenses, and these were due to CIP expenses that month.  
## 5 See backup tab for months that are allocated based on YTD totals.  
## 6 <NA>  
## months\_before\_assist delinquent\_num\_acc  
## 1 F 110  
## 2 F 423  
## 3 F 153  
## 4 F 206  
## 5 F 827  
## 6 F 65

While it is possible that the Coachella Valley Water District has nearly 100 million dollars in cash reserve, it is an order of magnitude larger than every other system in the survey, yet does have a huge number of service connections. For the sake of imputation, this system will be excluded.

# p\_missing <- unlist(lapply(allSmalls.responded, function(x) sum(is.na(x))))/nrow(smalls)  
# sort(p\_missing[p\_missing > 0], decreasing = TRUE)

Many of these variables are missing at high rates, such as sub-metered status,and unrestricted cash reserve. I’ll remove those. Also, the monetary values are bins and are not “missing” so much as they are just formatted long-wise binary. We’ll remove those too.

#define sub dataset  
allSmalls.simple <- allSmalls.requested.responded %>%   
 select(PWSID, Service\_Connections, Median\_12month\_HH\_income, CES\_3.0\_Score, months\_before\_assist, cash\_reserve\_total, delinquent\_num\_acc, delinquent\_amount\_dollars, revenue\_2020\_Total, revenue\_2019\_Total)  
  
#split the other half of the dataset for easy joining  
allSmalls.rest <- allSmalls %>%   
 filter(requested == "y") %>%   
 filter(responded == "y") %>%   
 select(-Service\_Connections, -Median\_12month\_HH\_income, -CES\_3.0\_Score, -months\_before\_assist, -cash\_reserve\_total, -delinquent\_num\_acc, -delinquent\_amount\_dollars, -revenue\_2020\_Total, -revenue\_2019\_Total)  
#see missing values  
md.pattern(allSmalls.simple)



## PWSID Service\_Connections Median\_12month\_HH\_income CES\_3.0\_Score  
## 261 1 1 1 1  
## 5 1 1 1 1  
## 1 1 1 1 1  
## 24 1 1 1 1  
## 20 1 1 1 1  
## 9 1 1 1 1  
## 7 1 1 1 1  
## 1 1 1 1 1  
## 1 1 1 1 1  
## 3 1 1 1 1  
## 1 1 1 1 1  
## 1 1 1 1 1  
## 2 1 1 1 1  
## 1 1 1 1 1  
## 3 1 1 1 1  
## 3 1 1 1 1  
## 1 1 1 1 1  
## 16 1 1 1 1  
## 11 1 1 0 0  
## 4 1 1 0 0  
## 1 1 1 0 0  
## 1 1 1 0 0  
## 0 0 17 17  
## delinquent\_num\_acc months\_before\_assist delinquent\_amount\_dollars  
## 261 1 1 1  
## 5 1 1 1  
## 1 1 1 1  
## 24 1 1 1  
## 20 1 1 1  
## 9 1 1 1  
## 7 1 1 0  
## 1 1 1 0  
## 1 1 1 0  
## 3 1 0 1  
## 1 1 0 1  
## 1 1 0 1  
## 2 1 0 1  
## 1 0 1 1  
## 3 0 1 0  
## 3 0 1 0  
## 1 0 0 0  
## 16 0 0 0  
## 11 1 1 1  
## 4 1 1 1  
## 1 1 1 0  
## 1 1 0 1  
## 24 25 33  
## cash\_reserve\_total revenue\_2020\_Total revenue\_2019\_Total   
## 261 1 1 1 0  
## 5 1 1 0 1  
## 1 1 0 1 1  
## 24 1 0 0 2  
## 20 0 1 1 1  
## 9 0 0 0 3  
## 7 1 1 1 1  
## 1 1 1 0 2  
## 1 1 0 0 3  
## 3 1 1 1 1  
## 1 1 0 0 3  
## 1 0 1 1 2  
## 2 0 0 0 4  
## 1 1 1 1 1  
## 3 1 1 1 2  
## 3 0 0 0 5  
## 1 0 1 1 4  
## 16 0 0 0 6  
## 11 1 1 1 2  
## 4 1 0 0 4  
## 1 1 1 1 3  
## 1 1 1 1 3  
## 52 61 66 295

At this step, we need to specify distributions for our to-be imputed variables and determine which variable we would like to leave out of the imputation prediction process. We will extract information on the predictor matrix and imputation methods to change them.

The Predictor Matrix informs us which variables are going to be used to predict a plausible value for variables (1 means a variable is used to predict another variable, 0 otherwise). Since no variable can predict itself, the intersection of one variable with itself in the matrix takes the value 0. We can manually determine if we would like to leave certain variables out of prediction. In this case, we will leave out the risk score which is based on other variables in this survey.

The *mice* package assumes a distribution for each variable and imputes missing variables according to that distribution. Hence, it is important to correctly specify each of these distributions. *mice* automatically chooses distributions for variables. If we would like to change them, we can do it by changing the methods’ characteristics.

# We run the mice code with 0 iterations   
imp <- mice(allSmalls.simple, maxit=0)

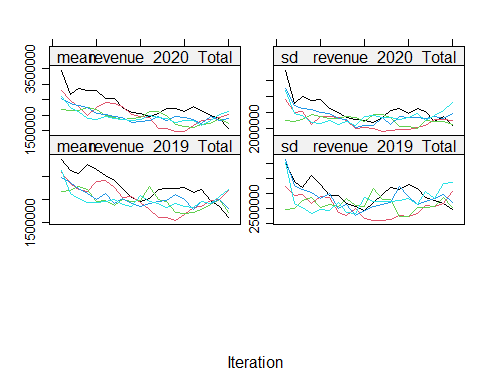
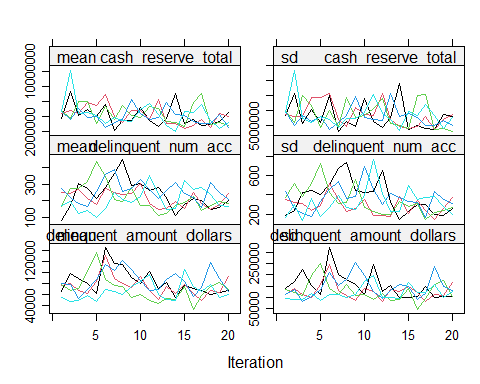
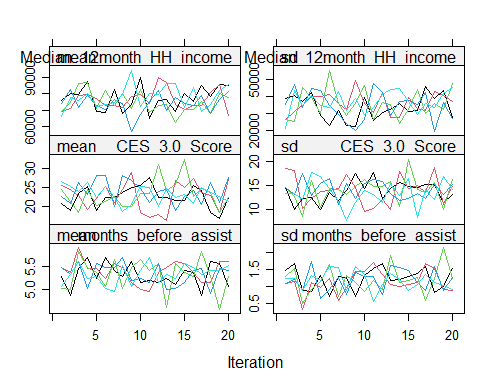
## Warning: Number of logged events: 1

# Extract predictorMatrix and methods of imputation   
predM <- imp$predictorMatrix  
meth <- imp$method  
  
# Setting values of variables I'd like to leave out to 0 in the predictor matrix  
predM[, c("PWSID")] <- 0 #name variable need not be considered  
# If you like, view the first few rows of the predictor matrix  
#head(predM)  
  
# Specify a separate imputation model for variables of interest   
# Ordered categorical variables   
poly <- c("months\_before\_assist")  
  
# Dichotomous variable  
#log <- c("")  
  
# Unordered categorical variable   
#poly2 <- c("voluntary")  
  
# Turn their methods matrix into the specified imputation models  
meth[poly] <- "polr"  
#meth[log] <- "logreg"  
#meth[poly2] <- "polyreg"  
meth

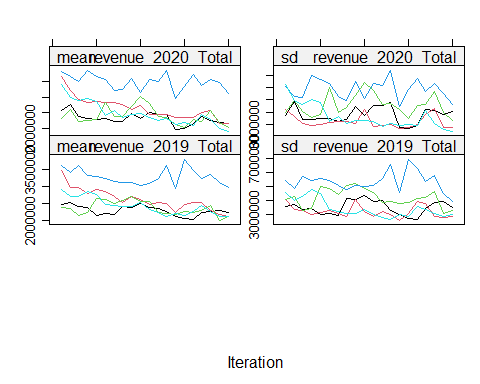
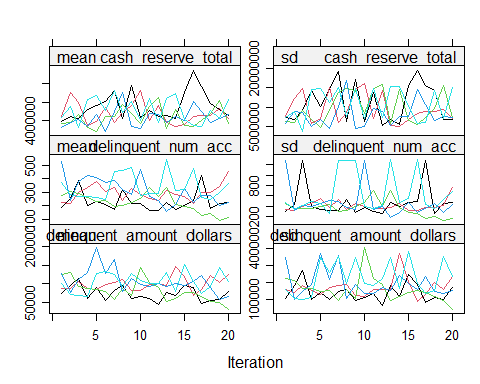
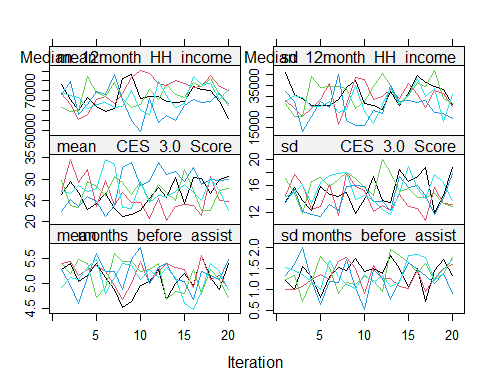
## PWSID Service\_Connections Median\_12month\_HH\_income   
## "" "" "pmm"   
## CES\_3.0\_Score months\_before\_assist cash\_reserve\_total   
## "pmm" "polr" "pmm"   
## delinquent\_num\_acc delinquent\_amount\_dollars revenue\_2020\_Total   
## "pmm" "pmm" "pmm"   
## revenue\_2019\_Total   
## "pmm"

Our variables of interest are now configured to be imputed with the imputation method we specified. Empty cells in the method matrix means that those variables aren’t going to be imputed.We are now ready for multiple imputation. This step may take a few minutes.

#There is a special function called quickpred() for a quick selection procedure of predictors, which can be handy for datasets containing many variables. selecting predictors according to data relations with a minimum correlation of ρ=.30 can be done by:  
ini <- mice(allSmalls.simple, pred=quickpred(allSmalls.simple, mincor=.3), print=F,  
 maxit = 20)  
plot(ini)

 Convergence is quite good for some variables (revenue 2020 and 2019 totals,cash reserves), but poor for others. Let’s try another imputation process:

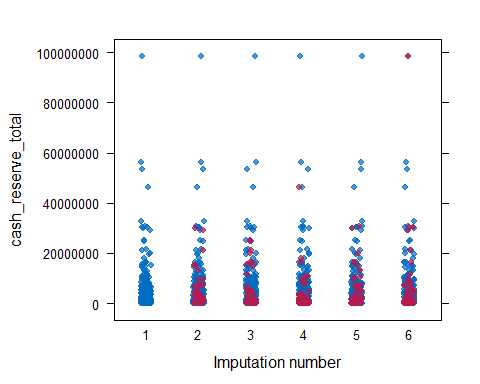
# With this command, we tell mice to impute the subset data, create 5 datasets, use predM as the predictor matrix and don't print the imputation process. If you would like to see the process, set print as TRUE  
imp2 <- mice(allSmalls.simple,   
 maxit =20, #may want to extend to 40  
 predictorMatrix = predM,   
 nnet.MaxNWts = 3000, # increase max neural network weights  
 method = "cart", #Classification and regression trees method  
 print = FALSE)  
plot(imp2)



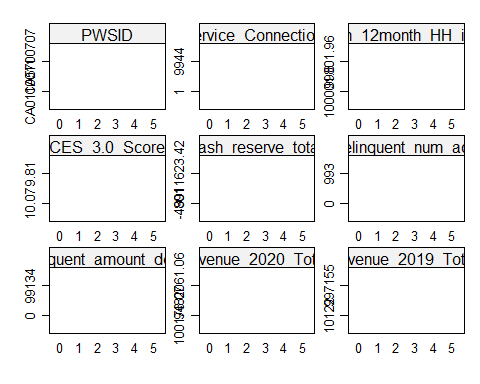
The *mice()* function implements an iterative Markov Chain Monte Carlo type of algorithm. The plots above show the mean(left) and standard deviation(right) of the imputed values. In general, we would like the streams to intermingle and be free of any trends at the later iterations.

Let’s run further diagnostics. Generally, one would prefer for the imputed data to be plausible values, i.e. values that could have been observed if they had not been missing. In order to form an idea about plausibility, one may check the imputations and compare them against the observed values. If we are willing to assume that the data are missing completely at random (MCAR), then the imputations should have the same distribution as the observed data. In general, distributions may be different because the missing data are MAR (or even MNAR). However, very large discrepancies need to be screened. Let us plot the observed and imputed data of total cash reserve.

# inspect quality of imputations  
stripplot(imp2, cash\_reserve\_total~.imp, pch = 19, xlab = "Imputation number")

 We can see above that the imputed values are indeed realistic. Let’s examine the other variables.

stripplot(imp2)



Observed data is plotted in blue, and imputed is in red. The figure graphs the data values of chl before and after imputation. Since the PMM method draws imputations from the observed data, imputed values have the same gaps as in the observed data, and are always within the range of the observed data. The figure indicates that the distributions of the imputed and the observed values are similar. The observed data have a particular feature that, for some reason, thedata cluster around the value of ???

Finally, we need to run the regression on each of the 40 datasets and pool the estimates together to get average regression coefficients and correct standard errors. The *with* function in the *mice* package allows us to do this.

#First, turn the datasets into long format  
allSmalls.simple\_long <- mice::complete(imp2, action="long", include = TRUE)  
  
# #provide integer tag for voluntary status  
# allSmalls.simple\_long %<>%   
# mutate(volunteered = case\_when(voluntary == "y" ~ 1,voluntary == "n" ~ 0))  
#provide integer tag for loan status  
  
# Convert back to mids type - mice can work with this type  
allSmalls.simple\_long\_mids <- as.mids(allSmalls.simple\_long)  
# Regression   
  
fitimp <- with(allSmalls.simple\_long\_mids,  
 lm(months\_before\_assist\_num ~ Service\_Connections + Median\_12month\_HH\_income + Median\_rent\_pct\_income + CES\_3.0\_Score + delinquent\_num\_acc + volunteered + delinquent\_amount\_dollars, na.action = na.omit, data = allSmalls.responded))  
  
summary(pool(fitimp))

## term estimate std.error statistic  
## 1 (Intercept) 4.5795100195280 0.5259296202924 8.7074579  
## 2 Service\_Connections 0.0000252272303 0.0000292580157 0.8622331  
## 3 Median\_12month\_HH\_income 0.0000066316540 0.0000028158423 2.3551226  
## 4 Median\_rent\_pct\_income 0.0192089051312 0.0122867983298 1.5633776  
## 5 CES\_3.0\_Score -0.0110661839976 0.0052732060727 -2.0985685  
## 6 delinquent\_num\_acc -0.0005409444659 0.0002575333814 -2.1004829  
## 7 volunteered -0.4053639147646 0.2362214347617 -1.7160336  
## 8 delinquent\_amount\_dollars -0.0000002359419 0.0000007066833 -0.3338722  
## df p.value  
## 1 341.9828 0.00000000  
## 2 341.9828 0.38916345  
## 3 341.9828 0.01908102  
## 4 341.9828 0.11888867  
## 5 341.9828 0.03658799  
## 6 341.9828 0.03641824  
## 7 341.9828 0.08706152  
## 8 341.9828 0.73868064

We can compare the pooled coefficients and p-values from the imputed datasets to see if any trends are altered (either become more pronounced or less). They should be similiar to the listwise-deletion technique.

Let’s look at the fmi and lambda. These should be quite low for a good model.

pool(fitimp)

## Class: mipo m = 5   
## term m estimate ubar b  
## 1 (Intercept) 5 4.5795100195280 0.2766019655008780109 0  
## 2 Service\_Connections 5 0.0000252272303 0.0000000008560314812 0  
## 3 Median\_12month\_HH\_income 5 0.0000066316540 0.0000000000079289679 0  
## 4 Median\_rent\_pct\_income 5 0.0192089051312 0.0001509654131979459 0  
## 5 CES\_3.0\_Score 5 -0.0110661839976 0.0000278067022851698 0  
## 6 delinquent\_num\_acc 5 -0.0005409444659 0.0000000663234425403 0  
## 7 volunteered 5 -0.4053639147646 0.0558005662408733727 0  
## 8 delinquent\_amount\_dollars 5 -0.0000002359419 0.0000000000004994012 0  
## t dfcom df riv lambda fmi  
## 1 0.2766019655008780109 344 341.9828 0 0 0.005797391  
## 2 0.0000000008560314812 344 341.9828 0 0 0.005797391  
## 3 0.0000000000079289679 344 341.9828 0 0 0.005797391  
## 4 0.0001509654131979459 344 341.9828 0 0 0.005797391  
## 5 0.0000278067022851698 344 341.9828 0 0 0.005797391  
## 6 0.0000000663234425403 344 341.9828 0 0 0.005797391  
## 7 0.0558005662408733727 344 341.9828 0 0 0.005797391  
## 8 0.0000000000004994012 344 341.9828 0 0 0.005797391

We can see that these values are very low. Let’s compare to the default model.

#First, turn the datasets into long format  
allSmalls.simple\_long\_1 <- mice::complete(imp, action="long", include = TRUE)  
  
# #provide integer tag for voluntary status  
# allSmalls.simple\_long %<>%   
# mutate(volunteered = case\_when(voluntary == "y" ~ 1,voluntary == "n" ~ 0))  
#provide integer tag for loan status  
  
# Convert back to mids type - mice can work with this type  
allSmalls.simple\_long\_mids\_1 <- as.mids(allSmalls.simple\_long\_1)  
# Regression   
  
fitimp\_1 <- with(allSmalls.simple\_long\_mids\_1,  
 lm(months\_before\_assist\_num ~ Service\_Connections + Median\_12month\_HH\_income + Median\_rent\_pct\_income + CES\_3.0\_Score + delinquent\_num\_acc + volunteered + delinquent\_amount\_dollars, na.action = na.omit, data = allSmalls.responded))  
  
summary(pool(fitimp\_1))

## term estimate std.error statistic  
## 1 (Intercept) 4.5795100195280 0.5259296202924 8.7074579  
## 2 Service\_Connections 0.0000252272303 0.0000292580157 0.8622331  
## 3 Median\_12month\_HH\_income 0.0000066316540 0.0000028158423 2.3551226  
## 4 Median\_rent\_pct\_income 0.0192089051312 0.0122867983298 1.5633776  
## 5 CES\_3.0\_Score -0.0110661839976 0.0052732060727 -2.0985685  
## 6 delinquent\_num\_acc -0.0005409444659 0.0002575333814 -2.1004829  
## 7 volunteered -0.4053639147646 0.2362214347617 -1.7160336  
## 8 delinquent\_amount\_dollars -0.0000002359419 0.0000007066833 -0.3338722  
## df p.value  
## 1 341.9828 0.00000000  
## 2 341.9828 0.38916345  
## 3 341.9828 0.01908102  
## 4 341.9828 0.11888867  
## 5 341.9828 0.03658799  
## 6 341.9828 0.03641824  
## 7 341.9828 0.08706152  
## 8 341.9828 0.73868064

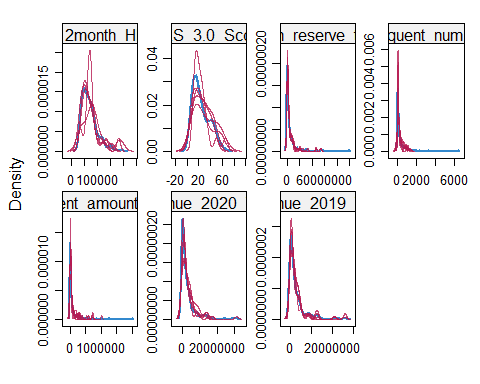
pool(fitimp\_1)

## Class: mipo m = 5   
## term m estimate ubar b  
## 1 (Intercept) 5 4.5795100195280 0.2766019655008780109 0  
## 2 Service\_Connections 5 0.0000252272303 0.0000000008560314812 0  
## 3 Median\_12month\_HH\_income 5 0.0000066316540 0.0000000000079289679 0  
## 4 Median\_rent\_pct\_income 5 0.0192089051312 0.0001509654131979459 0  
## 5 CES\_3.0\_Score 5 -0.0110661839976 0.0000278067022851698 0  
## 6 delinquent\_num\_acc 5 -0.0005409444659 0.0000000663234425403 0  
## 7 volunteered 5 -0.4053639147646 0.0558005662408733727 0  
## 8 delinquent\_amount\_dollars 5 -0.0000002359419 0.0000000000004994012 0  
## t dfcom df riv lambda fmi  
## 1 0.2766019655008780109 344 341.9828 0 0 0.005797391  
## 2 0.0000000008560314812 344 341.9828 0 0 0.005797391  
## 3 0.0000000000079289679 344 341.9828 0 0 0.005797391  
## 4 0.0001509654131979459 344 341.9828 0 0 0.005797391  
## 5 0.0000278067022851698 344 341.9828 0 0 0.005797391  
## 6 0.0000000663234425403 344 341.9828 0 0 0.005797391  
## 7 0.0558005662408733727 344 341.9828 0 0 0.005797391  
## 8 0.0000000000004994012 344 341.9828 0 0 0.005797391

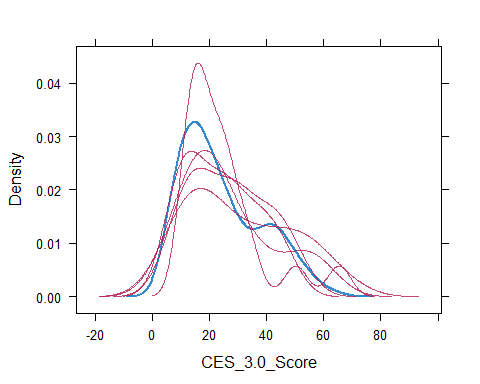
These two models are virtually the same.

Let’s further inspect the imputed values by plotting the histograms (using density plots) for the real (blue) and imputed (red) values.

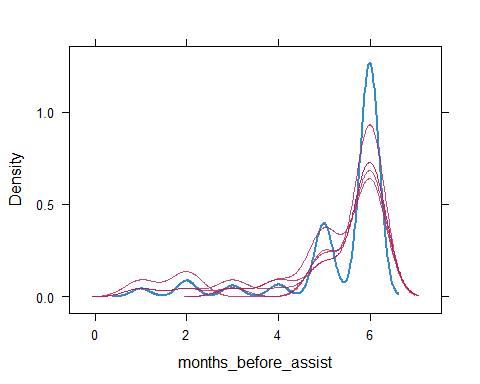
densityplot(imp)

 Many of these variables seem to be well-predicted by the mode. The worst seems to be CES 3.0 score and 12 month household income. Looking closer.

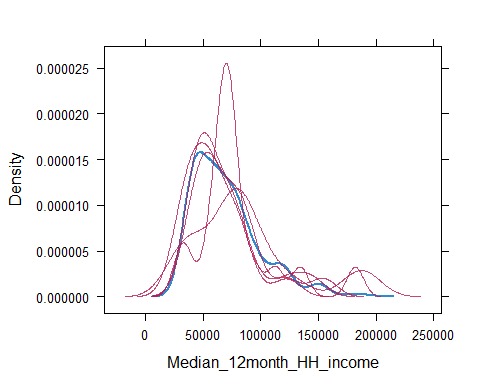
densityplot(imp, ~CES\_3.0\_Score)

 We can see here that the model biases towards lower CalEnviroScreen 3.0 scores.

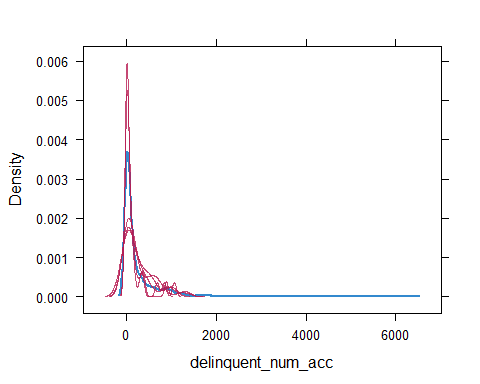
densityplot(imp, ~months\_before\_assist)

 This model seems to overestimate the number of systems that will need immediate assistance.

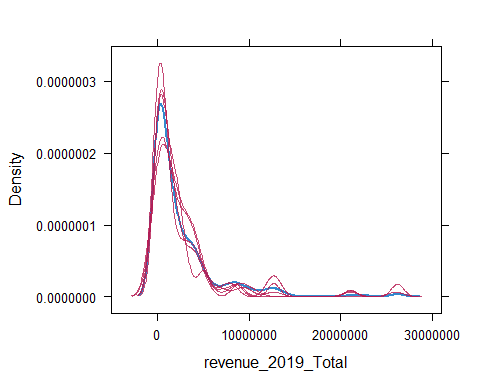
densityplot(imp, ~Median\_12month\_HH\_income)

 This model has not adequately learned income status.

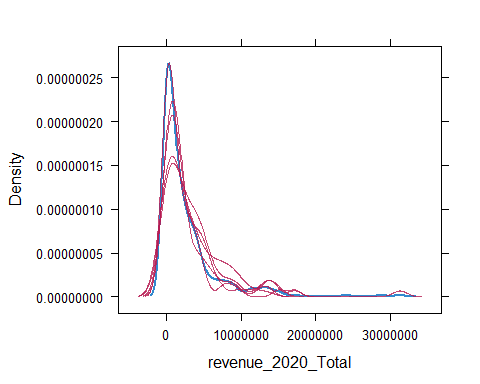
densityplot(imp, ~delinquent\_num\_acc)

 Also not great.

densityplot(imp, ~revenue\_2019\_Total)

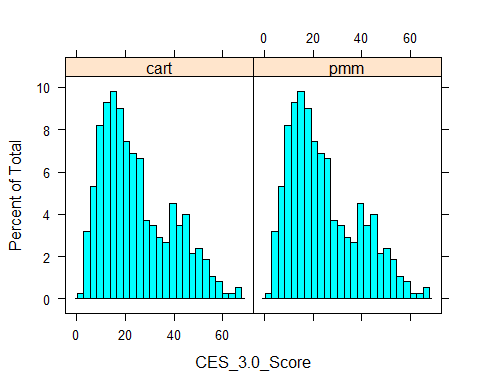
 Seems ok.

densityplot(imp, ~revenue\_2020\_Total)

 Great!

Let’s further compare the default imputation method (passive multiple imputation) with the other method employed (Classification and regression trees method) for CES 3.0 score.

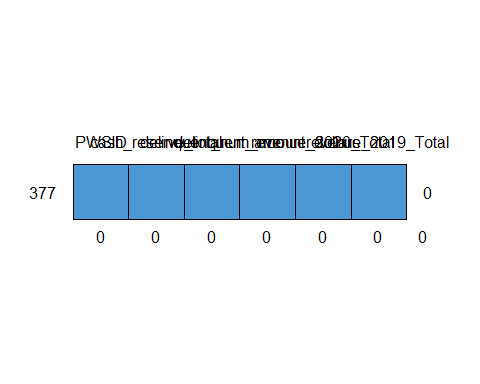
CES\_3.0\_Score <- c(complete(imp)$CES\_3.0\_Score, complete(imp)$CES\_3.0\_Score)  
method <- rep(c("pmm", "cart"), each = nrow(allSmalls.simple))  
CES\_3.0\_Score\_m <- data.frame(CES\_3.0\_Score = CES\_3.0\_Score, method = method)  
#plot histogram  
histogram( ~CES\_3.0\_Score | method, data = CES\_3.0\_Score\_m, nint = 25)

 Here we see that passive multiple imputation and the Classification and regression trees method do not seemingly differ in prediction. However, these values were still poorly predicted, so we will not use them further.

Now that we are satisfied with the selected imputed variables for dataset, let’s extract the completed data and confirm that there are no additional missing values.

#extract the third imputed data set  
simpleImputed <- complete(imp) %>% select(-CES\_3.0\_Score, -Median\_12month\_HH\_income, -months\_before\_assist, -Service\_Connections)  
#the imputed data can also be extracted in long format.  
#c.long <- complete(imp, "long")   
md.pattern(simpleImputed)

## /\ /\  
## { `---' }  
## { O O }  
## ==> V <== No need for mice. This data set is completely observed.  
## \ \|/ /  
## `-----'



## PWSID cash\_reserve\_total delinquent\_num\_acc delinquent\_amount\_dollars  
## 377 1 1 1 1  
## 0 0 0 0  
## revenue\_2020\_Total revenue\_2019\_Total   
## 377 1 1 0  
## 0 0 0

Here we can see that all data is completed.

We can also compare summary statistics for the imputed variables with those in the original dataset.

simpleImputed %>%   
 mutate(Grp = "Imputed") %>%  
 bind\_rows(mutate(allSmalls.requested.responded, Grp = "Original")) %>%  
 select(Grp, delinquent\_num\_acc, delinquent\_amount\_dollars, revenue\_2020\_Total, revenue\_2019\_Total) %>%  
 group\_by(Grp) %>%   
 drop\_na() %>%   
 summarize\_all(list(mean = "mean", median = "median"))

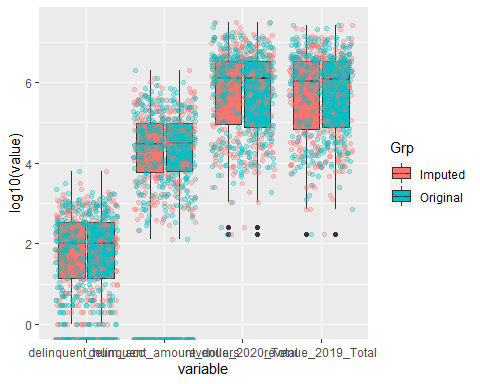
## # A tibble: 2 x 9  
## Grp delinquent\_num\_acc\_mean delinquent\_amount\_dollars\_mean  
## <chr> <dbl> <dbl>  
## 1 Imputed 241. 82064.  
## 2 Original 250. 85696.  
## revenue\_2020\_Total\_mean revenue\_2019\_Total\_mean delinquent\_num\_acc\_median  
## <dbl> <dbl> <int>  
## 1 2768490. 2432145. 58  
## 2 2649865. 2555228. 67  
## delinquent\_amount\_dollars\_median revenue\_2020\_Total\_median  
## <dbl> <dbl>  
## 1 21698. 1294782.  
## 2 23065. 1288004   
## revenue\_2019\_Total\_median  
## <dbl>  
## 1 1071950   
## 2 1187129.

Summary stats can be plotted.

#join data for plotting  
simpleImputed %>% mutate(Grp = "Imputed") %>%  
 bind\_rows(mutate(allSmalls.requested.responded, Grp = "Original")) %>%  
 select(Grp, delinquent\_num\_acc, delinquent\_amount\_dollars, revenue\_2020\_Total, revenue\_2019\_Total) %>%   
 melt() %>% #transforms values and variables to long format  
 ggplot(aes(x = variable, y = log10(value))) +  
 geom\_boxplot(aes(fill = Grp))+  
 geom\_jitter(aes(color = Grp), alpha = 0.3)

## Warning: Removed 417 rows containing non-finite values (stat\_boxplot).

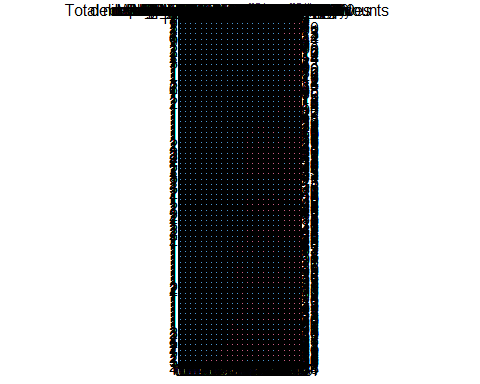
## Warning: Removed 184 rows containing missing values (geom\_point).

 We can see that imputation has not noticeably changed the summary statistics of these data. We shall now fill in the dataset with these imputed values.

#join data with imputed values to rest of dataset  
imputedSmalls <- right\_join(simpleImputed, allSmalls.rest, by = "PWSID")  
#also split the dataset that has requested but no response data, which we will only use to adjust weights in the next section  
allSmalls.requested.voluntary.rest <- allSmalls.requested.voluntary %>%   
 select(-cash\_reserve\_total, -delinquent\_num\_acc, -delinquent\_amount\_dollars, -revenue\_2020\_Total, -revenue\_2019\_Total)  
#join  
imputedSmalls.requested.voluntary <- right\_join(simpleImputed, allSmalls.requested.voluntary.rest, by = "PWSID")  
rm(allSmalls.requested.voluntary.rest)  
rm(allSmalls.rest)  
rm(imp)  
rm(imp2)  
rm(fitols)  
rm(allSmalls.simple2)

## Warning in rm(allSmalls.simple2): object 'allSmalls.simple2' not found

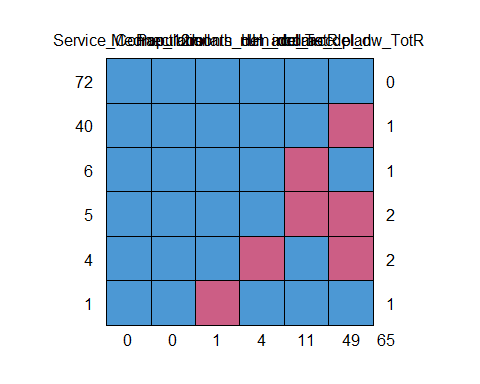
rm(allSmalls.simple\_long)  
rm(allSmalls.simple\_long\_1)  
rm(allSmalls.simple\_long\_mids)  
rm(allSmalls.simple\_long\_mids\_1)  
rm(fitimp)  
rm(fitimp\_1)  
rm(ini)  
rm(CES\_3.0\_Score\_m)  
#visualize remaining missing values  
md.pattern(imputedSmalls)

 Note that some variables are still missing, however these were deliberately not imputed either due to no being applicable (such as binned data) or would were calculated from other variables.

## Larges Imputation

The above steps will be generally repeated for large systems (>10,000 service connections) surveyed. Prior to determining if multiple imputation is necessary (as opposed to row-wise deletion or mean imputation), primary auxiliary and response variables will be examined for completeness in the matrix below.

larges <- allLarges %>% filter(responded == "y")  
#extract the meaningful parameters  
meaningfulVariables <- complete(larges) %>% select(dollars\_del\_acc\_TotR, dollars\_del\_dw\_TotR, Median\_12month\_HH\_income, Service\_Connections, Population, num\_del\_acc\_plan)  
#the imputed data can also be extracted in long format.  
#c.long <- complete(imp, "long")   
md.pattern(meaningfulVariables)



## Service\_Connections Population Median\_12month\_HH\_income dollars\_del\_acc\_TotR  
## 72 1 1 1 1  
## 40 1 1 1 1  
## 6 1 1 1 1  
## 5 1 1 1 1  
## 4 1 1 1 0  
## 1 1 1 0 1  
## 0 0 1 4  
## num\_del\_acc\_plan dollars\_del\_dw\_TotR   
## 72 1 1 0  
## 40 1 0 1  
## 6 0 1 1  
## 5 0 0 2  
## 4 1 0 2  
## 1 1 1 1  
## 11 49 65

The matrix above shows relatively few responses are missing for most variables. These can be viewed in tabular format as well.

#extract the meaningful parameters  
larges %>%   
 select(dollars\_del\_acc\_TotR, dollars\_del\_dw\_TotR, Median\_12month\_HH\_income, Service\_Connections, Population, num\_del\_acc\_plan, Fee\_Code) %>%   
 summarise\_all(funs(sum(is.na(.)))) %>% #summarize NA  
 t() #transpose

## Warning: `funs()` is deprecated as of dplyr 0.8.0.  
## Please use a list of either functions or lambdas:   
##   
## # Simple named list:   
## list(mean = mean, median = median)  
##   
## # Auto named with `tibble::lst()`:   
## tibble::lst(mean, median)  
##   
## # Using lambdas  
## list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_warnings()` to see where this warning was generated.

## [,1]  
## dollars\_del\_acc\_TotR 4  
## dollars\_del\_dw\_TotR 49  
## Median\_12month\_HH\_income 1  
## Service\_Connections 0  
## Population 0  
## num\_del\_acc\_plan 11  
## Fee\_Code 0

Luckily, auxiliary variables (service connections, population, fee code) are all complete. One system lacks median-12-month income data. Vital response variables are less complete with total delinquent dollars (drinking water) the most incomplete (49 out of 128 cases missing). Multiple imputation will be employed to fill in missing values using predictive statistics.

An Ordinary Least Squares (OLS) linear regression is used to predict total delinquent dollars (i.e. debt) based on reasonable predictors that are available for all systems (e.g. service connections, population, median 12 month household income, median rent percent income, fee code, number of delinquent account plans).

## Estimate an OLS Regression  
fitols <- lm(dollars\_del\_dw\_TotR ~ Service\_Connections + Median\_12month\_HH\_income + Fee\_Code + Population +  
 Median\_rent\_pct\_income + num\_del\_acc\_plan + dollars\_del\_acc\_TotR,  
 na.action = na.omit,   
 data = larges)  
summary(fitols)

##   
## Call:  
## lm(formula = dollars\_del\_dw\_TotR ~ Service\_Connections + Median\_12month\_HH\_income +   
## Fee\_Code + Population + Median\_rent\_pct\_income + num\_del\_acc\_plan +   
## dollars\_del\_acc\_TotR, data = larges, na.action = na.omit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -4848986 -245399 108510 326119 9057736   
##   
## Coefficients:  
## Estimate Std. Error t value  
## (Intercept) -1633812.36062 2693295.14129 -0.607  
## Service\_Connections -8.95018 18.21092 -0.491  
## Median\_12month\_HH\_income 0.13354 8.35601 0.016  
## Fee\_CodeDAVCL 89148.36909 819453.23399 0.109  
## Population 10.84986 3.93210 2.759  
## Median\_rent\_pct\_income 35188.17623 67117.27586 0.524  
## num\_del\_acc\_plan -148.94652 70.50575 -2.113  
## dollars\_del\_acc\_TotR 0.09934 0.01118 8.884  
## Pr(>|t|)   
## (Intercept) 0.54625   
## Service\_Connections 0.62477   
## Median\_12month\_HH\_income 0.98730   
## Fee\_CodeDAVCL 0.91371   
## Population 0.00755 \*\*   
## Median\_rent\_pct\_income 0.60190   
## num\_del\_acc\_plan 0.03855 \*   
## dollars\_del\_acc\_TotR 0.000000000000903 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 1504000 on 64 degrees of freedom  
## (56 observations deleted due to missingness)  
## Multiple R-squared: 0.9784, Adjusted R-squared: 0.976   
## F-statistic: 414.3 on 7 and 64 DF, p-value: < 0.00000000000000022

As we can see in the table above, a significant number of observations were deleted due to missingness (56 out of 128). Since this is greater than 43% of the whole dataset, our survey would be limited in power. However, as seen in the above matrix, auxiliary variables are complete, and the OLS demonstrates a strong predictive power between population and delinquent drinking water dollars (*p* < 0.01). This is intuitive due to economies of scale. Further, since delinquent dollars (total) are more available, this response variable can be used to predict debt for drinking water (*p*<0.00001). Imputation of these values will decrease variance, but potentially increase the uncertainty of this survey (albeit slightly). The relationship between these two variables is demonstrated in a linear regression.

## Estimate an OLS Regression  
summary(lm(dollars\_del\_dw\_TotR ~ dollars\_del\_acc\_TotR,  
 na.action = na.omit,   
 data = larges))

##   
## Call:  
## lm(formula = dollars\_del\_dw\_TotR ~ dollars\_del\_acc\_TotR, data = larges,   
## na.action = na.omit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -1653363 -684319 -490164 -67353 17576738   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 782198.824780 248966.417918 3.142 0.00238  
## dollars\_del\_acc\_TotR 0.171066 0.004695 36.432 < 0.0000000000000002  
##   
## (Intercept) \*\*   
## dollars\_del\_acc\_TotR \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 2188000 on 77 degrees of freedom  
## (49 observations deleted due to missingness)  
## Multiple R-squared: 0.9452, Adjusted R-squared: 0.9445   
## F-statistic: 1327 on 1 and 77 DF, p-value: < 0.00000000000000022

This relationship is plotted below.

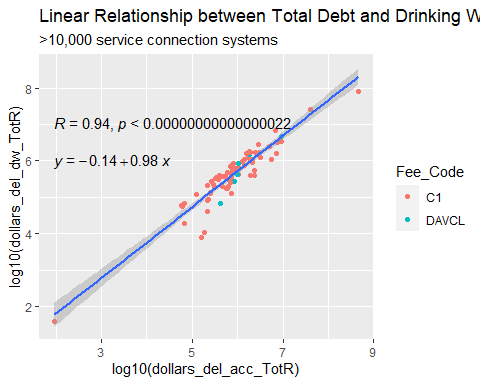
#define linear relationship  
larges %>%   
 ggplot(aes(x = log10(dollars\_del\_acc\_TotR), y = log10(dollars\_del\_dw\_TotR))) +  
 geom\_point(aes(color = Fee\_Code)) +  
 geom\_smooth(method = "lm") +  
 stat\_cor(label.y = 7)+ #this means at 35th unit in the y axis, the r squared and p value will be shown  
 stat\_regline\_equation(label.y = 6) + #this means at 30th unit regresion line equation will be shown  
 labs(title = "Linear Relationship between Total Debt and Drinking Water Debt",  
 subtitle = ">10,000 service connection systems")

## Warning: Removed 49 rows containing non-finite values (stat\_smooth).

## Warning: Removed 49 rows containing non-finite values (stat\_cor).

## Warning: Removed 49 rows containing non-finite values (stat\_regline\_equation).

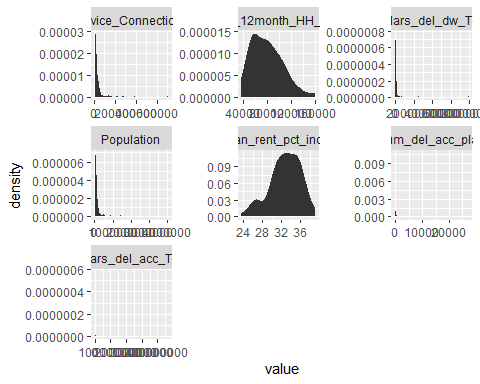
## Warning: Removed 49 rows containing missing values (geom\_point).



Prior to imputation, the dataset should be prepared such that variables remain in their rawest form, so any categorical factors should be left as so, instead of using the ordinal transformed variable from above. Further, we want to remove variables that have haver than 25% missing values because they may mess up the imputation (no such variables). It’s also important to remove variables that are highly correlated with others so as to stop the imputation working otherwise. Additionally, any extreme outliers should be removed, as they may dramatically impact results.

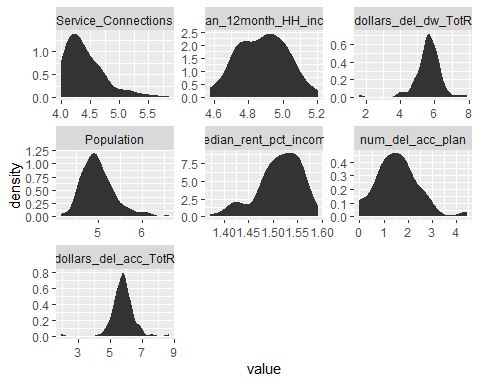
require(reshape2)  
larges %>%   
 select(PWSID, Service\_Connections, Median\_12month\_HH\_income, dollars\_del\_dw\_TotR, Fee\_Code, Population,  
 Median\_rent\_pct\_income,num\_del\_acc\_plan, dollars\_del\_acc\_TotR) %>%   
 melt() %>% #convert wide to long  
 ggplot(aes(x = value)) +   
 stat\_density() +   
 facet\_wrap(~variable, scales = "free")

## Warning: Removed 66 rows containing non-finite values (stat\_density).

 Everything looks more or less realistic, however several variables would benefit from log10 transformation for visualizing and possibly for statistics.

require(reshape2)  
larges %>%   
 select(PWSID, Service\_Connections, Median\_12month\_HH\_income, dollars\_del\_dw\_TotR, Fee\_Code, Population,  
 Median\_rent\_pct\_income,num\_del\_acc\_plan, dollars\_del\_acc\_TotR) %>%   
 melt() %>% #convert wide to long  
 mutate\_if(~is.numeric(.) && (.) > 0, log10) %>%   
 ggplot(aes(x = value)) +   
 stat\_density() +   
 facet\_wrap(~variable, scales = "free")

## Warning: Removed 104 rows containing non-finite values (stat\_density).

 Now distributions look more or less “normal.”

#log transform  
larges %<>% mutate(log.dollars\_del\_dw\_TotR = log10(dollars\_del\_dw\_TotR),  
 log.dollars\_del\_acc\_TotR = log10(dollars\_del\_acc\_TotR))  
  
#define sub dataset for imputation  
larges.simple <- larges %>%   
 select(PWSID, log.dollars\_del\_acc\_TotR, log.dollars\_del\_dw\_TotR, Median\_12month\_HH\_income, Service\_Connections, num\_del\_acc\_plan, Median\_rent\_pct\_income)  
  
# #split the other half of the dataset for easy joining  
# larges.rest <- larges %>%   
# select(-c(log.dollars\_del\_acc\_TotR, log.dollars\_del\_dw\_TotR, Median\_12month\_HH\_income, Service\_Connections, num\_del\_acc\_plan, Median\_rent\_pct\_income))

At this step, we need to specify distributions for our to-be imputed variables and determine which variable we would like to leave out of the imputation prediction process. We will extract information on the predictor matrix and imputation methods to change them.

The Predictor Matrix informs us which variables are going to be used to predict a plausible value for variables (1 means a variable is used to predict another variable, 0 otherwise). Since no variable can predict itself, the intersection of one variable with itself in the matrix takes the value 0. We can manually determine if we would like to leave certain variables out of prediction. In this case, we will leave out the risk score which is based on other variables in this survey.

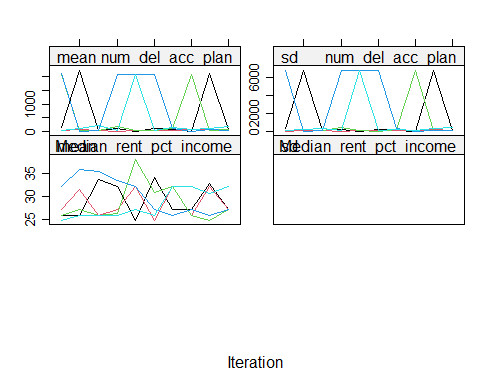
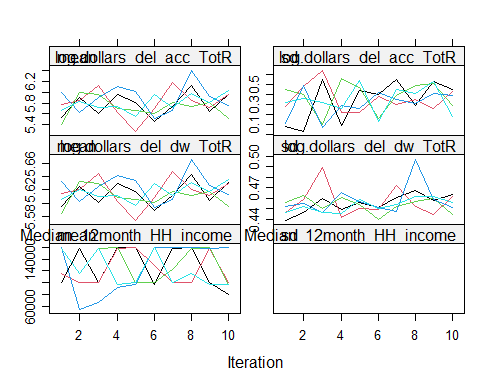
The *mice* package assumes a distribution for each variable and imputes missing variables according to that distribution. Hence, it is important to correctly specify each of these distributions. *mice* automatically chooses distributions for variables. If we would like to change them, we can do it by changing the methods’ characteristics.

Based on the strong linear relationship between significant vartiables a simple predictive mean matching model will be used, with a manual correction for one-way prediction between total debt and drinking water debt, to avoid circularity.

ini <- mice(larges.simple, maxit=0, print=F)

## Warning: Number of logged events: 1

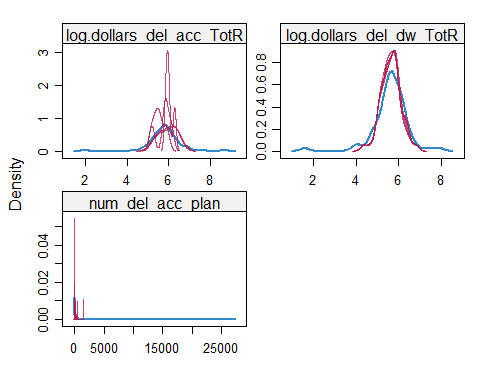
meth<- ini$meth  
pred <- ini$pred  
pred[c("log.dollars\_del\_acc\_TotR"), "log.dollars\_del\_dw\_TotR"] <- 0 #exclude drinking water debt as a predictor of total debt to avoid circularity  
meth["log.dollars\_del\_dw\_TotR"]<- "~ I(log.dollars\_del\_acc\_TotR)" #define predictor for drinking water debt  
passive.imp <- mice(larges.simple, meth=meth, pred=pred, maxit=10, seed=123, print=F)  
plot(passive.imp)



The *mice()* function implements an iterative Markov Chain Monte Carlo type of algorithm. The plots above show the mean(left) and standard deviation(right) of the imputed values. In general, we would like the streams to intermingle and be free of any trends at the later iterations.

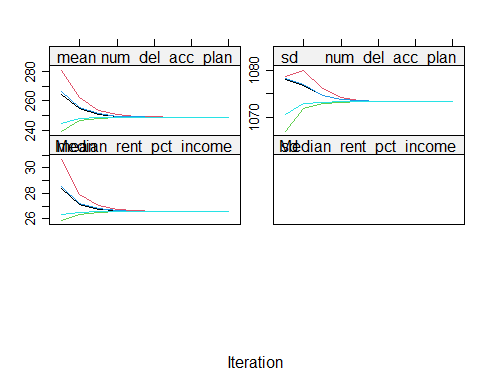
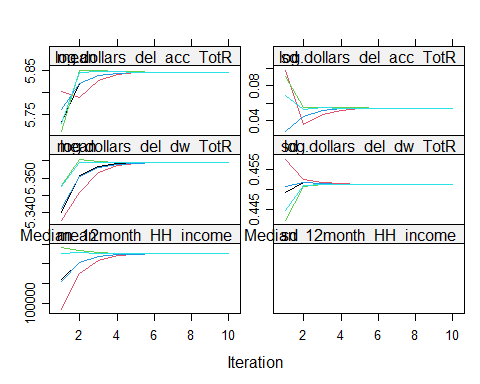
Let’s run further diagnostics. Generally, one would prefer for the imputed data to be plausible values, i.e. values that could have been observed if they had not been missing. In order to form an idea about plausibility, one may check the imputations and compare them against the observed values. If we are willing to assume that the data are missing completely at random (MCAR), then the imputations should have the same distribution as the observed data. In general, distributions may be different because the missing data are MAR (or even MNAR). However, very large discrepancies need to be screened. Observed and imputed data are plotted below.

densityplot(passive.imp)

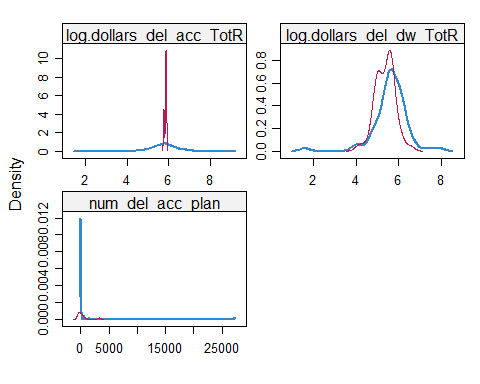
 This method was accurate for imputed deqlinquent drinking water debt, but was very poor for other parameters. A mixed model may perform better.

Bayesian linear regression will be tested as an imputation method.

normpredict.imp <- mice(larges.simple, meth="norm.predict", pred=pred, maxit=10, seed=123, print=F)  
plot(normpredict.imp)

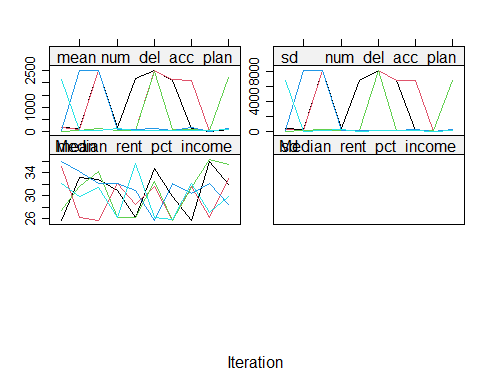
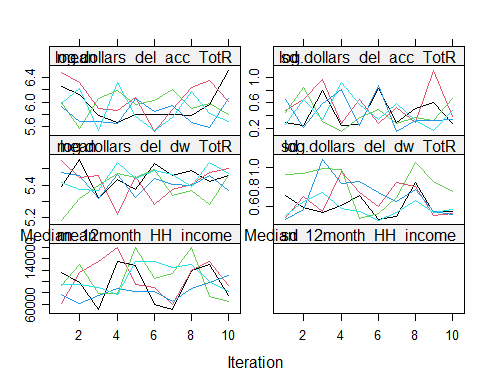
 Bayesian linear regression causes extreme convergence, as the model is quite simple.

densityplot(normpredict.imp)#, ~log.dollars\_del\_dw\_TotR)

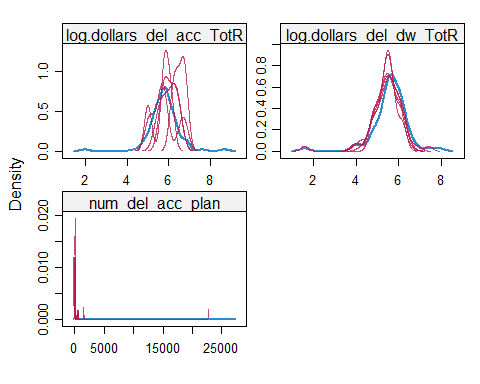


Bayesian linear regression performs quite poorly for predicting total debt. A random forest model may perform better.

rf.imp <- mice(larges.simple, meth="rf", pred=pred, maxit=10, seed=123, print=F)  
plot(rf.imp)

 Bayesian linear regression causes extreme convergence, as the model is quite simple.

densityplot(rf.imp)#, ~log.dollars\_del\_dw\_TotR)

 Random forest modeling seems to have improved the prediction of drinking water debt, and seems to have smoothed.

Observed data is plotted in blue, and imputed is in red. The figure graphs the data values of chl before and after imputation. The figure indicates that the distributions of the imputed and the observed values are similar. Finally, we need to run the regression on each of the 40 datasets and pool the estimates together to get average regression coefficients and correct standard errors. The *with* function in the *mice* package allows us to do this.

#First, turn the datasets into long format  
larges.simple\_long <- mice::complete(rf.imp, action="long", include = TRUE)  
  
# #provide integer tag for voluntary status  
# allSmalls.simple\_long %<>%   
# mutate(volunteered = case\_when(voluntary == "y" ~ 1,voluntary == "n" ~ 0))  
#provide integer tag for loan status  
  
# Convert back to mids type - mice can work with this type  
larges.simple\_long\_mids <- as.mids(larges.simple\_long)  
# Regression   
  
fitimp <- with(larges.simple\_long\_mids,  
 lm(log.dollars\_del\_dw\_TotR ~ Service\_Connections + Median\_12month\_HH\_income + Fee\_Code + Population +  
 Median\_rent\_pct\_income + num\_del\_acc\_plan + log.dollars\_del\_acc\_TotR,  
 na.action = na.omit, data = larges))  
  
summary(pool(fitimp))

## term estimate std.error statistic df  
## 1 (Intercept) -1.3211176945632 0.7050537057361 -1.8737831 62.08333  
## 2 Service\_Connections 0.0000034907419 0.0000026898084 1.2977660 62.08333  
## 3 Median\_12month\_HH\_income 0.0000007264613 0.0000015582354 0.4662077 62.08333  
## 4 Fee\_CodeDAVCL -0.1412559248547 0.1476847248049 -0.9564694 62.08333  
## 5 Population -0.0000006305195 0.0000004692327 -1.3437245 62.08333  
## 6 Median\_rent\_pct\_income 0.0329663779491 0.0120882211496 2.7271488 62.08333  
## 7 num\_del\_acc\_plan -0.0000125580545 0.0000119151414 -1.0539577 62.08333  
## 8 log.dollars\_del\_acc\_TotR 0.9809491774576 0.0855363802769 11.4682101 62.08333  
## p.value  
## 1 0.065668450  
## 2 0.199168939  
## 3 0.642698532  
## 4 0.342544914  
## 5 0.183928100  
## 6 0.008294736  
## 7 0.295986947  
## 8 0.000000000

We can compare the pooled coefficients and p-values from the imputed datasets to see if any trends are altered (either become more pronounced or less). They should be similiar to the listwise-deletion technique.

Let’s look at the fmi and lambda. These should be quite low for a good model.

pool(fitimp)

## Class: mipo m = 5   
## term m estimate ubar b  
## 1 (Intercept) 5 -1.3211176945632 0.4971007279722560179 0  
## 2 Service\_Connections 5 0.0000034907419 0.0000000000072350692 0  
## 3 Median\_12month\_HH\_income 5 0.0000007264613 0.0000000000024280976 0  
## 4 Fee\_CodeDAVCL 5 -0.1412559248547 0.0218107779407125876 0  
## 5 Population 5 -0.0000006305195 0.0000000000002201793 0  
## 6 Median\_rent\_pct\_income 5 0.0329663779491 0.0001461250905609044 0  
## 7 num\_del\_acc\_plan 5 -0.0000125580545 0.0000000001419705942 0  
## 8 log.dollars\_del\_acc\_TotR 5 0.9809491774576 0.0073164723508763133 0  
## t dfcom df riv lambda fmi  
## 1 0.4971007279722560179 64 62.08333 0 0 0.03072983  
## 2 0.0000000000072350692 64 62.08333 0 0 0.03072983  
## 3 0.0000000000024280976 64 62.08333 0 0 0.03072983  
## 4 0.0218107779407125876 64 62.08333 0 0 0.03072983  
## 5 0.0000000000002201793 64 62.08333 0 0 0.03072983  
## 6 0.0001461250905609044 64 62.08333 0 0 0.03072983  
## 7 0.0000000001419705942 64 62.08333 0 0 0.03072983  
## 8 0.0073164723508763133 64 62.08333 0 0 0.03072983

We can see that these values are very low. Let’s compare to the default model.

#First, turn the datasets into long format  
larges.simple\_long\_1 <- mice::complete(normpredict.imp, action="long", include = TRUE)  
  
# #provide integer tag for voluntary status  
# allSmalls.simple\_long %<>%   
# mutate(volunteered = case\_when(voluntary == "y" ~ 1,voluntary == "n" ~ 0))  
#provide integer tag for loan status  
  
# Convert back to mids type - mice can work with this type  
larges.simple\_long\_mids\_1 <- as.mids(larges.simple\_long\_1)  
# Regression   
  
fitimp\_1 <- with(larges.simple\_long\_mids\_1,  
 lm(log.dollars\_del\_dw\_TotR ~ Service\_Connections + Median\_12month\_HH\_income + Fee\_Code + Population +  
 Median\_rent\_pct\_income + num\_del\_acc\_plan + log.dollars\_del\_acc\_TotR,  
 na.action = na.omit, data = larges))  
  
summary(pool(fitimp\_1))

## term estimate std.error statistic df  
## 1 (Intercept) -1.3211176945632 0.7050537057361 -1.8737831 62.08333  
## 2 Service\_Connections 0.0000034907419 0.0000026898084 1.2977660 62.08333  
## 3 Median\_12month\_HH\_income 0.0000007264613 0.0000015582354 0.4662077 62.08333  
## 4 Fee\_CodeDAVCL -0.1412559248547 0.1476847248049 -0.9564694 62.08333  
## 5 Population -0.0000006305195 0.0000004692327 -1.3437245 62.08333  
## 6 Median\_rent\_pct\_income 0.0329663779491 0.0120882211496 2.7271488 62.08333  
## 7 num\_del\_acc\_plan -0.0000125580545 0.0000119151414 -1.0539577 62.08333  
## 8 log.dollars\_del\_acc\_TotR 0.9809491774576 0.0855363802769 11.4682101 62.08333  
## p.value  
## 1 0.065668450  
## 2 0.199168939  
## 3 0.642698532  
## 4 0.342544914  
## 5 0.183928100  
## 6 0.008294736  
## 7 0.295986947  
## 8 0.000000000

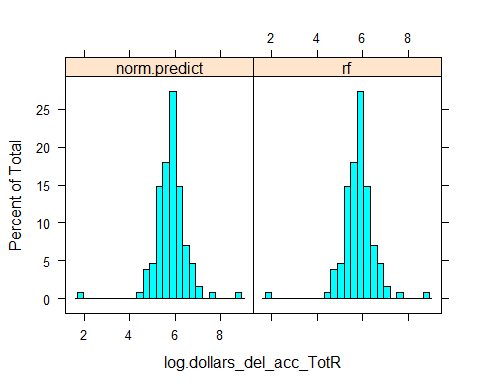
pool(fitimp\_1)

## Class: mipo m = 5   
## term m estimate ubar b  
## 1 (Intercept) 5 -1.3211176945632 0.4971007279722560179 0  
## 2 Service\_Connections 5 0.0000034907419 0.0000000000072350692 0  
## 3 Median\_12month\_HH\_income 5 0.0000007264613 0.0000000000024280976 0  
## 4 Fee\_CodeDAVCL 5 -0.1412559248547 0.0218107779407125876 0  
## 5 Population 5 -0.0000006305195 0.0000000000002201793 0  
## 6 Median\_rent\_pct\_income 5 0.0329663779491 0.0001461250905609044 0  
## 7 num\_del\_acc\_plan 5 -0.0000125580545 0.0000000001419705942 0  
## 8 log.dollars\_del\_acc\_TotR 5 0.9809491774576 0.0073164723508763133 0  
## t dfcom df riv lambda fmi  
## 1 0.4971007279722560179 64 62.08333 0 0 0.03072983  
## 2 0.0000000000072350692 64 62.08333 0 0 0.03072983  
## 3 0.0000000000024280976 64 62.08333 0 0 0.03072983  
## 4 0.0218107779407125876 64 62.08333 0 0 0.03072983  
## 5 0.0000000000002201793 64 62.08333 0 0 0.03072983  
## 6 0.0001461250905609044 64 62.08333 0 0 0.03072983  
## 7 0.0000000001419705942 64 62.08333 0 0 0.03072983  
## 8 0.0073164723508763133 64 62.08333 0 0 0.03072983

These two models differ dramatically, with the Bayesian linear regression having significantly lower variance due to nonrsesponse (lambdi) and fraction of information missing due to nonresponse (fmi), with a strong linear relationship retained between total delinquent debt and drinking water debt.

Let’s further compare the Bayesian linear regression and random forest methods for log.dollars\_del\_acc\_TotR.

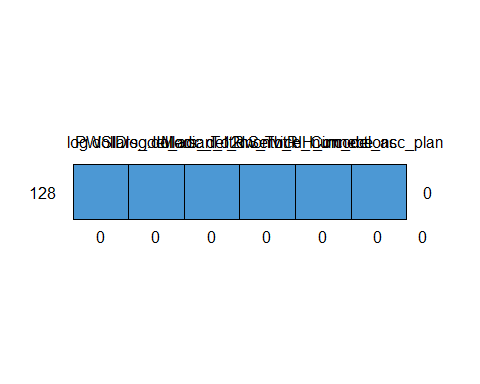
log.dollars\_del\_acc\_TotR <- c(complete(passive.imp)$log.dollars\_del\_acc\_TotR, complete(passive.imp)$log.dollars\_del\_acc\_TotR)  
method <- rep(c("norm.predict", "rf"), each = nrow(larges.simple))  
log.dollars\_del\_acc\_TotR\_m <- data.frame(log.dollars\_del\_acc\_TotR = log.dollars\_del\_acc\_TotR, method = method)  
#plot histogram  
histogram( ~log.dollars\_del\_acc\_TotR | method, data = log.dollars\_del\_acc\_TotR\_m, nint = 25)

 Here we see that these two method do not seemingly differ in prediction. Since the random forest performed better for the total water debt, and performed approximately the same for drinking water debt, we will use the random forest.

Now that we are satisfied with the selected imputed variables for dataset, let’s extract the completed data and confirm that there are no additional missing values.

#extract the meaningful parameters  
larges.simple.imputed <- complete(rf.imp) %>% select(PWSID, log.dollars\_del\_acc\_TotR, log.dollars\_del\_dw\_TotR, Median\_12month\_HH\_income, Service\_Connections, num\_del\_acc\_plan)  
#the imputed data can also be extracted in long format.  
#c.long <- complete(imp, "long")   
md.pattern(larges.simple.imputed)

## /\ /\  
## { `---' }  
## { O O }  
## ==> V <== No need for mice. This data set is completely observed.  
## \ \|/ /  
## `-----'



## PWSID log.dollars\_del\_acc\_TotR log.dollars\_del\_dw\_TotR  
## 128 1 1 1  
## 0 0 0  
## Median\_12month\_HH\_income Service\_Connections num\_del\_acc\_plan   
## 128 1 1 1 0  
## 0 0 0 0

Here we can see that all data is completed.

We can also compare summary statistics for the imputed variables with those in the original dataset.

larges.simple.imputed %>%   
 mutate(Grp = "Imputed") %>%  
 bind\_rows(mutate(larges, Grp = "Original")) %>%  
 select(Grp, log.dollars\_del\_acc\_TotR, log.dollars\_del\_dw\_TotR,   
 Median\_12month\_HH\_income, Service\_Connections, num\_del\_acc\_plan) %>%  
 group\_by(Grp) %>%   
 drop\_na() %>%   
 summarize\_all(list(mean = "mean", median = "median", sd = "sd"))

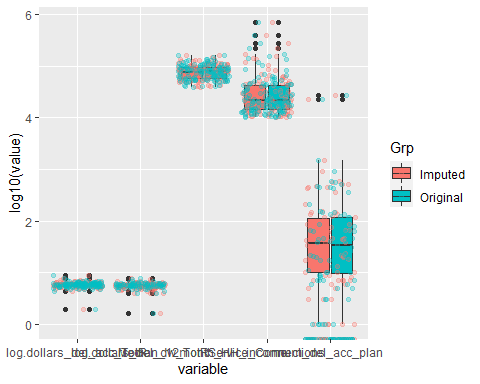
## # A tibble: 2 x 16  
## Grp log.dollars\_del\_acc\_TotR\_mean log.dollars\_del\_dw\_TotR\_mean  
## <chr> <dbl> <dbl>  
## 1 Imputed 5.82 5.56  
## 2 Original 5.93 5.66  
## Median\_12month\_HH\_income\_mean Service\_Connections\_mean num\_del\_acc\_plan\_mean  
## <dbl> <dbl> <dbl>  
## 1 80258. 43117. 468.  
## 2 82011. 51509. 797.  
## log.dollars\_del\_acc\_TotR\_median log.dollars\_del\_dw\_TotR\_median  
## <dbl> <dbl>  
## 1 5.83 5.60  
## 2 5.88 5.64  
## Median\_12month\_HH\_income\_median Service\_Connections\_median  
## <dbl> <dbl>  
## 1 77505. 21384  
## 2 77531. 24093  
## num\_del\_acc\_plan\_median log.dollars\_del\_acc\_TotR\_sd log.dollars\_del\_dw\_TotR\_sd  
## <dbl> <dbl> <dbl>  
## 1 10 0.685 0.712  
## 2 12 0.623 0.665  
## Median\_12month\_HH\_income\_sd Service\_Connections\_sd num\_del\_acc\_plan\_sd  
## <dbl> <dbl> <dbl>  
## 1 26086. 78063. 3122.  
## 2 27506. 92506. 4144.

Summary stats are not significantly altered. The mean and median number of service connections and number of deliquent account plans are beyond 10% different. Since service connections is an auxiliary variable this will be dropped from the imputed dataset. Imputation was not optimized for count data (number of delinquent accounts), and will not be used. Summary stats can be plotted.

require(reshape2)  
#join data for plotting  
larges.simple.imputed %>%   
 mutate(Grp = "Imputed") %>%  
 bind\_rows(mutate(larges, Grp = "Original")) %>%  
 select(Grp, log.dollars\_del\_acc\_TotR, log.dollars\_del\_dw\_TotR,   
 Median\_12month\_HH\_income, Service\_Connections, num\_del\_acc\_plan) %>%  
 melt() %>% #transforms values and variables to long format  
 ggplot(aes(x = variable, y = log10(value))) +  
 geom\_boxplot(aes(fill = Grp))+  
 geom\_jitter(aes(color = Grp), alpha = 0.3)

## Warning: Removed 145 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 65 rows containing missing values (geom\_point).

 We can see that imputation has not noticeably changed the summary statistics of these data. We shall now fill in the dataset with these imputed values.

#join data with imputed values to rest of dataset, providing imputed tag.  
larges.imputed.subset <- larges.simple.imputed %>%   
 mutate(dollars\_del\_acc\_TotR.imputed = 10 ^ log.dollars\_del\_acc\_TotR,  
 dollars\_del\_dw\_TotR.imputed = 10 ^ log.dollars\_del\_dw\_TotR,  
 Median\_12month\_HH\_income.imputed = Median\_12month\_HH\_income) %>%   
 select(PWSID, dollars\_del\_acc\_TotR.imputed, dollars\_del\_dw\_TotR.imputed, Median\_12month\_HH\_income.imputed)  
  
allLarges <- right\_join(larges.imputed.subset, allLarges, by = "PWSID")#join  
  
rm(larges.imputed.subset)  
rm(larges.simple.imputed)  
rm(larges.simple\_long)  
rm(larges.simple\_long\_1)  
rm(allSmalls.rest)

## Warning in rm(allSmalls.rest): object 'allSmalls.rest' not found

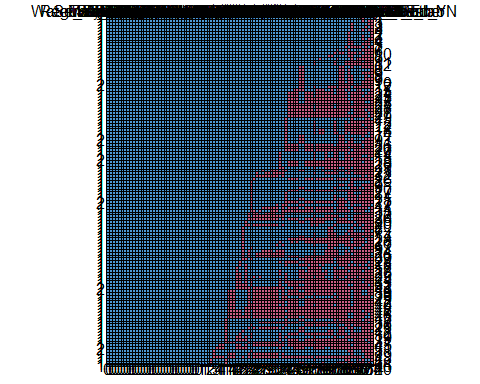
rm(imp)

## Warning in rm(imp): object 'imp' not found

rm(imp2)

## Warning in rm(imp2): object 'imp2' not found

rm(fitols)  
rm(fitimp)  
rm(fitimp\_1)  
rm(ini)  
  
#visualize remaining missing values  
md.pattern(larges)

 Note that some variables are still missing, however these were deliberately not imputed either due to no being applicable (such as binned data) or would were calculated from other variables.

# Survey Weighting

Survey weights are a key component to producing population estimates. As described in Valliant et al. 2018, an estimated total has the form where is a response provided by the *i*th sample member and is the corresponding analysis weight. Without the use of weights, estimates from survey data may simply reflecy nuances of a particular sample, containing significant levels of bias. The series of weighting that will be applied to this survey data include computation of base weights, nonresponse adjustments, and use of auxiliary data to reduce variances (i.e. calibration).

## Base Weights

The first basic step to make reliable extrapolations to the population is to generate base weights. Note that this is one of three weight adjustments that will be applied,including non-response and calibration adjustments to weights.

The first step in weighing is taking into account the different probabilities of being sampled that respondents may have simply based on proportions in the population. This is also known as generating *base weights*. Note that this survey is a random stratified sampling design without replacement. We will calculate inclusion probabilities for each strata for the small systems (tags A, B, C, D). Large systems (>10,000 service connections) will be treated as a single strata for the calculation of base weights, and will be handled separated.

### Small Systems

probSumm <- allSmalls %>%   
 filter(voluntary != "y") %>% #remove volunteers  
 mutate(sample = case\_when(requested == "y" ~ 1,requested == "n" ~ 0)) %>% #provide integer tag for sampled  
 group\_by(tag) %>%   
 summarize(N = n(),samples = sum(sample), respondedTotal = sum(response, na.rm = TRUE), prob = samples/N, base.weight = 1/prob,   
 sum.base.weight = base.weight \* samples, fpc = samples/N) %>%   
 drop\_na  
#join the probability to each sample  
allSmalls <- right\_join(allSmalls, probSumm, by = "tag")  
allSmalls.requested.responded <- right\_join(allSmalls.requested.responded, probSumm, by = "tag")  
#print  
probSumm

## # A tibble: 4 x 8  
## tag N samples respondedTotal prob base.weight sum.base.weight fpc  
## <fct> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 Bin A 1914 228 154 0.119 8.39 1914. 0.119  
## 2 Bin B 259 135 99 0.521 1.92 259. 0.521  
## 3 Bin C 107 84 69 0.785 1.27 107. 0.785  
## 4 Bin D 76 69 55 0.908 1.10 76 0.908

Note in the code above that *finite population correction* (fpc) is defined as the value of the sampling rate, *n/N* - not *1 - n/N* - which is the textbook definition of the *fpc*. This is an idiosyncracy of the *survey* package in R and is consistent with other programming languages such as Stata or SAS.

We can see here that the probability of being chosen within each strata is slightly different. A basic but important test that should be performed after computing the probabilities is making sure that all probabilities are between 0 and 1.

We will also calculate the scaled base weights, which should add up to the total number of respondents.

scaledSumm <- allSmalls.requested.responded %>%   
 group\_by(tag) %>%   
 summarize(n = n(), scaled.base.weight = base.weight/N\*(samples), sum.scaled.base.weight  
 = sum(scaled.base.weight))  
  
allSmalls %>%   
 mutate(scaled.base.weight = 1)  
  
allSmalls.requested.responded %>%   
 mutate(scaled.base.weight = 1)  
#print  
head(scaledSumm)

Here we can see that the scaled base weights are nominal (i.e. 1), and all add up to the respondent sample size for each bin. Later we will calcualted non-response weights and multiply them by the scaled base weights.

rm(probSumm, scaledSumm)  
probabilities <- allSmalls %>%  
 group\_by(tag) %>%   
 summarise(min.probability = min(prob, na.rm = T),  
 mean.probability = mean(prob, na.rm = T),  
 max.probability = max(prob, na.rm = T)) %>%  
 as.vector()  
print(probabilities)

## # A tibble: 4 x 4  
## tag min.probability mean.probability max.probability  
## <fct> <dbl> <dbl> <dbl>  
## 1 Bin A 0.119 0.119 0.119  
## 2 Bin B 0.521 0.521 0.521  
## 3 Bin C 0.785 0.785 0.785  
## 4 Bin D 0.908 0.908 0.908

#check to ensure probabiliities are realistic  
#if(probabilities$min.probability < 0){stop("Minimum probability of being sampled is smaller than 0. Review sampling probabilities before computing base weights.")}else if(probabilities$max.probability > 1){stop("Maximum probability of being sampled is larger than 1. Review sampling probabilities before computing base weights.")}  
rm(probabilities)

There may be underlying factors that make some types of water systems more or less likely to be sampled. For instance, let’s look at the mean probabilities by fee code.

allSmalls %>%  
 filter(!is.na(prob)) %>%  
 group\_by(tag, Fee\_Code) %>%  
 summarise(n = n(),  
 mean.prob.percentage = mean(prob, na.rm = T)\*100) %>%  
 arrange(desc(mean.prob.percentage)) %>%   
 head()

## # A tibble: 6 x 4  
## # Groups: tag [3]  
## tag Fee\_Code n mean.prob.percentage  
## <fct> <fct> <int> <dbl>  
## 1 Bin D C1 64 90.8  
## 2 Bin D DAVCL 12 90.8  
## 3 Bin C C1 83 78.5  
## 4 Bin C DAVCL 25 78.5  
## 5 Bin B C1 176 52.1  
## 6 Bin B DAVCL 87 52.1

Since some fee codes roughly fit within sampling bins (strata), it makes sense that we see discrete groupings. Other factors such as demographics may have other probabilities of being sampled. Let’s see what the difference in probabilities are for months before assistance, which has six discrete levels (A-F).

allSmalls %>%  
 group\_by(months\_before\_assist, tag) %>%  
 filter(!is.na(months\_before\_assist)) %>%   
 summarise(n = n(),mean.prob.percentage = mean(prob, na.rm = T)\*100) %>%  
 arrange(desc(mean.prob.percentage)) %>%   
 head()

## # A tibble: 6 x 4  
## # Groups: months\_before\_assist [6]  
## months\_before\_assist tag n mean.prob.percentage  
## <fct> <fct> <int> <dbl>  
## 1 A Bin D 1 90.8  
## 2 B Bin D 1 90.8  
## 3 C Bin D 2 90.8  
## 4 D Bin D 3 90.8  
## 5 E Bin D 16 90.8  
## 6 F Bin D 30 90.8

We can see there are minor, albeit seemingly significant differences in probabilities for sampling by months before assist.

To correct for these differential probabilities, we must design weights (sometimes called base weights) so that our sample does not over- or under-represent relevant groups. The design weights are equal to the inverse of the probability of inclusion to the sample. Therefore, the design weight (d0) of a respondent (i) will be equal to: d0[i] = 1/pi[i] where pi[1] is the probability of that unit being included in the sampling.

A simple interpretation of design weights is ‘the number of units in our population that each unit in our sample represents’. There is a simple but important test that we should perform after computing design weights. **The sum of all design weights should be equal to the total number of units in our population.**

Now to ensure design weights sum up to the entire population from which each population (bin) was drawn, we repeat the code from above for all smalls.

allSmalls %>%   
 mutate(sample = case\_when(requested == "y" ~ 1,requested == "n" ~ 0)) %>% #provide integer tag for sampled  
 group\_by(tag) %>%   
 summarize(n = n(),samples = sum(sample), prob = samples/n, base.weight = 1/prob,   
 sum.base.weight = base.weight\* samples) %>%   
 drop\_na

## # A tibble: 4 x 6  
## tag n samples prob base.weight sum.base.weight  
## <fct> <int> <dbl> <dbl> <dbl> <dbl>  
## 1 Bin A 1944 228 0.117 8.53 1944.  
## 2 Bin B 267 135 0.506 1.98 267   
## 3 Bin C 108 84 0.778 1.29 108.  
## 4 Bin D 76 69 0.908 1.10 76

### Large Systems

Below these steps are briefly repeated for large systems (greater than 10,000 service connections).

probSumm <- allLarges %>%   
 mutate(sample = case\_when(responded == "y" ~ 1,responded == "n" ~ 0)) %>% #provide integer tag for sampled  
 group\_by(tag) %>%   
 summarize(N = n(),samples = sum(sample), prob = samples/N, base.weight = 1/prob,  
 sum.base.weight = base.weight \* samples, fpc = samples/N)  
#join the probability to each sample  
allLarges <- right\_join(allLarges, probSumm, by = "tag")  
#print  
probSumm

## # A tibble: 1 x 7  
## tag N samples prob base.weight sum.base.weight fpc  
## <fct> <int> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 E 223 128 0.574 1.74 223 0.574

## Response Propensity Adjustment

Now that we have designed our basic weights, we now need to account for differences in the propensity to respond. It’s possible that certain profiles (e.g. lower income communities) had different propensities to respond than another profile (e.g. higher income communities). We could then imagine that the characteristics of both profiles were associated with reponse variables that we collected in this survey (e.g. water systems with lower income having more delinquent accounts than higher income systems). As we then would have a larger proportion of higher income/fewer delinquent accounts in our sample, our analyses would be biased.

### Small Systems

First let’s see the response rate for each of the bins for the small systems.

allSmalls %>% group\_by(tag) %>% filter(requested == "y") %>%   
 summarize(completeness = sum(response)/n() \* 100 , completed = sum(response), total = n())

## # A tibble: 4 x 4  
## tag completeness completed total  
## <fct> <dbl> <dbl> <int>  
## 1 Bin A 67.5 154 228  
## 2 Bin B 73.3 99 135  
## 3 Bin C 82.1 69 84  
## 4 Bin D 79.7 55 69

Computing the probability of replying to this survey is challenging because we can not directly observe the probability of replying to the survey, therefore we need to estimate it. This may be done using information which we know for both respondent and non-respondent units. The most reliable (albeit most complicated) of calculating non-response probabilities is through predictive modelling.

[Valliant et al (2013)](https://link.springer.com/book/10.1007%2F978-1-4614-6449-5) recommend estimating the response propensities and then grouping them in classes. The grouping step should avoid extreme weights. One way of estimating the response propensities is using logistic regression. This logistic regression should be unweighted. We will use a general linear model function to predict non-responses with hypothesized response variables for which we have data, including service connections, population, fee code, and regulating agency.

#trim data to those in sample list with auxiliary data  
allSmalls.requested.response.model <- allSmalls %>%   
 filter(requested == "y") %>%   
 select(response, Service\_Connections, Population, CES\_3.0\_Score, Median\_12month\_HH\_income,  
 Median\_rent\_pct\_income) %>%   
 drop\_na()  
  
#prepare formula  
formula.resp <- as.formula("response ~ Service\_Connections + Population + CES\_3.0\_Score + Median\_12month\_HH\_income + Median\_rent\_pct\_income")  
options(na.action = 'na.pass')  
  
#format as matrix  
x.matrix <- model.matrix(formula.resp , data = allSmalls.requested.response.model)[, -1]  
  
#fit binomial model  
log.reg.m <- glm(formula.resp, data = allSmalls.requested.response.model,family = "binomial")  
  
coef.response.log <- coef(log.reg.m)  
predicted.log <- log.reg.m$fitted.values  
non.responsepredicted.log <- predicted.log  
  
predicted.log %>% head()

## 1 2 3 4 5 6   
## 0.8703461 0.7392626 0.8655408 0.9114733 0.8905100 0.5794538

The above output shows the first six estimated response propensities of our dataset. Since we don’t have pardata for all samples, we are now computing our predictor estimates from a subset of sampled units.

Let’s define our survey structure in the *survey* package.

#join baseweights to imputed smalls  
imputedSmalls <- left\_join(imputedSmalls,   
 allSmalls %>% select(PWSID, base.weight, fpc),  
 by = "PWSID")  
##### Specify sampling design ####  
#here we will use our imputed survey dataset  
#(stratified)  
srv.dsgn <- svydesign(data = imputedSmalls,   
 weights = ~base.weight, #if weighted. Don't forget tilde ~  
 fpc = ~fpc, #population size for each stratum  
 strata = ~tag, #bins  
 id = ~1) #specifies independent sampling (without replacement)  
#specify replicate survey design  
rclus1 <- as.svrepdesign(srv.dsgn)  
#print summary of design  
summary(srv.dsgn)

The package *PracTools* has tools to determine propensity classes. We will use service connections, population, fee code, and regulating agency as predictor variables.

First let’s start by testing which variables are predictive of response.

#redefine all smalls  
allSmalls.requested <- allSmalls %>% filter(requested == "y")  
#fit  
fitResponse <- lm(response ~ Service\_Connections + Population + Median\_12month\_HH\_income + Median\_rent\_pct\_income + CES\_3.0\_Score + Fee\_Code, na.action = na.omit, data = allSmalls.requested)  
summary(fitResponse)

##   
## Call:  
## lm(formula = response ~ Service\_Connections + Population + Median\_12month\_HH\_income +   
## Median\_rent\_pct\_income + CES\_3.0\_Score + Fee\_Code, data = allSmalls.requested,   
## na.action = na.omit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.9158 -0.5981 0.2313 0.3057 0.5302   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.6826891707 0.1765457230 3.867 0.000125 \*\*\*  
## Service\_Connections 0.0000418843 0.0000202749 2.066 0.039376 \*   
## Population -0.0000074645 0.0000043509 -1.716 0.086871 .   
## Median\_12month\_HH\_income -0.0000003250 0.0000008822 -0.368 0.712762   
## Median\_rent\_pct\_income 0.0033984795 0.0037709554 0.901 0.367916   
## CES\_3.0\_Score -0.0027866305 0.0017033705 -1.636 0.102499   
## Fee\_CodeDAVCL 0.0842018936 0.0658133935 1.279 0.201367   
## Fee\_CodeDAVCS 0.0039998124 0.1135471346 0.035 0.971914   
## Fee\_CodeSC -0.0390688342 0.0632269529 -0.618 0.536922   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4403 on 485 degrees of freedom  
## (22 observations deleted due to missingness)  
## Multiple R-squared: 0.03724, Adjusted R-squared: 0.02136   
## F-statistic: 2.345 on 8 and 485 DF, p-value: 0.01764

Here we can see that the number of service connections and population are good predictors for response propensity, and possibly fee code (DAVCL). This is convenient, because we know these values for all systems. We will now define classes using this parameters using a logistic regression.

#be sure to use the dataset with non-response data  
out <- pclass(formula = response ~ Service\_Connections + Population + Fee\_Code,  
 data = allSmalls.requested,  
 type = "unwtd", #already have base weights  
 link = "logit",  
 numcl = 4) #classes to create  
table(out$p.class, useNA = "always") # ensures no unit has a missing class value

##   
## [0.369,0.672] (0.672,0.707] (0.707,0.795] (0.795,0.921] <NA>   
## 129 129 129 129 0

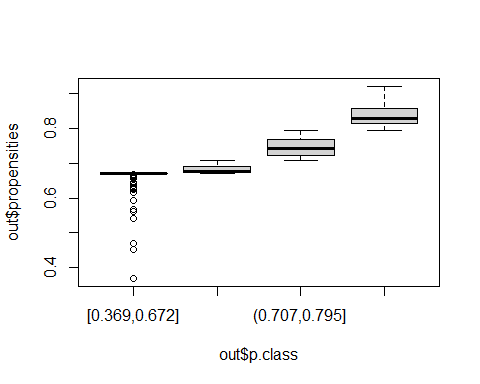
Further examine propensities.

summary(out$propensities)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.3694 0.6716 0.7073 0.7306 0.7954 0.9209

These propensity classes can be viewed in boxplot form.

boxplot(out$propensities ~ out$p.class)

 We can see discrete propensity classes with rather tight spread for the four uppermost categories. We see a string of outliers in the lowest class. We have several options to adjust our weights for non-response. These include multiplying the input weights by the inverse of cell response propensities. Let’s further examine these options.

cbind(  
 "mean" = by(data = out$propensities, INDICES = out$p.class, FUN = mean),  
 "median" = by(data = out$propensities, INDICES = out$p.class, FUN = median),  
 "weighted" = by(data = data.frame(preds = out$propensities, wt = allSmalls.requested[,"base.weight"]), out$p.class, function(x) {weighted.mean(x$preds, x$wt)}))

## mean median weighted  
## [0.369,0.672] 0.6575032 0.6706685 0.6670318  
## (0.672,0.707] 0.6827610 0.6769865 0.6801729  
## (0.707,0.795] 0.7446261 0.7414726 0.7373895  
## (0.795,0.921] 0.8375862 0.8295460 0.8355645

We can see that these are all quite similiar. So weighting by the base weights may not be necessary or distinct. Just to ensure, let’s perform a check on covariate balance by fitting an ANOVA model to service connections, which is continuous. We do not use the survey weights below since the interest is in whether balance has been achieved in the sample that was selected. Checks could be made using the weights, in which case the check would be on whether the census-fit model shows evidence of balance.

#extract classes  
p.class <- out$p.class  
#build glm  
chk1 <- glm(Service\_Connections ~ p.class + response + p.class\*response,  
data = allSmalls.requested)  
#print  
summary(chk1)

##   
## Call:  
## glm(formula = Service\_Connections ~ p.class + response + p.class \*   
## response, data = allSmalls.requested)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4434.7 -698.8 -451.4 634.6 8968.4   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 713.57 261.49 2.729 0.006576  
## p.class(0.672,0.707] 67.46 420.69 0.160 0.872674  
## p.class(0.707,0.795] 1687.28 447.50 3.770 0.000182  
## p.class(0.795,0.921] 4957.69 478.46 10.362 < 0.0000000000000002  
## response -243.19 342.95 -0.709 0.478581  
## p.class(0.672,0.707]:response 77.78 514.86 0.151 0.879982  
## p.class(0.707,0.795]:response 819.51 534.83 1.532 0.126078  
## p.class(0.795,0.921]:response 193.64 559.46 0.346 0.729389  
##   
## (Intercept) \*\*   
## p.class(0.672,0.707]   
## p.class(0.707,0.795] \*\*\*  
## p.class(0.795,0.921] \*\*\*  
## response   
## p.class(0.672,0.707]:response   
## p.class(0.707,0.795]:response   
## p.class(0.795,0.921]:response   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 3692485)  
##   
## Null deviance: 4079789347 on 515 degrees of freedom  
## Residual deviance: 1875782600 on 508 degrees of freedom  
## AIC: 9277.1  
##   
## Number of Fisher Scoring iterations: 2

In this case, the *p.class* factors all have coefficients that are significant while the p.class*resp interactions are not—the desired outcomes if mean service connections differs between classes but is the same for respondents and nonrespondents within a class. Another check is to fit a second model that includes only* p.class\* and to test whether the models are equivalent:

chk2 <- glm(Service\_Connections ~ p.class, data = allSmalls.requested)  
anova(chk2, chk1, test="F")

## Analysis of Deviance Table  
##   
## Model 1: Service\_Connections ~ p.class  
## Model 2: Service\_Connections ~ p.class + response + p.class \* response  
## Resid. Df Resid. Dev Df Deviance F Pr(>F)  
## 1 512 1885652237   
## 2 508 1875782600 4 9869637 0.6682 0.6143

The F-statistic is 0.647 with 506 and 502 degrees of freedom and has a p-value of 0.63. Thus, the model without a factor for responding is judged to be adequate.

#### Classification Algorithms = CART

In the nonrespondent application, the decision tree will classify cases using available covariates into classes that are related to their likelihood of being respondents. Advantages of CART compared to propensity modeling are that: 1. Interactions of covariates are handled automatically. 2. The way in which covariates enter the model does not have to be made explicit. 3. Selection of which covariates and associated interactions should be included is done automatically. 4. Variable values, whether categorical or continuous, are combined (grouped) automatically.

require(rpart)

## Warning: package 'rpart' was built under R version 4.0.3

set.seed(15097)  
t1 <- rpart(response ~ Service\_Connections + Population + Fee\_Code + Median\_rent\_pct\_income + Median\_12month\_HH\_income,  
 method = "class",  
 control = rpart.control(minbucket = 20, cp=0), #requires that there be at least 19 cases (responded + nonrespondents) in the final grouping of variable values of the terminal node of the tre..  
 data = imputedSmalls.requested.voluntary)  
print(t1, digits=4)

## n= 555   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 555 139 1 (0.2505 0.7495)   
## 2) Service\_Connections< 14.5 25 9 0 (0.6400 0.3600) \*  
## 3) Service\_Connections>=14.5 530 123 1 (0.2321 0.7679)   
## 6) Population< 57 43 16 1 (0.3721 0.6279) \*  
## 7) Population>=57 487 107 1 (0.2197 0.7803)   
## 14) Service\_Connections< 2874 315 77 1 (0.2444 0.7556)   
## 28) Population>=8779 21 10 1 (0.4762 0.5238) \*  
## 29) Population< 8779 294 67 1 (0.2279 0.7721)   
## 58) Median\_rent\_pct\_income< 32.91 175 47 1 (0.2686 0.7314)   
## 116) Median\_rent\_pct\_income>=27.4 103 35 1 (0.3398 0.6602)   
## 232) Median\_rent\_pct\_income< 28.66 20 9 0 (0.5500 0.4500) \*  
## 233) Median\_rent\_pct\_income>=28.66 83 24 1 (0.2892 0.7108) \*  
## 117) Median\_rent\_pct\_income< 27.4 72 12 1 (0.1667 0.8333) \*  
## 59) Median\_rent\_pct\_income>=32.91 119 20 1 (0.1681 0.8319) \*  
## 15) Service\_Connections>=2874 172 30 1 (0.1744 0.8256) \*

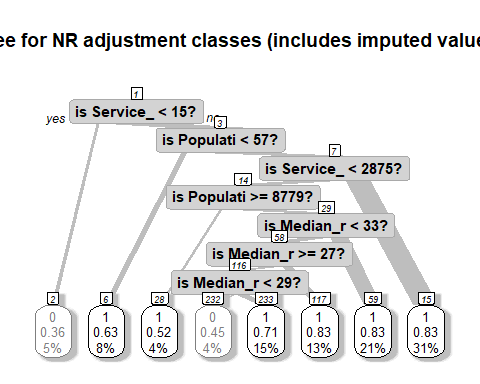
Plot an interpretable tree.

require(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.0.3

cols <- ifelse(t1$frame$yval == 1, "gray50", "black")  
prp(t1, main="Tree for NR adjustment classes (includes imputed values)",  
 extra=106, # display prob of survival and percent of obs  
 nn=TRUE, # display node numbers  
 fallen.leaves=TRUE, # put leaves on the bottom of page  
 branch=.5, # change angle of branch lines  
 faclen=0, # do not abbreviate factor levels  
 trace=1, # print automatically calculated cex  
 shadow.col="gray", # shadows under the leaves  
 branch.lty=1, # draw branches using solid lines  
 branch.type=5, # branch lines width = weight(frame$wt), no. of cases here  
 split.cex=1.2, # make split text larger than node text  
 split.prefix="is ", # put "is " before split text  
 split.suffix="?", # put "?" after split text  
 col=cols, border.col=cols, # cols[2] if survived  
 split.box.col="lightgray", # lightgray split boxes (default is white)  
 split.border.col="darkgray", # darkgray border on split boxes  
 split.round=0.5) # round the split box corners a tad

## cex 0.8 xlim c(0, 1) ylim c(0, 1)

 The tree can now be clearly visualized. This exercise will be repeated for the *non-imputed* samples without volunteers.

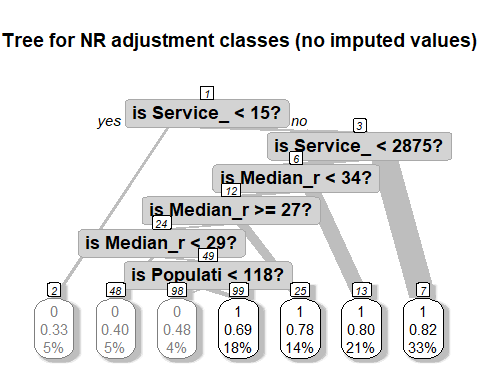
require(rpart)  
set.seed(15097)  
t2 <- rpart(response ~ Service\_Connections + Population + Fee\_Code + Median\_rent\_pct\_income + Median\_12month\_HH\_income,  
 method = "class",  
 control = rpart.control(minbucket = 20, cp=0), #requires that there be at least 19 cases (responded + nonrespondents) in the final grouping of variable values of the terminal node of the tre..  
 data = allSmalls.requested)  
print(t2, digits=4)

## n= 516   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 516 139 1 (0.2694 0.7306)   
## 2) Service\_Connections< 14.5 24 8 0 (0.6667 0.3333) \*  
## 3) Service\_Connections>=14.5 492 123 1 (0.2500 0.7500)   
## 6) Service\_Connections< 2874 321 93 1 (0.2897 0.7103)   
## 12) Median\_rent\_pct\_income< 34.33 214 72 1 (0.3364 0.6636)   
## 24) Median\_rent\_pct\_income>=27.17 142 56 1 (0.3944 0.6056)   
## 48) Median\_rent\_pct\_income< 28.66 25 10 0 (0.6000 0.4000) \*  
## 49) Median\_rent\_pct\_income>=28.66 117 41 1 (0.3504 0.6496)   
## 98) Population< 118 23 11 0 (0.5217 0.4783) \*  
## 99) Population>=118 94 29 1 (0.3085 0.6915) \*  
## 25) Median\_rent\_pct\_income< 27.17 72 16 1 (0.2222 0.7778) \*  
## 13) Median\_rent\_pct\_income>=34.33 107 21 1 (0.1963 0.8037) \*  
## 7) Service\_Connections>=2874 171 30 1 (0.1754 0.8246) \*

Plot an interpretable tree.

require(rpart.plot)  
cols <- ifelse(t2$frame$yval == 1, "gray50", "black")  
prp(t2, main="Tree for NR adjustment classes (no imputed values)",  
 extra=106, # display prob of survival and percent of obs  
 nn=TRUE, # display node numbers  
 fallen.leaves=TRUE, # put leaves on the bottom of page  
 branch=.5, # change angle of branch lines  
 faclen=0, # do not abbreviate factor levels  
 trace=1, # print automatically calculated cex  
 shadow.col="gray", # shadows under the leaves  
 branch.lty=1, # draw branches using solid lines  
 branch.type=5, # branch lines width = weight(frame$wt), no. of cases here  
 split.cex=1.2, # make split text larger than node text  
 split.prefix="is ", # put "is " before split text  
 split.suffix="?", # put "?" after split text  
 col=cols, border.col=cols, # cols[2] if survived  
 split.box.col="lightgray", # lightgray split boxes (default is white)  
 split.border.col="darkgray", # darkgray border on split boxes  
 split.round=0.5) # round the split box corners a tad

## cex 0.948 xlim c(0, 1) ylim c(0, 1)



#### Classification

A single regression tree does tend to overfit the data in the sense of creating a model that may not be accurate for a new dataset (like the units that were not sampled or another sample selected using the same methods that is also subject to nonresponse). For a nonresponse adjustment, the fitted model from a single tree may not be the best representation of the underlying response mechanism. Breiman (2001) formulated random forests as a way of creating predictions that suffer less from this “shrinkage” problem. Random forests fit many regression trees and average the results with the goal of producing more robust, lower variance predictions. Random forests can incorporate a large number of weighting variables and can find complicated relationships between adjustment variables that a researcher may not be aware of in advance( [Zhao et al. 2016](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4818178/)).

require(party)

## Warning: package 'party' was built under R version 4.0.3

## Warning: package 'mvtnorm' was built under R version 4.0.3

## Warning: package 'modeltools' was built under R version 4.0.3

## Warning: package 'strucchange' was built under R version 4.0.3

## Warning: package 'zoo' was built under R version 4.0.3

## Warning: package 'sandwich' was built under R version 4.0.3

crf.srvy <- cforest(as.factor(response) ~ Service\_Connections + Fee\_Code + Population + Median\_rent\_pct\_income + Median\_12month\_HH\_income,   
 controls = cforest\_control(ntree = 500,   
 mincriterion = qnorm(0.8),   
 trace = TRUE), # adds project bar because it's very slow  
 data=imputedSmalls.requested.voluntary)  
  
crfsrvy.prob <- predict(crf.srvy,newdata=imputedSmalls.requested.voluntary,type="prob")  
rpart.prob <- predict(t1, newdata=imputedSmalls.requested.voluntary,type="prob")  
crf.prob <- matrix(unlist(crfsrvy.prob), ncol=2, byrow=TRUE)  
apply(crf.prob,2,mean)  
#[1] 0.2524514 0.7475486  
tab <- round(cbind(by(rpart.prob[,2], INDICES=t1$where, mean), by(crf.prob[,2], INDICES=t1$where, mean)), 4)  
colnames(tab) <- c("rpart", "cforest")  
tab

The estimated overall response rate is 0.748. This is very close to the actual overall response rate of 0.7468. Above we can see the different propensity classes predicted by the rforest and the cforest.

require(party)  
crf.srvy.2 <- cforest(as.factor(response) ~ Service\_Connections + Fee\_Code + Population + Median\_rent\_pct\_income + Median\_12month\_HH\_income,   
 controls = cforest\_control(ntree = 500,   
 mincriterion = qnorm(0.8),   
 trace = TRUE), # adds project bar because it's very slow  
 data=allSmalls.requested)  
  
crfsrvy.prob2 <- predict(crf.srvy.2,newdata = allSmalls.requested,type="prob")  
rpart.prob2 <- predict(t2, newdata = allSmalls.requested,type="prob")  
crf.prob2 <- matrix(unlist(crfsrvy.prob2), ncol=2, byrow=TRUE)  
apply(crf.prob2,2,mean)  
#[1] 0.2711782 0.7288218  
tab <- round(cbind(by(rpart.prob2[,2], INDICES=t2$where, mean), by(crf.prob2[,2], INDICES=t2$where, mean)), 4)  
colnames(tab) <- c("rpart (no imputation)", "cforest (no imputation)")  
tab

Adjusted weights can now be computed and merged into the data file.

##### join to non-imputed data ####  
# compute NR adjustments based on classes formed by tree  
# Unweighted response rate  
unwt.rr <- by(as.numeric(allSmalls.requested[, "response"]), t2$where, mean)  
# Weighted response rate  
wt.rr <- by(data = data.frame(resp = as.numeric(allSmalls.requested[, "response"]), wt =  
 allSmalls.requested[,"base.weight"]),  
 t2$where, function(x) {weighted.mean(x$resp, x$wt)} )  
# merge NR class and response rates onto file  
allSmalls.requested.NR <- cbind(allSmalls.requested, NR.class=t2$where)  
tmp2 <- cbind(NR.class=as.numeric(names(wt.rr)), unwt.rr, wt.rr)  
allSmalls.requested.NR <- merge(allSmalls.requested.NR, data.frame(tmp2), by="NR.class")  
  
allSmalls.requested.NR <- allSmalls.requested.NR[order(allSmalls.requested.NR$PWSID),]  
#use inverse of response probabilitiy to generate non-response weight  
allSmalls.requested.NR %<>%   
 mutate(wt.rr.final = 1/wt.rr) %>%   
 mutate(unwt.rr.final = 1/unwt.rr)  
#display  
allSmalls.requested.NR$wt.rr.final %>%  
 unique %>%  
 sort

## [1] 1.201262 1.254923 1.314483 1.324701 2.090909 2.317558 3.000000

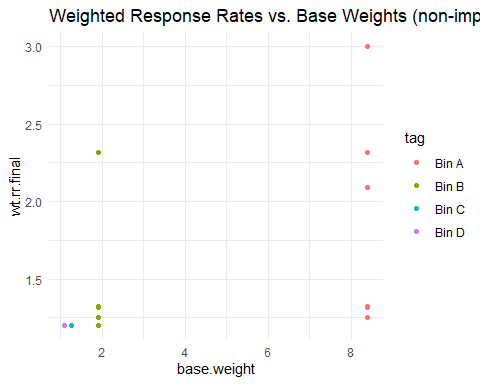
Here we can see the range of weighted nonresponse weights for the non-imputed survey (without volunteers).We can see below that these weighted nonresponse weights are quite similiar to the unweighted nonreponse weights for the non-imputed survey.

allSmalls.requested.NR$unwt.rr.final %>%  
 unique %>%  
 sort

## [1] 1.212766 1.244186 1.285714 1.446154 2.090909 2.500000 3.000000

These nonresponse weights are plotted relative to respective baseweights below.

#plot versus base weights  
ggplot(data = allSmalls.requested.NR, aes(x = base.weight, y = wt.rr.final, color = tag)) + geom\_point() +  
 labs(title = "Weighted Response Rates vs. Base Weights (non-imputed data)",  
 xlab = "Base Weights",  
 ylab = "Weighted Reponse Rates") +  
 theme\_minimal()

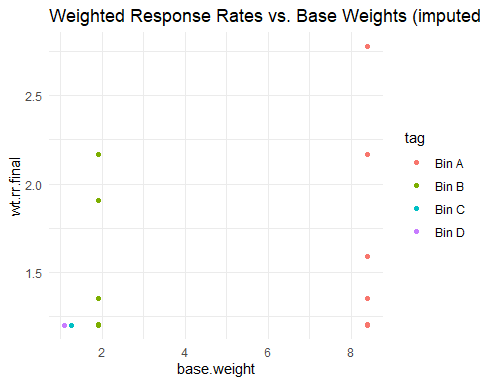


# extract base weights and join to data  
baseWeights <- allSmalls %>%   
 select(PWSID, base.weight, samples, N)  
  
# compute NR adjustments based on classes formed by tree  
# Unweighted response rate  
unwt.rr <- by(as.numeric(imputedSmalls.requested.voluntary[, "response"]), t1$where, mean)  
  
#join base weights to imputed  
imputedSmalls.requested.voluntary <- left\_join(imputedSmalls.requested.voluntary, baseWeights, by ="PWSID")   
  
# Weighted response rate  
wt.rr <- by(data = data.frame(resp = as.numeric(imputedSmalls.requested.voluntary[, "response"]), wt =  
 imputedSmalls.requested.voluntary[,"base.weight"]),  
 t1$where, function(x) {weighted.mean(x$resp, x$wt)} )  
# merge NR class and response rates onto file  
imputedSmalls.requested.voluntary.NR <- cbind(imputedSmalls.requested.voluntary, NR.class=t1$where)  
tmp1 <- cbind(NR.class=as.numeric(names(wt.rr)), unwt.rr, wt.rr)  
imputedSmalls.requested.voluntary.NR <- merge(imputedSmalls.requested.voluntary.NR, data.frame(tmp1), by="NR.class")  
imputedSmalls.requested.voluntary.NR <- imputedSmalls.requested.voluntary.NR[order(imputedSmalls.requested.voluntary.NR$PWSID),]  
#use inverse of response probabilitiy to generate non-response weight  
imputedSmalls.requested.voluntary.NR %<>%   
 mutate(wt.rr.final = 1/wt.rr) %>%   
 mutate(unwt.rr.final = 1/unwt.rr)  
  
imputedSmalls.requested.voluntary.NR$wt.rr.final %>%  
 unique %>%  
 sort

## [1] 1.199856 1.200000 1.203111 1.354916 1.592593 1.909091 2.164748 2.777778

The imputed data set contains slightly different weighted nonreponse weights.

#plot versus base weights  
ggplot(data = imputedSmalls.requested.voluntary.NR, aes(x = base.weight, y = wt.rr.final, color = tag)) + geom\_point() +  
 labs(title = "Weighted Response Rates vs. Base Weights (imputed data, with volunteers)",  
 xlab = "Base Weights",  
 ylab = "Weighted Reponse Rates") +  
 theme\_minimal()



If we had no information about population estimates, we would end the weighting procedure here. The ‘final weight’ would be the multiplication of both base scaled weight and the scaled non-response weight. Here we will call this new weights ‘final weights’ although we still have to perform adjustments to them and so will not really be ‘final’.

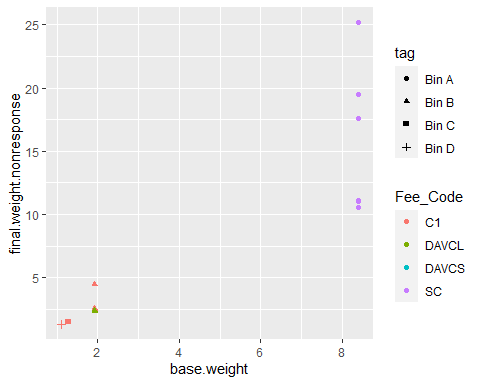
Before going to the next step we will include the computed non-response weights using adjustment classes to the main ‘data’ dataframe object. Then we will drop all non-respondents as we are not going to use them any more in the next steps of our analysis. After that, we will scale the non-response weights to the sample of respondents and multiply the design weights and the non-response weights.

##### Join to non-imputed data #####  
#join imputed smalls weights to full data frame  
allSmalls %<>%   
 left\_join(allSmalls.requested.NR %>% select(PWSID, wt.rr, wt.rr.final, unwt.rr, unwt.rr.final),  
 by = "PWSID")  
  
#join non-response weights to just the sample frame  
allSmalls.requested.responded %<>%   
 left\_join(allSmalls.requested.NR %>% select(PWSID, wt.rr, wt.rr.final, unwt.rr, unwt.rr.final),  
 by = "PWSID")  
  
#join the final weights to the respondent list  
allSmalls.requested.responded %<>%   
 mutate(final.weight.nonresponse = wt.rr.final \* base.weight)   
  
#view summary  
allSmalls.requested.responded %>%   
 select(tag, PWSID, base.weight, final.weight.nonresponse, N, samples) %>%   
 group\_by(tag) %>%   
 summarize(sum.final.weight = sum(final.weight.nonresponse), totalSamples = max(samples), population = max(N), percent.diff = 100\*(1 - sum.final.weight/(population)))

## # A tibble: 4 x 5  
## tag sum.final.weight totalSamples population percent.diff  
## <fct> <dbl> <dbl> <int> <dbl>  
## 1 Bin A 1932. 228 1914 -0.933  
## 2 Bin B 246. 135 259 5.10   
## 3 Bin C 106. 84 107 1.32   
## 4 Bin D 72.8 69 76 4.25

Here we can see that the sum of the weights adjusted for non-response are slightly different from the population. This is fine as they are within 6%. The adjusted weights are plotted below:

allSmalls.requested.responded %>%   
 ggplot(aes(x = base.weight, y =final.weight.nonresponse, color = Fee\_Code, shape = tag)) +  
 geom\_point()

 We can see clearly that the high non-response in the smaller bins has inflated the base weights.

Repeat these steps for the imputed data.

#join imputed smalls weights to imputed dataset  
imputedSmalls %<>%  
 left\_join(imputedSmalls.requested.voluntary.NR %>% select(PWSID, wt.rr.final, unwt.rr, unwt.rr.final),  
 by = "PWSID")  
  
#join non-response weights to just the sample frame  
imputedSmalls.requested.voluntary %<>%   
 left\_join(imputedSmalls.requested.voluntary.NR %>% select(PWSID, wt.rr.final, unwt.rr, unwt.rr.final),  
 by = "PWSID")  
  
#join the final weights to the respondent list  
imputedSmalls.requested.voluntary %<>%   
 mutate(final.weight.nonresponse = wt.rr.final \* base.weight) %>%   
 mutate(final.weight.nonresponse.unwt = unwt.rr.final \* base.weight)  
  
#view summary  
imputedSmalls.requested.voluntary %>%   
 select(tag, PWSID, base.weight, final.weight.nonresponse, final.weight.nonresponse.unwt, N, samples) %>%   
 group\_by(tag) %>%   
 summarize(sum.final.weight = sum(final.weight.nonresponse), sum.final.weight.unwt.rr = sum(final.weight.nonresponse.unwt), totalSamples = max(samples), population = max(N), percent.diff = 100\*(1 - sum.final.weight/(population)))

## # A tibble: 4 x 6  
## tag sum.final.weight sum.final.weight.unwt.rr totalSamples population  
## <fct> <dbl> <dbl> <dbl> <int>  
## 1 Bin A 3268. 3299. 228 1914  
## 2 Bin B 375. 379. 135 259  
## 3 Bin C 130. 131. 84 107  
## 4 Bin D 91.2 92.1 69 76  
## percent.diff  
## <dbl>  
## 1 -70.7  
## 2 -44.9  
## 3 -21.4  
## 4 -20.0

We can see that the dataset that included the volunteers dramatically inflates the nonresponse-adjusted weights. This is an unnaceptable difference.

rm(imputedSmalls.requested.voluntary.NR, allSmalls.requested.NR, chk1, chk2, crf.prob2, crfsrvy.prob2, crf.srvy.2, crf.prob, crf.srvy, crfsrvy.prob, fitResponse, allSmalls.requested.response.model, allSmalls.simple, baseWeights, completenessSummary, log.reg.m, out, probSumm, t1, t2, tab, tmp1, tmp2, transposed, x.matrix, rpart.prob, predM, MissingMatrix, scaledSumm, smalls, srv.dsgn, rclus1)

## Warning in rm(imputedSmalls.requested.voluntary.NR, allSmalls.requested.NR, :  
## object 'completenessSummary' not found

## Warning in rm(imputedSmalls.requested.voluntary.NR, allSmalls.requested.NR, :  
## object 'transposed' not found

## Warning in rm(imputedSmalls.requested.voluntary.NR, allSmalls.requested.NR, :  
## object 'scaledSumm' not found

### Large Systems

First let’s see the response rate for each of the bins for the large systems.

allLarges %>% filter(requested == "y") %>%  
 summarize(completeness = sum(response)/n() \* 100 , completed = sum(response), total = n())

## completeness completed total  
## 1 83.33333 125 150

Computing the probability of replying to this survey is challenging because we can not directly observe the probability of replying to the survey, therefore we need to estimate it. This may be done using information which we know for both respondent and non-respondent units. The most reliable (albeit most complicated) of calculating non-response probabilities is through predictive modelling.

[Valliant et al (2013)](https://link.springer.com/book/10.1007%2F978-1-4614-6449-5) recommend estimating the response propensities and then grouping them in classes. The grouping step should avoid extreme weights. One way of estimating the response propensities is using logistic regression. This logistic regression should be unweighted. We will use a general linear model function to predict non-responses with hypothesized response variables for which we have data, including service connections, population, fee code, and regulating agency.

#trim data to those in sample list with auxiliary data  
allLarges.requested.response.model <- allLarges %>%  
 filter(requested == "y") %>%  
 select(response, Service\_Connections, Population, CES\_3.0\_Score, Median\_12month\_HH\_income,  
 Median\_rent\_pct\_income) %>%  
 drop\_na()  
  
#prepare formula  
formula.resp <- as.formula("response ~ Service\_Connections + Population + CES\_3.0\_Score + Median\_12month\_HH\_income + Median\_rent\_pct\_income")  
options(na.action = 'na.pass')  
  
#format as matrix  
x.matrix <- model.matrix(formula.resp , data = allLarges.requested.response.model)[, -1]  
  
#fit binomial model  
log.reg.m <- glm(formula.resp, data = allLarges.requested.response.model,family = "binomial")  
  
coef.response.log <- coef(log.reg.m)  
predicted.log <- log.reg.m$fitted.values  
non.responsepredicted.log <- predicted.log  
  
predicted.log %>% head()

## 1 2 3 4 5 6   
## 1.0000000 0.9999996 0.9999848 0.9997569 0.9988361 0.9983223

The above output shows the first six estimated response propensities of our dataset. Since we don’t have pardata for all samples, we are now computing our predictor estimates from a subset of sampled units.

Let’s define our survey structure in the *survey* package.

##### Specify sampling design ####  
#here we will use our imputed survey dataset  
#(stratified)  
srv.dsgn.larges <- svydesign(data = allLarges,  
 weights = ~base.weight, #if weighted. Don't forget tilde ~  
 fpc = ~fpc, #population size for each stratum  
 strata = ~tag, #bins  
 id = ~1) #specifies independent sampling (without replacement)  
#print summary of design  
summary(srv.dsgn.larges)

The package *PracTools* has tools to determine propensity classes. We will use service connections, population, fee code, and regulating agency as predictor variables.

First let’s start by testing which variables are predictive of response.

#redefine all larges  
allLarges.requested <- allLarges %>% filter(requested == "y")  
#fit  
fitResponse <- lm(response ~ Service\_Connections + Population + Median\_12month\_HH\_income + Median\_rent\_pct\_income + CES\_3.0\_Score + Fee\_Code, na.action = na.omit, data = allLarges.requested)  
summary(fitResponse)

##   
## Call:  
## lm(formula = response ~ Service\_Connections + Population + Median\_12month\_HH\_income +   
## Median\_rent\_pct\_income + CES\_3.0\_Score + Fee\_Code, data = allLarges.requested,   
## na.action = na.omit)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.88331 0.04215 0.15029 0.20839 0.31883   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 1.2925492669 0.5068248900 2.550 0.0118 \*  
## Service\_Connections 0.0000025976 0.0000022025 1.179 0.2402   
## Population -0.0000003729 0.0000004223 -0.883 0.3787   
## Median\_12month\_HH\_income -0.0000030378 0.0000018087 -1.680 0.0952 .  
## Median\_rent\_pct\_income -0.0044052821 0.0122772700 -0.359 0.7203   
## CES\_3.0\_Score -0.0046871891 0.0037238508 -1.259 0.2102   
## Fee\_CodeDAVCL 0.1482603278 0.1235563826 1.200 0.2321   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.3704 on 143 degrees of freedom  
## Multiple R-squared: 0.05852, Adjusted R-squared: 0.01902   
## F-statistic: 1.481 on 6 and 143 DF, p-value: 0.1886

Unlike small systems, there are no good predictors for response propensity for large systems. This may be indicative of a non-systematic non-response.

#be sure to use the dataset with non-response data  
out <- pclass(formula = response ~ Service\_Connections + Population + Fee\_Code,  
 data = allLarges.requested,  
 type = "unwtd", #already have base weights  
 link = "logit",  
 numcl = 4) #classes to create  
table(out$p.class, useNA = "always") # ensures no unit has a missing class value

##   
## [0.687,0.753] (0.753,0.808] (0.808,0.913] (0.913,1] <NA>   
## 38 37 37 38 0

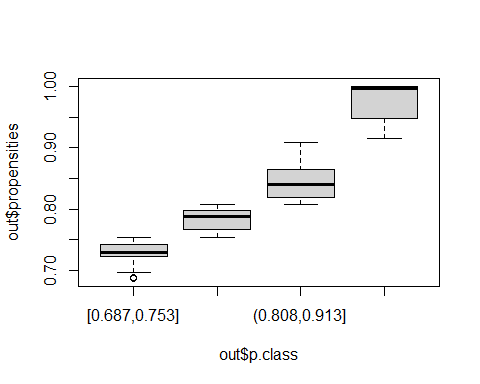
Further examine propensities.

summary(out$propensities)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.6874 0.7534 0.8082 0.8333 0.9134 1.0000

These propensity classes can be viewed in boxplot form.

boxplot(out$propensities ~ out$p.class)

 We can see discrete propensity classes with rather tight spread for the four uppermost categories. Let’s perform a check on covariate balance by fitting an ANOVA model to service connections, which is continuous. We do not use the survey weights below since the interest is in whether balance has been achieved in the sample that was selected. Checks could be made using the weights, in which case the check would be on whether the census-fit model shows evidence of balance.

#extract classes  
p.class <- out$p.class  
#build glm  
chk1 <- glm(Service\_Connections ~ p.class + response + p.class\*response,  
data = allLarges.requested)  
#print  
summary(chk1)

##   
## Call:  
## glm(formula = Service\_Connections ~ p.class + response + p.class \*   
## response, data = allLarges.requested)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -84924 -5392 -1217 1291 613647   
##   
## Coefficients: (1 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 12982.7 21975.8 0.591 0.556   
## p.class(0.753,0.808] 5197.3 32035.0 0.162 0.871   
## p.class(0.808,0.913] 18081.3 32035.0 0.564 0.573   
## p.class(0.913,1] 82859.5 16256.0 5.097 0.00000107 \*\*\*  
## response -353.8 25155.8 -0.014 0.989   
## p.class(0.753,0.808]:response 2576.1 36414.2 0.071 0.944   
## p.class(0.808,0.913]:response -159.8 36414.2 -0.004 0.997   
## p.class(0.913,1]:response NA NA NA NA   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 4346432665)  
##   
## Null deviance: 785009242279 on 149 degrees of freedom  
## Residual deviance: 621539871061 on 143 degrees of freedom  
## AIC: 3763.4  
##   
## Number of Fisher Scoring iterations: 2

In this case, the *p.class* factors do not have coefficients that are significant (except for the upper-most p-class). Further, the p.class*resp interactions are not significant—the desired outcomes if mean service connections differs between classes but is the same for respondents and nonrespondents within a class. Another check is to fit a second model that includes only* p.class\* and to test whether the models are equivalent:

chk2 <- glm(Service\_Connections ~ p.class, data = allLarges.requested)  
anova(chk2, chk1, test="F")

## Analysis of Deviance Table  
##   
## Model 1: Service\_Connections ~ p.class  
## Model 2: Service\_Connections ~ p.class + response + p.class \* response  
## Resid. Df Resid. Dev Df Deviance F Pr(>F)  
## 1 146 621573352295   
## 2 143 621539871061 3 33481234 0.0026 0.9998

The F-statistic is 0.0026 with 146 and 143 degrees of freedom and has a p-value of 0.9998. Thus, the model without a factor for responding is judged to be adequate.

#### Classification Algorithms = CART

In the nonrespondent application, the decision tree will classify cases using available covariates into classes that are related to their likelihood of being respondents. Advantages of CART compared to propensity modeling are that: 1. Interactions of covariates are handled automatically. 2. The way in which covariates enter the model does not have to be made explicit. 3. Selection of which covariates and associated interactions should be included is done automatically. 4. Variable values, whether categorical or continuous, are combined (grouped) automatically.

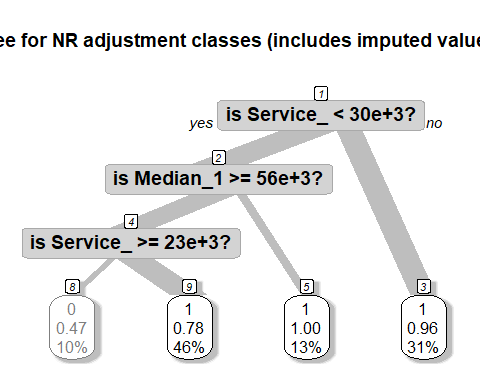
require(rpart)  
set.seed(15097)  
t1 <- rpart(response ~ Service\_Connections + Population + Fee\_Code + Median\_rent\_pct\_income + Median\_12month\_HH\_income,  
 method = "class",  
 control = rpart.control(minbucket = 15, cp=0), #requires that there be at least 19 cases (responded + nonrespondents) in the final grouping of variable values of the terminal node of the tre..  
 data = allLarges.requested)  
print(t1, digits=4)

## n= 150   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 150 25 1 (0.16667 0.83333)   
## 2) Service\_Connections< 3.011e+04 103 23 1 (0.22330 0.77670)   
## 4) Median\_12month\_HH\_income>=5.639e+04 84 23 1 (0.27381 0.72619)   
## 8) Service\_Connections>=2.328e+04 15 7 0 (0.53333 0.46667) \*  
## 9) Service\_Connections< 2.328e+04 69 15 1 (0.21739 0.78261) \*  
## 5) Median\_12month\_HH\_income< 5.639e+04 19 0 1 (0.00000 1.00000) \*  
## 3) Service\_Connections>=3.011e+04 47 2 1 (0.04255 0.95745) \*

Plot an interpretable tree.

require(rpart.plot)  
cols <- ifelse(t1$frame$yval == 1, "gray50", "black")  
prp(t1, main="Tree for NR adjustment classes (includes imputed values)",  
 extra=106, # display prob of survival and percent of obs  
 nn=TRUE, # display node numbers  
 fallen.leaves=TRUE, # put leaves on the bottom of page  
 branch=.5, # change angle of branch lines  
 faclen=0, # do not abbreviate factor levels  
 trace=1, # print automatically calculated cex  
 shadow.col="gray", # shadows under the leaves  
 branch.lty=1, # draw branches using solid lines  
 branch.type=5, # branch lines width = weight(frame$wt), no. of cases here  
 split.cex=1.2, # make split text larger than node text  
 split.prefix="is ", # put "is " before split text  
 split.suffix="?", # put "?" after split text  
 col=cols, border.col=cols, # cols[2] if survived  
 split.box.col="lightgray", # lightgray split boxes (default is white)  
 split.border.col="darkgray", # darkgray border on split boxes  
 split.round=0.5) # round the split box corners a tad

## cex 1 xlim c(0, 1) ylim c(0, 1)



#### Classification

A single regression tree does tend to overfit the data in the sense of creating a model that may not be accurate for a new dataset (like the units that were not sampled or another sample selected using the same methods that is also subject to nonresponse). For a nonresponse adjustment, the fitted model from a single tree may not be the best representation of the underlying response mechanism. Breiman (2001) formulated random forests as a way of creating predictions that suffer less from this “shrinkage” problem. Random forests fit many regression trees and average the results with the goal of producing more robust, lower variance predictions. Random forests can incorporate a large number of weighting variables and can find complicated relationships between adjustment variables that a researcher may not be aware of in advance( [Zhao et al. 2016](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4818178/)).

require(party)  
crf.srvy.larges <- cforest(as.factor(response) ~ Service\_Connections + Fee\_Code + Population + Median\_rent\_pct\_income + Median\_12month\_HH\_income,  
 controls = cforest\_control(ntree = 500,  
 mincriterion = qnorm(0.8),  
 trace = TRUE), # adds project bar because it's very slow  
 data=allLarges.requested)  
  
crfsrvy.prob.larges <- predict(crf.srvy.larges,newdata=allLarges.requested,type="prob")  
rpart.prob.larges <- predict(t1, newdata=allLarges.requested,type="prob")  
crf.prob.larges <- matrix(unlist(crfsrvy.prob.larges), ncol=2, byrow=TRUE)  
apply(crf.prob.larges,2,mean)  
#[1] 0.1701657 0.8298343  
tab <- round(cbind(by(rpart.prob.larges[,2], INDICES=t1$where, mean), by(crf.prob.larges[,2], INDICES=t1$where, mean)), 4)  
colnames(tab) <- c("rpart", "cforest")  
tab

The estimated overall response rate is 0.8298. This is very close to the actual overall response rate of 0.833. Above we can see the different propensity classes predicted by the rforest and the cforest. The cforest seemed to generate more realistic propensity classes that match the data (see boxplot above).

Adjusted weights can now be computed and merged into the data file.

##### join to non-imputed data ####  
# compute NR adjustments based on classes formed by tree  
# Unweighted response rate  
  
#unwt.rr <- by(as.numeric(unlist(allLarges.requested[, "response"])), crf.prob.larges[,2], mean)  
  
# # Weighted response rate  
#wt.rr <- by(data = data.frame(resp = as.numeric(unlist(allLarges.requested[, "response"])),wt = as.numeric(unlist(allLarges.requested[,"base.weight"]))),  
# crf.prob.larges[,2], function(x) {weighted.mean(x$resp, x$wt)} )  
#  
# unwt.rr  
# # merge NR class and response rates onto file  
 # allLarges.requested.NR <- cbind(allLarges.requested, NR.class=crf.prob.larges[,2])  
 # tmp2 <- data.frame(cbind(NR.class=as.numeric(names(wt.rr)), unwt.rr, wt.rr))  
 # allLarges.requested.NR <- left\_join(allLarges.requested.NR, data.frame(tmp2), by="NR.class")  
#  
 # allLarges.requested.NR <- allLarges.requested.NR[order(allLarges.requested.NR$PWSID),]  
  
#extract response propensities from CART  
unwt.rr <- as.data.frame(crf.prob.larges[,2])  
#append to dataset  
allLarges.requested.NR <- cbind(allLarges.requested, unwt.rr)  
allLarges.requested.NR %<>% rename(c("unwt.rr" = `crf.prob.larges[, 2]`))  
#use inverse of response probabilitiy to generate non-response weight  
allLarges.requested.NR %<>%  
 mutate(unwt.rr.final = 1/unwt.rr)  
#display  
allLarges.requested.NR$unwt.rr.final %>%  
 unique %>%  
 sort

## [1] 1.018274 1.018946 1.019195 1.021061 1.022284 1.024795 1.025349 1.029813  
## [9] 1.032465 1.032820 1.034235 1.034735 1.035547 1.035569 1.036394 1.037952  
## [17] 1.038420 1.040481 1.041059 1.041555 1.043095 1.044565 1.048608 1.053234  
## [25] 1.057827 1.058501 1.059835 1.060522 1.068071 1.070041 1.071270 1.072454  
## [33] 1.072849 1.073904 1.077108 1.077889 1.080527 1.086333 1.087073 1.090726  
## [41] 1.091263 1.092958 1.093776 1.097694 1.099201 1.100280 1.100583 1.102730  
## [49] 1.104026 1.104289 1.107506 1.110761 1.112706 1.115361 1.117382 1.118045  
## [57] 1.121640 1.126347 1.126794 1.128269 1.132682 1.134564 1.135758 1.137673  
## [65] 1.139261 1.140321 1.142467 1.143065 1.143638 1.143824 1.144279 1.146196  
## [73] 1.148880 1.155428 1.155941 1.157872 1.160714 1.161081 1.170602 1.176663  
## [81] 1.182862 1.183461 1.184467 1.185414 1.202297 1.210265 1.212545 1.216678  
## [89] 1.218884 1.221027 1.221881 1.223461 1.226169 1.226461 1.233573 1.234879  
## [97] 1.237489 1.237781 1.238656 1.239810 1.257561 1.264989 1.279552 1.280178  
## [105] 1.281235 1.282022 1.289293 1.289968 1.303474 1.305638 1.312134 1.314026  
## [113] 1.314523 1.315763 1.322209 1.327147 1.327215 1.336327 1.350335 1.367290  
## [121] 1.388395 1.390945 1.392759 1.403273 1.415402 1.419795 1.425021 1.438948  
## [129] 1.444469 1.463698 1.489879 1.494060 1.511230 1.511289 1.524122 1.547445  
## [137] 1.554574 1.556485 1.564907 1.605095 1.609787 1.614493 1.615918 1.625159  
## [145] 1.641292 1.685041 1.729847 1.757939 1.794773 1.823946

Here we can see the range of weighted nonresponse weights for the non-imputed survey (without volunteers).

If we had no information about population estimates, we would end the weighting procedure here. The ‘final weight’ would be the multiplication of both base scaled weight and the scaled non-response weight. Here we will call this new weights ‘final weights’ although we still have to perform adjustments to them and so will not really be ‘final’.

Before going to the next step we will include the computed non-response weights using adjustment classes to the main ‘data’ dataframe object. Then we will drop all non-respondents as we are not going to use them any more in the next steps of our analysis. After that, we will scale the non-response weights to the sample of respondents and multiply the design weights and the non-response weights.

##### Join to non-imputed data #####  
#join imputed smalls weights to full data frame  
allLarges %<>%  
 left\_join(allLarges.requested.NR %>% select(PWSID, unwt.rr, unwt.rr.final),  
 by = "PWSID")  
allLarges %<>% mutate(final.weight.nonresponse = (unwt.rr.final \* base.weight)/1.4437)  
  
  
# #join non-response weights to just the sample frame  
# allLarges.requested.responded %<>%  
# left\_join(allLarges.requested.NR %>% select(PWSID, unwt.rr, unwt.rr.final),  
# by = "PWSID")  
  
# #join the final weights to the respondent list  
# allLarges.requested.responded %<>%  
# mutate(final.weight.nonresponse = unwt.rr.final \* base.weight)  
  
#view summary  
allLarges %>%  
 select(PWSID, base.weight, final.weight.nonresponse, N, samples) %>%  
 drop\_na() %>%  
 summarize(sum.final.weight = sum(final.weight.nonresponse), totalSamples = max(samples), population = max(N), percent.diff = 100\*(1 - sum.final.weight/(population)))

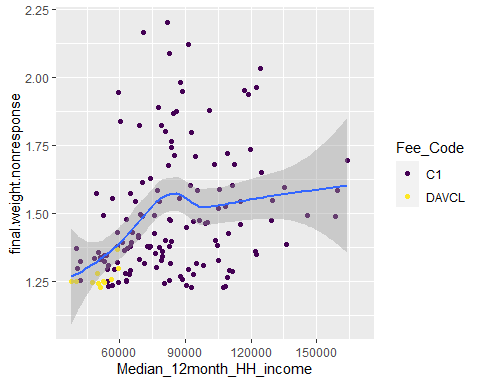
## sum.final.weight totalSamples population percent.diff  
## 1 222.0626 128 223 0.4203467

Here we can see that the sum of the weights adjusted for non-response are the same as the population. The adjusted weights are plotted below:

allLarges %>%  
 ggplot(aes(x = Median\_12month\_HH\_income, y =final.weight.nonresponse)) +  
 geom\_point(aes(color = Fee\_Code))+  
 scale\_color\_viridis\_d()+  
 geom\_smooth()

## Warning: Removed 73 rows containing non-finite values (stat\_smooth).

## Warning: Removed 73 rows containing missing values (geom\_point).

 Although the relationship between non-response and median 12 month household income is non-linear (as predicted above using a multi-variate, machine-learning classification-based approach), a relationship may be observed between response propensity and median income in the above figure.

## Weight Calibration

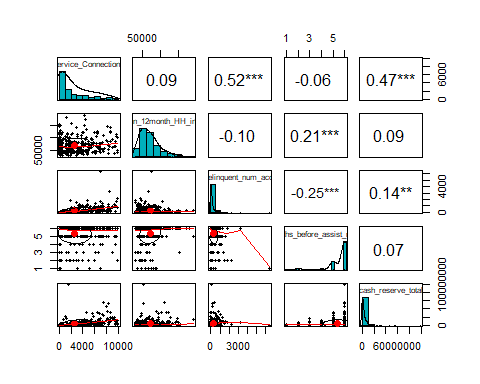
We’ve now generated *base weights* and *nonresponse adjustments* to those weights. The last step, which is extremely important in many surveys, is to use auxiliary data to correct coverage problems and to reduce standard errors.By auxiliary data, we mean information that is available for the entire frame or target population, either for each individual population unit or in aggregate form (i.e. househould income, service connections, population, fee code, regulating agency).Using auxiliary variable to adjust survey variables may reduce variances, but improve the representativeness of the survey.

Since our auxiliary variables are both quantitative(i.e. househould income, service connections, population) and qualitative (fee code, regulating agency), we will use a general regression estimator or GREG approach, which is a type of raking estimator. A GREG is approximately unbiased in repeated sampling if the frame provides full coverage of the target population (in this case we have census data for these auxiliary variables) ( [Vallian et al 2018](http://link.springer.com/10.1007/978-3-319-93632-1)).

### Evaluation of Correlative factors

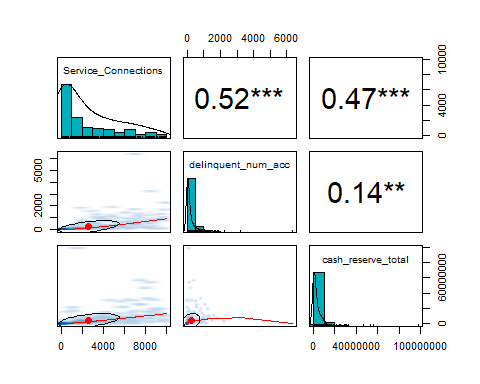
The first step in selecting appropriate auxiliary variables to use for raking is to deterimine correlations with response variables of interest. This can be achieved using a scatterplot matrix of the quantitative variables in the dataset.

require(psych)  
#trim data  
aux.response <- allSmalls.requested.responded %>% select(Service\_Connections,Median\_12month\_HH\_income, delinquent\_num\_acc, months\_before\_assist\_num, cash\_reserve\_total)  
# plot matrix  
pairs.panels(aux.response,  
 method = "pearson",  
 hist.col = "#00AFBB", # color  
 density = TRUE, #show histograms  
 ellipses = TRUE,#annotate correlation elliopses  
 stars = TRUE) #show significance)



Here we can see that all of the quantitative auxiliary variables (Service\_Connections, Median\_12month\_HH\_income) are correlated with the selected response variables (delinquent number of accounts, months before assistance needed, cash reserve total). These are appropriate variables to use for raking. Let’s look a little closer at the relationships between service connections and delinquent number of accounts and total cash reserve.

require(psych)  
#trim data  
aux.response <- allSmalls.requested.responded %>%   
 select(Service\_Connections, delinquent\_num\_acc, cash\_reserve\_total)  
# plot matrix  
pairs.panels(aux.response,  
 method = "pearson",  
 hist.col = "#00AFBB", # color  
 density = TRUE, #show histograms  
 ellipses = TRUE,#annotate correlation elliopses  
 smoother = TRUE,  
 stars = TRUE) #show significance)

 We can see that service connections is a useful linear predictor for delinquent number of accounts (r = 0.52) and total cash reserve (r = 0.47). Interestingly, the delinquent number of accoutns and total cash reserves has a very non-linear relationship, resembling a volcano plot.

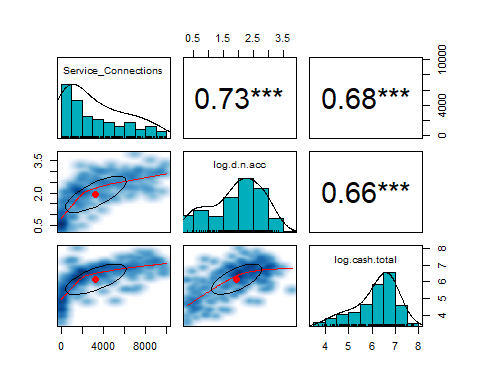
It may be better to visualize the number of delinquent accounts and cash totals on a log10 scale.

#trim data  
aux.response\_log <- allSmalls.requested.responded %>%   
 select(Service\_Connections, delinquent\_num\_acc, cash\_reserve\_total) %>%   
 mutate(log.d.n.acc = log10(delinquent\_num\_acc)) %>%   
 mutate(log.cash.total = log10(cash\_reserve\_total)) %>%   
 drop\_na() %>%   
 filter(log.d.n.acc > 0) %>%   
 filter(log.cash.total > 0) %>%   
 select(Service\_Connections, log.d.n.acc, log.cash.total)

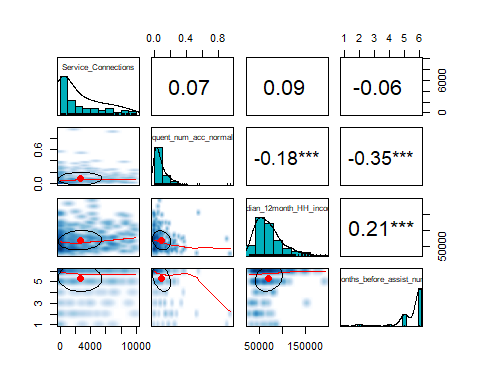
## Warning: Problem with `mutate()` input `log.cash.total`.  
## i NaNs produced  
## i Input `log.cash.total` is `log10(cash\_reserve\_total)`.

## Warning in mask$eval\_all\_mutate(dots[[i]]): NaNs produced

# plot matrix  
pairs.panels(aux.response\_log,  
 method = "pearson",  
 hist.col = "#00AFBB", # color  
 density = TRUE, #show histograms  
 ellipses = TRUE,#annotate correlation elliopses  
 smoother = TRUE,  
 stars = TRUE) #show significance)

 After log transformation we can see more linear relationships between the predictor (service connections) and response variables. Delinquent number of accounts normalized to service connections is a critical response variable that may be predicted from auxiliary variables.

#trim data  
aux.response <- allSmalls.requested.responded %>%   
 mutate(logServiceConnections = log10(Service\_Connections)) %>%   
 select(Service\_Connections, delinquent\_num\_acc\_normalized, Median\_12month\_HH\_income, months\_before\_assist\_num)  
# plot matrix  
pairs.panels(aux.response,  
 method = "pearson",  
 hist.col = "#00AFBB", # color  
 density = TRUE, #show histograms  
 ellipses = TRUE,#annotate correlation elliopses  
 smoother = TRUE,  
 stars = TRUE) #show significance)



Here we can see that service connections and median 12 month household income are not good predictors of the service-connection *normalized* number of delinquent accounts. However, a strong relationship exists between the *normalized* number of delinquent accounts and the qualitative variable for months before assistance needed. This is not useful for raking, as neither variables are auxiliary, however this is an interesting (and expected) correlation. Again, median 12 month household income and months before assistance needed are significantly correlated.

To continue raking, we will now perform more formal analyses. Using all reliable, complete auxiliary data present, let’s see the linear relationships to the continuous variable months before assistance needed (from factor).

m2 <- glm(months\_before\_assist\_num ~ Service\_Connections + Median\_12month\_HH\_income + Median\_rent\_pct\_income + Fee\_Code, na.action = na.exclude, data = allSmalls.requested.responded)  
summary(m2)

##   
## Call:  
## glm(formula = months\_before\_assist\_num ~ Service\_Connections +   
## Median\_12month\_HH\_income + Median\_rent\_pct\_income + Fee\_Code,   
## data = allSmalls.requested.responded, na.action = na.exclude)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.7491 -0.2229 0.3951 0.6825 1.3910   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 4.137306312 0.527438955 7.844 0.0000000000000615  
## Service\_Connections -0.000025077 0.000033778 -0.742 0.458373  
## Median\_12month\_HH\_income 0.000008711 0.000002563 3.399 0.000761  
## Median\_rent\_pct\_income 0.021822315 0.012416149 1.758 0.079750  
## Fee\_CodeDAVCL -0.389227265 0.206986478 -1.880 0.060930  
## Fee\_CodeDAVCS 0.032420714 0.393655413 0.082 0.934412  
## Fee\_CodeSC 0.015550889 0.209543346 0.074 0.940886  
##   
## (Intercept) \*\*\*  
## Service\_Connections   
## Median\_12month\_HH\_income \*\*\*  
## Median\_rent\_pct\_income .   
## Fee\_CodeDAVCL .   
## Fee\_CodeDAVCS   
## Fee\_CodeSC   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 1.419319)  
##   
## Null deviance: 502.33 on 335 degrees of freedom  
## Residual deviance: 466.96 on 329 degrees of freedom  
## (41 observations deleted due to missingness)  
## AIC: 1080.1  
##   
## Number of Fisher Scoring iterations: 2

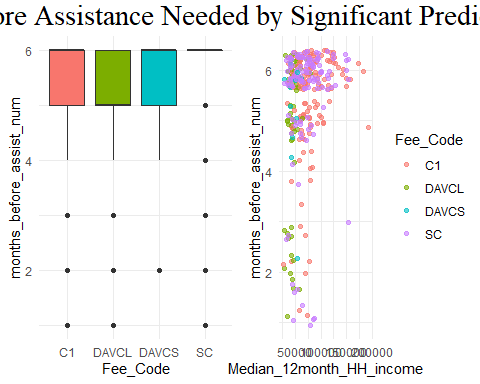
Again we can see that median 12 month household income is a strong linear predictor for months before assistance needed. A non-significant trend exists between disadvantaged smalls and months before assistance (as would be expected).

The relationship between fee code and months before assistance needed can be visualized as a boxplot.

p1 <- allSmalls.requested.responded %>%   
 ggplot(aes(x = Fee\_Code, y = months\_before\_assist\_num, fill = Fee\_Code)) +  
 geom\_boxplot() +  
 theme\_minimal() +  
 theme(legend.position = "none")  
p2 <- allSmalls.requested.responded %>%   
 ggplot(aes(x = Median\_12month\_HH\_income, y = months\_before\_assist\_num, color = Fee\_Code)) +  
 geom\_point(position = "jitter", alpha = 0.6) +  
 theme\_minimal()  
  
grid.arrange(p1, p2,ncol = 2,  
 top = textGrob("Months Before Assistance Needed by Significant Predictive Factors",  
 gp=gpar(fontsize = 22,font=6)))

## Warning: Removed 25 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 41 rows containing missing values (geom\_point).

 We can see quite clearly the relationship between fee code and months before assistance needed, with the disadvantaged community larges representing the most at-risk overall, and the community smalls representing lowest risk. Again remember that these values are unweighted and are only used for selecting auxiliary variables for raking.Nonetheless, the relationship between median 12 month household income and months before assistance needed seems to be quite robust. There are, however, some outliers that may negatively affect raking. We may choose to remove these outliers for raking purposes.

require(car)  
#glm for only predictive variables  
m3 <- glm(months\_before\_assist\_num ~ Median\_12month\_HH\_income + Fee\_Code,na.action = na.exclude, data = allSmalls.requested.responded)  
# Bonferonni p-value for most extreme obs  
outlierTest(m3)

## rstudent unadjusted p-value Bonferroni p  
## 305 -3.933773 0.000083623 0.028097  
## 341 -3.925684 0.000086484 0.029059  
## 259 -3.875518 0.000106400 0.035750

It seems that three values are extreme outliers.

#code row number  
allSmalls.requested.responded %<>%   
 mutate(id = row\_number())  
#examine cases  
allSmalls.requested.responded %>%  
 filter(id == 305 | id == 341 | id == 259)

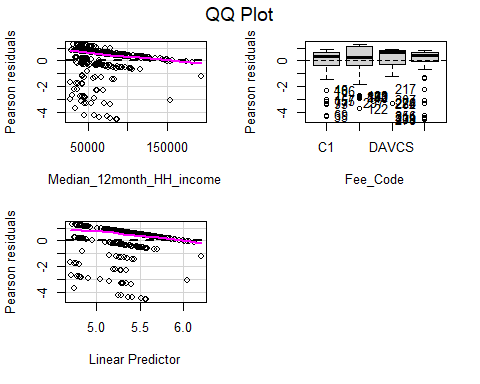
Here we can see these are quite small systems. The Fisherman’s retreat in Riverside, Belmont manor in Fresno, and Wayside motel apparments in San Joaquin county. The notes indicate that the last one does not have financial information for the water system and the first is just for seven months of expenses. The first and third report no cash reserves. The third reports that many people are not paying rent.All are in relatively moderate income areas (78-86 k/year median). These are important data points that should be considered later, but may innapropriately skew the raking process.

Let’s look at residuals for predictive variables.

summary(m3)

##   
## Call:  
## glm(formula = months\_before\_assist\_num ~ Median\_12month\_HH\_income +   
## Fee\_Code, data = allSmalls.requested.responded, na.action = na.exclude)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -4.5772 -0.2820 0.4354 0.6860 1.2787   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 4.846150113 0.217354419 22.296 < 0.0000000000000002  
## Median\_12month\_HH\_income 0.000007077 0.000002411 2.935 0.00357  
## Fee\_CodeDAVCL -0.350304495 0.205185521 -1.707 0.08871  
## Fee\_CodeDAVCS 0.075870097 0.367880088 0.206 0.83673  
## Fee\_CodeSC 0.112144482 0.147551287 0.760 0.44777  
##   
## (Intercept) \*\*\*  
## Median\_12month\_HH\_income \*\*   
## Fee\_CodeDAVCL .   
## Fee\_CodeDAVCS   
## Fee\_CodeSC   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 1.425995)  
##   
## Null deviance: 502.33 on 335 degrees of freedom  
## Residual deviance: 472.00 on 331 degrees of freedom  
## (41 observations deleted due to missingness)  
## AIC: 1079.7  
##   
## Number of Fisher Scoring iterations: 2

residualPlots(m3, main="QQ Plot") #qq plot for studentized resid



## Test stat Pr(>|Test stat|)  
## Median\_12month\_HH\_income 2.0059 0.1567  
## Fee\_Code

Since there are considerable deviant residuals for this model, let’s try a more simple model with an intuitive and likely simpler dependent variable (total cash reserves).

#create simple dataset for modeling that drops missing values and provides log transform   
simpleModel <- allSmalls.requested.responded %>%   
 select(PWSID, base.weight, final.weight.nonresponse, Service\_Connections, delinquent\_num\_acc, cash\_reserve\_total, Fee\_Code, Median\_12month\_HH\_income) %>%   
 mutate(log.d.n.acc = log10(delinquent\_num\_acc)) %>%   
 mutate(log.cash.total = log10(cash\_reserve\_total)) %>%   
 drop\_na() %>%   
 filter(log.d.n.acc > 0) %>%   
 filter(log.cash.total > 0) %>%   
 select(Service\_Connections, log.d.n.acc, log.cash.total, PWSID, base.weight, final.weight.nonresponse, Service\_Connections, delinquent\_num\_acc, cash\_reserve\_total, Fee\_Code, Median\_12month\_HH\_income) %>%   
 droplevels()

## Warning: Problem with `mutate()` input `log.cash.total`.  
## i NaNs produced  
## i Input `log.cash.total` is `log10(cash\_reserve\_total)`.

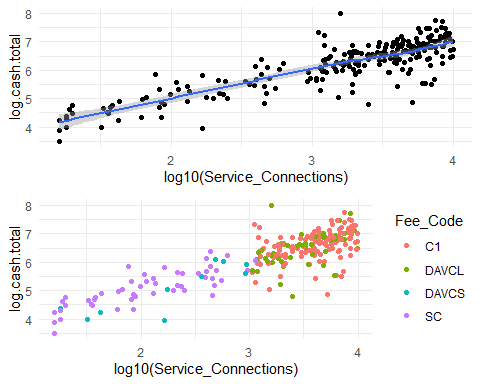
## Warning in mask$eval\_all\_mutate(dots[[i]]): NaNs produced

#build glm  
m5 <- glm(log.cash.total ~ Service\_Connections + Median\_12month\_HH\_income + Fee\_Code, na.action = na.exclude, data = simpleModel)  
#report  
summary(m5)

##   
## Call:  
## glm(formula = log.cash.total ~ Service\_Connections + Median\_12month\_HH\_income +   
## Fee\_Code, data = simpleModel, na.action = na.exclude)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.7542 -0.2662 0.0561 0.3109 1.6958   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 5.976528956 0.132823414 44.996 < 0.0000000000000002  
## Service\_Connections 0.000100326 0.000016102 6.231 0.00000000218  
## Median\_12month\_HH\_income 0.000001885 0.000001379 1.367 0.173  
## Fee\_CodeDAVCL 0.062504988 0.099081745 0.631 0.529  
## Fee\_CodeDAVCS -1.030640099 0.190510953 -5.410 0.00000015735  
## Fee\_CodeSC -1.042674500 0.113021073 -9.225 < 0.0000000000000002  
##   
## (Intercept) \*\*\*  
## Service\_Connections \*\*\*  
## Median\_12month\_HH\_income   
## Fee\_CodeDAVCL   
## Fee\_CodeDAVCS \*\*\*  
## Fee\_CodeSC \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.2821804)  
##   
## Null deviance: 174.486 on 236 degrees of freedom  
## Residual deviance: 65.184 on 231 degrees of freedom  
## AIC: 380.65  
##   
## Number of Fisher Scoring iterations: 2

Here we can see a much simpler model with a lower AIC (380.65) compared to the models that tried to predict months before assistance needed (AIC = 1079.7). We will use this simple relationship to calibrate the data. Let’s see this highly correlated relationship in graphical form.

p1 <- simpleModel %>%   
 ggplot(aes(x = log10(Service\_Connections), y = log.cash.total)) +  
 geom\_point() +  
 geom\_smooth()+  
 theme\_minimal()   
  
p2 <- simpleModel %>%   
 ggplot(aes(x = log10(Service\_Connections), y = log.cash.total, color = Fee\_Code)) +  
 geom\_point() +  
 theme\_minimal()   
grid.arrange(p1, p2)

 Here we can see the strong linear relationship between cash total and service connections, which will form the basis for building a model to rake the data.

### Calibration

#### Small Systems

The code below uses the *sampling* package to select a sample with probability proportional to the median 12 month household income . The method of selection is to randomize the order of the population and then select a systematic sample (see Hartley and Rao 1962).

#get population totals for items of interest  
N.PS <- xtabs(~Fee\_Code, data = allSmalls)

require(survey)  
#build survey design with no weights  
srv.dsgn.unweighted <- svydesign(ids = ~0,  
 strata = ~tag,  
 data = allSmalls.requested.responded,  
 fpc = ~fpc,  
 weights = ~NULL)  
#build survey design with non-adjusted weights  
srv.dsgn.unadjusted <- svydesign(ids = ~ 0, # no clusters  
 strata = ~tag,  
 data = allSmalls.requested.responded,  
 fpc = ~fpc,  
 weights = ~base.weight)  
#build survey design with adjusted non-response weights  
srv.dsgn <- svydesign(ids = ~ 0, # no clusters  
 strata = ~tag,  
 data = allSmalls.requested.responded,  
 fpc = ~fpc,  
 weights = ~final.weight.nonresponse)  
#PostStratify by fee code  
ps.dsgn <- postStratify(design = srv.dsgn,  
 strata = ~Fee\_Code,  
 partial = TRUE,  
 population = N.PS)  
  
#IMPUTED WITH VOLUNTEERS  
#filter non-responses  
imputedSmalls.survey <- imputedSmalls.requested.voluntary %>% filter(responded == "y")  
  
fpcID <- allSmalls %>% select(PWSID, fpc)  
imputedSmalls.survey <- left\_join(imputedSmalls.survey, fpcID)  
#Build survey design with volunteer data and imputed values  
srv.dsgn.imputed.volunteers <- svydesign(ids = ~ 0, # no clusters  
 strata = ~tag,  
 data = imputedSmalls.survey,  
 fpc = ~fpc,  
 weights = ~base.weight) #notice that base weights are used isntead of nonresponse adjusted as they dramatically overestimate  
  
#postStratify with volunteer data and imputed values  
ps.dsgn.imputed.volunteers <- postStratify(design = srv.dsgn.imputed.volunteers,  
 strata = ~Fee\_Code,  
 partial = TRUE,  
 population = N.PS)

#if multiple post-strata are used, check that weights are calibrated  
svytotal(~interaction(Fee\_Code, tag), ps.dsgn)

## total SE  
## interaction(Fee\_Code, tag)C1.Bin A 0.0000 0.0000  
## interaction(Fee\_Code, tag)DAVCL.Bin A 0.0000 0.0000  
## interaction(Fee\_Code, tag)DAVCS.Bin A 238.0000 0.0000  
## interaction(Fee\_Code, tag)SC.Bin A 1697.5278 1.7103  
## interaction(Fee\_Code, tag)C1.Bin B 174.4956 4.7611  
## interaction(Fee\_Code, tag)DAVCL.Bin B 88.3369 3.1696  
## interaction(Fee\_Code, tag)DAVCS.Bin B 0.0000 0.0000  
## interaction(Fee\_Code, tag)SC.Bin B 2.4722 1.7103  
## interaction(Fee\_Code, tag)C1.Bin C 89.1467 3.2598  
## interaction(Fee\_Code, tag)DAVCL.Bin C 25.5965 2.7052  
## interaction(Fee\_Code, tag)DAVCS.Bin C 0.0000 0.0000  
## interaction(Fee\_Code, tag)SC.Bin C 0.0000 0.0000  
## interaction(Fee\_Code, tag)C1.Bin D 68.3577 2.2319  
## interaction(Fee\_Code, tag)DAVCL.Bin D 11.0665 1.2730  
## interaction(Fee\_Code, tag)DAVCS.Bin D 0.0000 0.0000  
## interaction(Fee\_Code, tag)SC.Bin D 0.0000 0.0000

Let’s compare the weights calculated from post-stratification with the non-reponse-adjusted weights and the volunteer imputed survey weights.

#examine new weights  
rbind(summary(weights(srv.dsgn)), summary(weights(ps.dsgn)), summary(weights(ps.dsgn.imputed.volunteers)))

## Min. 1st Qu. Median Mean 3rd Qu. Max.  
## [1,] 1.323129 1.530179 2.521859 6.249337 11.03474 25.18421  
## [2,] 1.383315 1.682014 2.657070 6.352785 10.73382 24.79450  
## [3,] 1.261676 1.733332 2.610617 5.757212 10.22688 13.22222

We can see that post-stratification by fee code alone did not alter the weights significantly. This is not surprising due to the original survey design. However, significantly smaller weights are avilable for the post-stratified imputed survey set with volunteers (line 3).

The estimated proportion of fee codes and their their SEs are reported below for the post-stratified.

svytotal(~Fee\_Code, ps.dsgn, na.rm = TRUE)

## total SE  
## Fee\_CodeC1 332 0  
## Fee\_CodeDAVCL 125 0  
## Fee\_CodeDAVCS 238 0  
## Fee\_CodeSC 1700 0

Coefficients of variation produced by the post-stratification are provided below.

cv(svytotal(~Fee\_Code, ps.dsgn, na.rm = TRUE))

## Fee\_CodeC1 Fee\_CodeDAVCL Fee\_CodeDAVCS   
## 0.00000000000000001404514 0.00000000000000003453847 0.00000000000000005681339   
## Fee\_CodeSC   
## 0.00000000000000004320784

If we compare these values to those in the original survey design (without post-stratification), we can see the differences in totals and standard errors.

svytotal(~Fee\_Code, srv.dsgn, na.rm = TRUE)

## total SE  
## Fee\_CodeC1 302.03 9.0372  
## Fee\_CodeDAVCL 119.56 8.4429  
## Fee\_CodeDAVCS 186.75 44.4154  
## Fee\_CodeSC 1747.66 61.8026

cv(svytotal(~Fee\_Code, srv.dsgn, na.rm = TRUE))

## Fee\_CodeC1 Fee\_CodeDAVCL Fee\_CodeDAVCS Fee\_CodeSC   
## 0.02992148 0.07061537 0.23783391 0.03536308

Since we also post-stratified the imputed data, we can view the total estimaes as well.

svytotal(~Fee\_Code, ps.dsgn.imputed.volunteers, na.rm = TRUE)

## total SE  
## Fee\_CodeC1 332 0  
## Fee\_CodeDAVCL 125 0  
## Fee\_CodeDAVCS 238 0  
## Fee\_CodeSC 1700 0

Let’s see how the change in weights changes means for response variables.

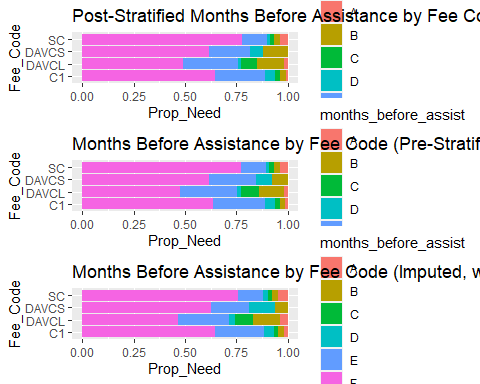
#calcualte unweighted means  
srv.mean.unweighted <- svymean(~cash\_reserve\_total + cash\_reserve\_restricted + cash\_reserve\_unrestricted +  
 months\_before\_assist\_num + delinquent\_num\_acc +   
 delinquent\_amount\_dollars,   
 srv.dsgn.unweighted, na.rm = TRUE)  
srv.mean.unweighted <- data.frame(srv.mean.unweighted)  
#reassign names  
srv.mean.unweighted <- setNames(cbind(rownames(srv.mean.unweighted), srv.mean.unweighted, row.names = NULL),  
 c("variable", "srv.Mean.unweighted", "srv.SE.unweighted"))   
  
#calcualte un-adjusted weights means  
srv.mean.unadjusted <- svymean(~cash\_reserve\_total + cash\_reserve\_restricted + cash\_reserve\_unrestricted +  
 months\_before\_assist\_num + delinquent\_num\_acc +   
 delinquent\_amount\_dollars,   
 srv.dsgn.unadjusted, na.rm = TRUE)  
srv.mean.unadjusted <- data.frame(srv.mean.unadjusted)  
#reassign names  
srv.mean.unadjusted <- setNames(cbind(rownames(srv.mean.unadjusted), srv.mean.unadjusted, row.names = NULL),  
 c("variable", "srv.Mean.unadjusted", "srv.SE.unadjusted"))   
  
#calcualte survey means  
srv.mean <- svymean(~cash\_reserve\_total + cash\_reserve\_restricted + cash\_reserve\_unrestricted +  
 months\_before\_assist\_num + delinquent\_num\_acc +   
 delinquent\_amount\_dollars,   
 srv.dsgn, na.rm = TRUE)  
srv.mean <- data.frame(srv.mean)  
#reassign names  
srv.mean <- setNames(cbind(rownames(srv.mean), srv.mean, row.names = NULL),  
 c("variable", "srv.Mean", "srv.SE"))   
#calcualte post-stratification means  
post.strat.mean <- svymean(~cash\_reserve\_total + cash\_reserve\_restricted +  
 cash\_reserve\_unrestricted +  
 months\_before\_assist\_num + delinquent\_num\_acc +   
 delinquent\_amount\_dollars,  
 ps.dsgn, na.rm = TRUE)  
  
post.strat.mean <- data.frame(post.strat.mean)  
post.strat.mean <- setNames(cbind(rownames(post.strat.mean), post.strat.mean, row.names = NULL),  
 c("variable", "post.strat.Mean", "post.strate.SE"))  
  
#calcualte post-stratified means for IMPUTED data  
srv.mean.imputed.volunteers <- svymean(~cash\_reserve\_total + cash\_reserve\_restricted + cash\_reserve\_unrestricted  
 +  
 months\_before\_assist\_num + delinquent\_num\_acc +   
 delinquent\_amount\_dollars,   
 srv.dsgn.imputed.volunteers, na.rm = TRUE)  
srv.mean.imputed.volunteers <- data.frame(srv.mean.imputed.volunteers)  
#reassign names  
srv.mean.imputed.volunteers <- setNames(cbind(rownames(srv.mean.imputed.volunteers), srv.mean.imputed.volunteers, row.names = NULL),  
 c("variable", "srv.mean.imputed.volunteers", "srv.SE.imputed.volunteers"))   
  
  
#join tables  
smallsSurveySummary <- left\_join(left\_join(left\_join(left\_join(srv.mean,post.strat.mean, by = "variable"), srv.mean.unadjusted),srv.mean.unweighted), srv.mean.imputed.volunteers)  
#save  
write.csv(smallsSurveySummary,  
 file = "R/output/smallsSurveySummary.csv")  
#print  
smallsSurveySummary

## variable srv.Mean srv.SE post.strat.Mean  
## 1 cash\_reserve\_total 2598181.106343 357593.5158799 2777232.253704  
## 2 cash\_reserve\_restricted 1174506.502467 176728.1797411 1247737.213428  
## 3 cash\_reserve\_unrestricted 1423099.105766 190613.0910411 1528903.078129  
## 4 months\_before\_assist\_num 5.469616 0.1090541 5.466149  
## 5 delinquent\_num\_acc 116.842693 12.3033905 123.803254  
## 6 delinquent\_amount\_dollars 39570.682922 3948.8522954 41953.490575  
## post.strate.SE srv.Mean.unadjusted srv.SE.unadjusted srv.Mean.unweighted  
## 1 379886.4581813 2885224.336375 376400.0635541 6253564.708  
## 2 187778.5011243 1294389.077525 188058.5944320 2717783.336  
## 3 202046.8861951 1590226.291165 199373.4483943 3534690.974  
## 4 0.1052986 5.447426 0.1128974 5.250  
## 5 12.6260560 132.376153 12.7767566 326.859  
## 6 4077.5253871 44920.184822 4073.1190765 112497.612  
## srv.SE.unweighted srv.mean.imputed.volunteers srv.SE.imputed.volunteers  
## 1 508603.41043066 2812100.10378 359452.7736417  
## 2 271316.12831840 1208981.87982 173501.5739901  
## 3 274817.51824244 1510729.50966 185246.6808348  
## 4 0.06989041 5.46491 0.1048538  
## 5 21.21705349 125.78647 11.8749504  
## 6 6975.39366256 43041.22889 3790.1854911

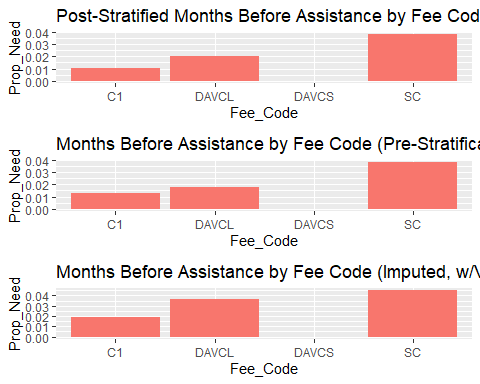
We can see that adjusted for non-response and post-stratifying by fee code has increased the standard error and the mean relative to no adjustment for most variables.

# construct and display a frequency table STRATIFIED  
tab\_fee\_need\_cond <- svytable(~Fee\_Code + months\_before\_assist,  
 design = ps.dsgn) %>%  
 # Add conditional proportions to table  
 as.data.frame() %>%  
 group\_by(Fee\_Code) %>%  
 mutate(n\_Fee\_Code = sum(Freq), Prop\_Need = Freq/n\_Fee\_Code) %>%  
 ungroup()  
# repeate for unweighted  
tab\_fee\_need\_cond\_unwt <- svytable(~Fee\_Code + months\_before\_assist,  
 design = srv.dsgn.unweighted) %>%  
 # Add conditional proportions to table  
 as.data.frame() %>%  
 group\_by(Fee\_Code) %>%  
 mutate(n\_Fee\_Code = sum(Freq), Prop\_Need = Freq/n\_Fee\_Code) %>%  
 ungroup()  
#repeat for volunteers and imputed data  
tab\_fee\_need\_cond\_imputed <- svytable(~Fee\_Code + months\_before\_assist,  
 design = srv.dsgn.imputed.volunteers) %>%  
 # Add conditional proportions to table  
 as.data.frame() %>%  
 group\_by(Fee\_Code) %>%  
 mutate(n\_Fee\_Code = sum(Freq), Prop\_Need = Freq/n\_Fee\_Code) %>%  
 ungroup()

# Create a segmented bar graph of the conditional proportions in table  
p1 <- ggplot(data = tab\_fee\_need\_cond,  
 mapping = aes(x = Fee\_Code, y = Prop\_Need, fill = months\_before\_assist)) +   
 geom\_col() +   
 coord\_flip() +  
 labs(title = "Post-Stratified Months Before Assistance by Fee Code",  
 xlab = "Proportion")  
#before strat  
p2 <- ggplot(data = tab\_fee\_need\_cond\_unwt,  
 mapping = aes(x = Fee\_Code, y = Prop\_Need, fill = months\_before\_assist)) +   
 geom\_col() +   
 coord\_flip() +  
 labs(title = "Months Before Assistance by Fee Code (Pre-Stratification)",  
 xlab = "Proportion")   
# Imputed with volunteers  
p3 <- ggplot(data = tab\_fee\_need\_cond\_imputed,  
 mapping = aes(x = Fee\_Code, y = Prop\_Need, fill = months\_before\_assist)) +   
 geom\_col() +   
 coord\_flip() +  
 labs(title = "Months Before Assistance by Fee Code (Imputed, w/Volunteers)",  
 xlab = "Proportion")   
  
grid.arrange(p1, p2, p3)

 We can see that the weighing does not apprecitably change the results. However there does seem to be a slight change in the number of disadvantaged community larges with immediate need in the volunteer group with imputation.

# Create a segmented bar graph of the conditional proportions in table  
p1 <- tab\_fee\_need\_cond %>%   
 filter(months\_before\_assist == "A") %>%   
 ggplot(mapping = aes(x = Fee\_Code, y = Prop\_Need, fill = months\_before\_assist)) +   
 geom\_col() +  
  
 labs(title = "Post-Stratified Months Before Assistance by Fee Code",  
 xlab = "Proportion")+  
 theme(legend.position = "none")  
#before strat  
p2 <- tab\_fee\_need\_cond\_unwt %>%   
 filter(months\_before\_assist == "A") %>%   
 ggplot(  
 mapping = aes(x = Fee\_Code, y = Prop\_Need, fill = months\_before\_assist)) +   
 geom\_col() +   
 labs(title = "Months Before Assistance by Fee Code (Pre-Stratification)",  
 xlab = "Proportion") +  
 theme(legend.position = "none")  
# Imputed with volunteers  
p3 <- tab\_fee\_need\_cond\_imputed %>%   
 filter(months\_before\_assist == "A") %>%   
 ggplot(  
 mapping = aes(x = Fee\_Code, y = Prop\_Need, fill = months\_before\_assist)) +   
 geom\_col() +   
 labs(title = "Months Before Assistance by Fee Code (Imputed, w/Volunteers)",  
 xlab = "Proportion") +  
 theme(legend.position = "none")  
  
grid.arrange(p1, p2, p3)

 Our survey has now been weighted to adequately represent fee code distributions.

This is repeated below for totals.

#calcualte unweighted totals  
srv.total.unweighted <- svytotal(~cash\_reserve\_total + cash\_reserve\_restricted + cash\_reserve\_unrestricted +  
 months\_before\_assist\_num + delinquent\_num\_acc + `Total number of delinquent residential accounts`+  
 delinquent\_amount\_dollars,   
 srv.dsgn.unweighted, na.rm = TRUE)  
srv.total.unweighted <- data.frame(srv.total.unweighted)  
#reassign names  
srv.total.unweighted <- setNames(cbind(rownames(srv.total.unweighted), srv.total.unweighted, row.names = NULL),  
 c("variable", "srv.total.unweighted", "srv.SE.unweighted"))   
  
#calcualte un-adjusted weights totals  
srv.total.unadjusted <- svytotal(~cash\_reserve\_total + cash\_reserve\_restricted + cash\_reserve\_unrestricted +  
 months\_before\_assist\_num + delinquent\_num\_acc + `Total number of delinquent residential accounts`+  
 delinquent\_amount\_dollars,   
 srv.dsgn.unadjusted, na.rm = TRUE)  
srv.total.unadjusted <- data.frame(srv.total.unadjusted)  
#reassign names  
srv.total.unadjusted <- setNames(cbind(rownames(srv.total.unadjusted), srv.total.unadjusted, row.names = NULL),  
 c("variable", "srv.total.unadjusted", "srv.SE.unadjusted"))   
  
#calcualte survey totals  
srv.total <- svytotal(~cash\_reserve\_total + cash\_reserve\_restricted + cash\_reserve\_unrestricted +  
 months\_before\_assist\_num + delinquent\_num\_acc + `Total number of delinquent residential accounts`+  
 delinquent\_amount\_dollars,   
 srv.dsgn, na.rm = TRUE)  
srv.total <- data.frame(srv.total)  
#reassign names  
srv.total <- setNames(cbind(rownames(srv.total), srv.total, row.names = NULL),  
 c("variable", "srv.total", "srv.SE"))   
#calcualte post-stratification totals  
post.strat.total <- svytotal(~cash\_reserve\_total + cash\_reserve\_restricted +  
 cash\_reserve\_unrestricted +  
 months\_before\_assist\_num + delinquent\_num\_acc + `Total number of delinquent residential accounts`+  
 delinquent\_amount\_dollars,  
 ps.dsgn, na.rm = TRUE)  
  
post.strat.total <- data.frame(post.strat.total)  
post.strat.total <- setNames(cbind(rownames(post.strat.total), post.strat.total, row.names = NULL),  
 c("variable", "post.strat.total", "post.strate.SE"))  
  
#calcualte post-stratified totals for IMPUTED data  
srv.total.imputed.volunteers <- svytotal(~cash\_reserve\_total + cash\_reserve\_restricted + cash\_reserve\_unrestricted  
 +  
 months\_before\_assist\_num + delinquent\_num\_acc + `Total number of delinquent residential accounts`+  
 delinquent\_amount\_dollars,   
 srv.dsgn.imputed.volunteers, na.rm = TRUE)  
srv.total.imputed.volunteers <- data.frame(srv.total.imputed.volunteers)  
#reassign names  
srv.total.imputed.volunteers <- setNames(cbind(rownames(srv.total.imputed.volunteers), srv.total.imputed.volunteers, row.names = NULL),  
 c("variable", "srv.total.imputed.volunteers", "srv.SE.imputed.volunteers"))   
  
  
#join tables  
smallsSurveySummaryTotal <- left\_join(left\_join(left\_join(left\_join(srv.total,post.strat.total, by = "variable"), srv.total.unadjusted),srv.total.unweighted), srv.total.imputed.volunteers)  
#save  
write.csv(smallsSurveySummaryTotal,  
 file = "R/output/smallsSurveySummaryTotal.csv")  
#print  
smallsSurveySummaryTotal

## variable srv.total  
## 1 cash\_reserve\_total 1696718846.203  
## 2 cash\_reserve\_restricted 793446408.793  
## 3 cash\_reserve\_unrestricted 902842637.811  
## 4 months\_before\_assist\_num 3638.909  
## 5 delinquent\_num\_acc 85506.583  
## 6 `Total number of delinquent residential accounts` 77066.167  
## 7 delinquent\_amount\_dollars 28625705.575  
## srv.SE post.strat.total post.strate.SE srv.total.unadjusted  
## 1 179287285.992 1810899162.09 185849053.841 1359976481.128  
## 2 97412620.342 840982565.18 100392911.382 631638687.735  
## 3 90866508.269 969467325.58 95224520.185 728010781.860  
## 4 370.148 3691.72 366.440 2649.762  
## 5 5835.251 92030.31 6252.176 69656.106  
## 6 5804.327 82963.12 6239.526 62880.692  
## 7 1724883.066 30833307.14 1863912.640 23366632.085  
## srv.SE.unadjusted srv.total.unweighted srv.SE.unweighted  
## 1 141397861.7083 907435521 76131986.1025  
## 2 76392643.0986 414349770 41700676.7255  
## 3 72194334.9260 492915648 39391577.3134  
## 4 247.3404 772 36.6245  
## 5 4748.6274 49946 3127.4455  
## 6 4727.1233 44604 3125.9529  
## 7 1410070.5602 16993366 1010429.4772  
## srv.total.imputed.volunteers srv.SE.imputed.volunteers  
## 1 1442455890.909 155864799.0347  
## 2 642055932.601 76525148.5961  
## 3 746486978.234 72378104.2543  
## 4 2909.147 265.1186  
## 5 70856.214 4768.0997  
## 6 63686.095 4744.0505  
## 7 24361293.917 1489443.4255

Just for debt.

#calcualte unweighted total debt  
srv.total.unweighted.debt <- as.data.frame(svytotal(~delinquent\_amount\_dollars,   
 srv.dsgn.unweighted, na.rm = TRUE))  
#reassign names  
srv.total.unweighted.debt <- setNames(cbind(rownames(srv.total.unweighted.debt), srv.total.unweighted.debt, row.names = NULL),  
 c("variable", "srv.total.unweighted", "srv.SE.unweighted"))   
  
#calcualte un-adjusted weights total debt  
srv.total.unadjusted.debt <- as.data.frame(svytotal(~delinquent\_amount\_dollars,   
 srv.dsgn.unadjusted, na.rm = TRUE))  
#reassign names  
srv.total.unadjusted.debt <- setNames(cbind(rownames(srv.total.unadjusted.debt), srv.total.unadjusted.debt, row.names = NULL),  
 c("variable", "srv.total.unadjusted", "srv.SE.unadjusted"))   
  
#calcualte survey total debt  
srv.total.debt <- as.data.frame(svytotal(~delinquent\_amount\_dollars,   
 srv.dsgn, na.rm = TRUE))  
#reassign names  
srv.total.debt <- setNames(cbind(rownames(srv.total.debt), srv.total.debt, row.names = NULL),  
 c("variable", "srv.total", "srv.SE"))   
#calcualte post-stratification totals  
post.strat.total.debt <- as.data.frame(svytotal(~delinquent\_amount\_dollars,  
 ps.dsgn, na.rm = TRUE))  
  
post.strat.total.debt <- setNames(cbind(rownames(post.strat.total.debt), post.strat.total.debt, row.names = NULL),  
 c("variable", "post.strat.total", "post.strate.SE"))  
  
#calcualte post-stratified totals for IMPUTED data  
srv.total.imputed.volunteers.debt <- as.data.frame(svytotal(~delinquent\_amount\_dollars,   
 srv.dsgn.imputed.volunteers, na.rm = TRUE))  
  
#reassign names  
srv.total.imputed.volunteers.debt <- setNames(cbind(rownames(srv.total.imputed.volunteers.debt), srv.total.imputed.volunteers.debt, row.names = NULL),  
 c("variable", "srv.total.imputed.volunteers", "srv.SE.imputed.volunteers"))   
  
  
#join tables  
smallsSurveySummaryTotalDebt <- left\_join(left\_join(left\_join(left\_join(srv.total.debt,post.strat.total.debt, by = "variable"), srv.total.unadjusted.debt),srv.total.unweighted.debt), srv.total.imputed.volunteers.debt)  
#save  
write.csv(smallsSurveySummaryTotalDebt,  
 file = "R/output/smallsSurveySummaryTotalDebt.csv")  
#print  
smallsSurveySummaryTotalDebt

## variable srv.total srv.SE post.strat.total post.strate.SE  
## 1 delinquent\_amount\_dollars 53566965 2305146 57595916 2470236  
## srv.total.unadjusted srv.SE.unadjusted srv.total.unweighted srv.SE.unweighted  
## 1 42872920 1728195 29264025 1130439  
## srv.total.imputed.volunteers srv.SE.imputed.volunteers  
## 1 51801880 3616318

Just for delinquent number of accouunts

#calcualte unweighted total accounts  
srv.total.unweighted.accounts <- as.data.frame(svytotal(~`Total number of delinquent residential accounts`,   
 srv.dsgn.unweighted, na.rm = TRUE))  
#reassign names  
srv.total.unweighted.accounts <- setNames(cbind(rownames(srv.total.unweighted.accounts), srv.total.unweighted.accounts, row.names = NULL),  
 c("variable", "srv.total.unweighted", "srv.SE.unweighted"))   
  
#calcualte un-adjusted weights total accounts  
srv.total.unadjusted.accounts <- as.data.frame(svytotal(~`Total number of delinquent residential accounts`,   
 srv.dsgn.unadjusted, na.rm = TRUE))  
#reassign names  
srv.total.unadjusted.accounts <- setNames(cbind(rownames(srv.total.unadjusted.accounts), srv.total.unadjusted.accounts, row.names = NULL),  
 c("variable", "srv.total.unadjusted", "srv.SE.unadjusted"))   
  
#calcualte survey total accounts  
srv.total.accounts <- as.data.frame(svytotal(~`Total number of delinquent residential accounts`,   
 srv.dsgn, na.rm = TRUE))  
#reassign names  
srv.total.accounts <- setNames(cbind(rownames(srv.total.accounts), srv.total.accounts, row.names = NULL),  
 c("variable", "srv.total", "srv.SE"))   
#calcualte post-stratification totals  
post.strat.total.accounts <- as.data.frame(svytotal(~`Total number of delinquent residential accounts`,  
 ps.dsgn, na.rm = TRUE))  
  
post.strat.total.accounts <- setNames(cbind(rownames(post.strat.total.accounts), post.strat.total.accounts, row.names = NULL),  
 c("variable", "post.strat.total", "post.strate.SE"))  
  
#calcualte post-stratified totals for IMPUTED data  
srv.total.imputed.volunteers.accounts <- as.data.frame(svytotal(~`Total number of delinquent residential accounts`,   
 srv.dsgn.imputed.volunteers, na.rm = TRUE))  
  
#reassign names  
srv.total.imputed.volunteers.accounts <- setNames(cbind(rownames(srv.total.imputed.volunteers.accounts), srv.total.imputed.volunteers.accounts, row.names = NULL),  
 c("variable", "srv.total.imputed.volunteers", "srv.SE.imputed.volunteers"))   
  
  
#join tables  
smallsSurveySummaryTotalaccounts <- left\_join(left\_join(left\_join(left\_join(srv.total.accounts,post.strat.total.accounts, by = "variable"), srv.total.unadjusted.accounts),srv.total.unweighted.accounts), srv.total.imputed.volunteers.accounts)  
#save  
write.csv(smallsSurveySummaryTotalaccounts,  
 file = "R/output/smallsSurveySummaryTotalaccounts.csv")  
#print  
smallsSurveySummaryTotalaccounts

## variable srv.total srv.SE  
## 1 `Total number of delinquent residential accounts` 135911.7 6877.058  
## post.strat.total post.strate.SE srv.total.unadjusted srv.SE.unadjusted  
## 1 147065.7 7529.182 109917.2 5517.409  
## srv.total.unweighted srv.SE.unweighted srv.total.imputed.volunteers  
## 1 74194 3324.726 116612.6  
## srv.SE.imputed.volunteers  
## 1 5723.055

Now that we have our adjusted weights, let’s extract them and bind to the dataframes.

#bind post-stratified weights to all smalls (no volunteers)  
allSmalls.requested.responded <- cbind(allSmalls.requested.responded, weights(ps.dsgn))  
#bind post-stratified weights to imputed survey with volunteers  
imputedSmalls.survey <- cbind(imputedSmalls.survey, weights(ps.dsgn.imputed.volunteers))  
  
#Save to csv  
# write.csv(x = allSmalls.requested.responded %>%   
# select(PWSID,`weights(ps.dsgn)`),  
# file = "Datasets/noVolunteerSurveyWEIGHTS.csv"  
# )  
  
#Save to csv  
# write.csv(x = imputedSmalls.survey %>%   
# select(PWSID,`weights(ps.dsgn.imputed.volunteers)`),  
# file = "Datasets/fullSurveyWEIGHTS.csv")  
#Save to csv  
write.csv(x = imputedSmalls.survey,  
 file = "Datasets/fullSurveyIMPUTED.csv")

#### Large systems

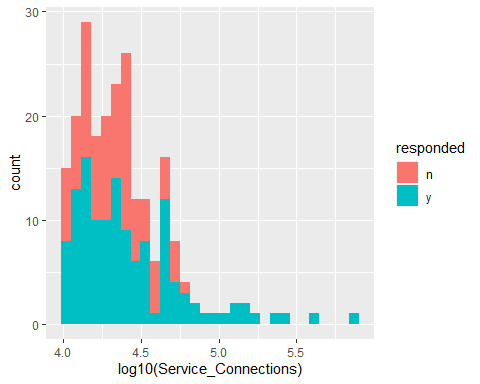
The above steps are repeated for large systems, using service connections asa proxy for representativeness. Jenk’s Natural Breaks are used to bin systems naturally by service connections for calibration.

#jenk's breaks for larges  
#determine mean and stdev  
allLarges %>%   
 summarize(mean = mean(Service\_Connections),  
 var = var(Service\_Connections))

## mean var  
## 1 34878.89 3635566334

Above we can see the mean and variance of service connections for large systems. This is visualized as histrograms.

allLarges %>%   
 ggplot(aes(x = log10(Service\_Connections), fill = responded)) +  
 geom\_histogram()

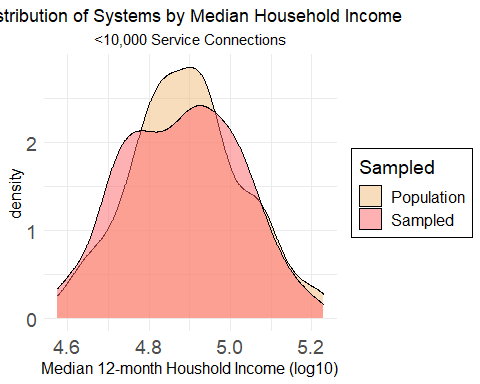


require(wesanderson)

## Warning: package 'wesanderson' was built under R version 4.0.3

allLarges %>%   
 ggplot(aes(x = log10(Median\_12month\_HH\_income), fill = responded)) +  
 geom\_density(alpha = 0.5)+  
 scale\_fill\_manual(values = wes\_palette("GrandBudapest1"),  
 name = "Sampled",  
 labels = c("Population", "Sampled")) +  
 xlab("Median 12-month Houshold Income (log10)") +  
 labs(title = "Distribution of Systems by Median Household Income",  
 subtitle = "<10,000 Service Connections") +  
 theme\_minimal() +  
 theme(  
 legend.box.background = element\_rect(color = "black"),  
 legend.title = element\_text(size = 14),  
 legend.text = element\_text(size = 12),  
 axis.text = element\_text(size = 14),  
 axis.title = element\_text(size = 12),  
 plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))

## Warning: Removed 6 rows containing non-finite values (stat\_density).

 Natural breaks of service connections and median 12-month household income are determined using the Jenk’s method.

#### Natural Breaks ####  
#Determine natural breaks and assign  
breaks <- getJenksBreaks(allLarges$Service\_Connections, 4)  
  
#specify bin labels  
postStrata <- c("A", "B", "C")#, "D")#, "E", "F")#, "G", "H", "I")  
  
#bucket values into bins  
bins <- cut(allLarges$Service\_Connections,  
 breaks = breaks,  
 include.lowest = TRUE,  
 right = FALSE,  
 labels = postStrata)  
  
#inspect bins  
summary(bins)

## A B C   
## 213 8 2

A summary of the number of systems in each bin ( by service connections) is shown above.

#Store group as new column  
allLarges <-as\_tibble(allLarges) %>%   
 mutate(postStrata = case\_when(  
 Service\_Connections >= breaks[1] & Service\_Connections < breaks[2] ~postStrata[1],  
 Service\_Connections >= breaks[2] & Service\_Connections < breaks[3] ~postStrata[2],  
 Service\_Connections >= breaks[3] & Service\_Connections <= breaks[4] ~postStrata[3],  
 ))  
  
#tag is character vector, so convert to factor  
allLarges$postStrata <- factor(allLarges$postStrata,  
 ordered = FALSE)  
breaks

## [1] 10008 105731 389835 709135

The discrete breaks for service connections is above. These steps are repeated for median 12-month household income below.

#### Natural Breaks ####  
#Determine natural breaks and assign  
breaks <- getJenksBreaks(allLarges$Median\_12month\_HH\_income, 4)  
  
#specify bin labels  
postStrata\_income <- c("A", "B", "C")#, "D")#, "E", "F")#, "G", "H", "I")  
  
#bucket values into bins  
bins <- cut(allLarges$Median\_12month\_HH\_income,  
 breaks = breaks,  
 include.lowest = TRUE,  
 right = FALSE,  
 labels = postStrata\_income)  
  
#inspect bins  
summary(bins)

## A B C NA's   
## 92 81 44 6

A summary of the number of systems in each bin (by median 12-month household income) is shown above.

#Store group as new column  
allLarges <-as\_tibble(allLarges) %>%   
 mutate(postStrata\_income = case\_when(  
 Median\_12month\_HH\_income >= breaks[1] & Median\_12month\_HH\_income < breaks[2] ~postStrata\_income[1],  
 Median\_12month\_HH\_income >= breaks[2] & Median\_12month\_HH\_income < breaks[3] ~postStrata\_income[2],  
 Median\_12month\_HH\_income >= breaks[3] & Median\_12month\_HH\_income <= breaks[4] ~postStrata\_income[3],  
 ))  
  
#replace missing value with median (B)  
allLarges$postStrata\_income %<>%   
 replace\_na("B")  
  
#tag is character vector, so convert to factor  
allLarges$postStrata\_income <- factor(allLarges$postStrata\_income,  
 ordered = FALSE)  
breaks

## [1] 37597.92 71235.12 103552.52 169636.28

The discrete breaks for the three bins of income are displayed above (in US dollars).

Now that we have defined strata to adjust the data with, calibration can be carried out. These will also be used to calibrate the surveys.

#get population totals for items of interest  
N.PS <- xtabs(~Fee\_Code, data = allLarges)  
N.SC <- xtabs(~postStrata, data = allLarges)  
N.MH <- xtabs(~postStrata\_income, data = allLarges)  
#combined  
N.PS.SC <- xtabs(#~Fee\_Code +   
 ~postStrata + postStrata\_income, data = allLarges)

Before calibration, non-response-adjusted weights will be examined for completeness.

require(survey)  
#separate samples from population  
larges <- allLarges %>%   
 filter(responded == "y")  
  
#examine adjusted weights  
summary(larges$final.weight.nonresponse)

## Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
## 1.229 1.294 1.371 1.409 1.489 1.981 3

It seems that 3 out of 150 weights are missing, likely due to missing data for predictor variables (likely median household income). Since this is a very small percentage of the total, replacement with mean values is reasonable.

#several weights are missing. Replace with average values  
larges$final.weight.nonresponse %<>%   
 replace\_na(1.412023)  
  
#examine adjusted weights  
summary(larges$final.weight.nonresponse)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 1.229 1.295 1.374 1.409 1.482 1.981

Now that there are no missing adjusted weights, calibration may proceed,

#build survey design with no weights  
srv.dsgn.unweighted.larges <- svydesign(ids = ~0,  
 strata = ~tag,  
 data = larges,  
 fpc = ~fpc,  
 weights = ~NULL)  
#build survey design with non-adjusted weights  
srv.dsgn.unadjusted.larges <- svydesign(ids = ~ 0, # no clusters  
 strata = ~tag,  
 data = larges,  
 fpc = ~fpc,  
 weights = ~base.weight)  
  
# #build survey design with adjusted non-response weights  
srv.dsgn.adjusted.larges <- svydesign(ids = ~ 0, # no clusters  
 strata = ~tag,  
 data = larges,  
 fpc = ~fpc,  
 weights = ~final.weight.nonresponse)  
  
#PostStratify by fee code  
ps.dsgn.fee.code <- postStratify(design = srv.dsgn.adjusted.larges,  
 strata = ~Fee\_Code,  
 partial = TRUE,  
 population = N.PS)  
#Post stratify further by service connections  
ps.dsgn.fee.code.SC <- postStratify(design = ps.dsgn.fee.code,  
 strata = ~postStrata,  
 partial = TRUE,  
 population = N.SC)  
#post-stratify by fee code, service connections, and income  
ps.dsgn.larges <- postStratify(design = srv.dsgn.adjusted.larges,  
 strata = #~Fee\_Code +   
 ~postStrata + postStrata\_income,  
 partial = TRUE,  
 population = N.PS.SC)

Weights have now been adjusted for median household income and service connections. The interaction of these weights are seen below.

#if multiple post-strata are used, check that weights are calibrated  
svytotal(~interaction(postStrata, postStrata\_income), ps.dsgn.larges)

## total SE  
## interaction(postStrata, postStrata\_income)A.A 87 0  
## interaction(postStrata, postStrata\_income)B.A 4 0  
## interaction(postStrata, postStrata\_income)C.A 1 0  
## interaction(postStrata, postStrata\_income)A.B 85 0  
## interaction(postStrata, postStrata\_income)B.B 1 0  
## interaction(postStrata, postStrata\_income)C.B 1 0  
## interaction(postStrata, postStrata\_income)A.C 41 0  
## interaction(postStrata, postStrata\_income)B.C 3 0  
## interaction(postStrata, postStrata\_income)C.C 0 0

Let’s compare the weights calculated from post-stratification with the non-reponse-adjusted weights and the volunteer imputed survey weights.

#examine new weights  
weights.summary <- rbind(summary(weights(srv.dsgn.unadjusted.larges)),summary(weights(srv.dsgn.adjusted.larges)),  
 summary(weights(ps.dsgn.fee.code)), summary(weights(ps.dsgn.fee.code.SC)), summary(weights(ps.dsgn.larges)))  
#name rows  
rownames(weights.summary) <- c("Unadjusted", "Non-Response Adjusted", "Post-Stratified (fee codes)", "Post-Stratified(Fee Code/Srv.Conn)", "Post-Stratified (Income/Srv.Conn.)")  
weights.summary

## Min. 1st Qu. Median Mean  
## Unadjusted 1.7421875 1.742188 1.742188 1.742188  
## Non-Response Adjusted 1.2288037 1.294544 1.373847 1.408894  
## Post-Stratified (fee codes) 1.4926195 1.601936 1.694713 1.742188  
## Post-Stratified(Fee Code/Srv.Conn) 0.9531179 1.641447 1.738562 1.742188  
## Post-Stratified (Income/Srv.Conn.) 0.9746389 1.653418 1.755933 1.742188  
## 3rd Qu. Max.  
## Unadjusted 1.742188 1.742188  
## Non-Response Adjusted 1.482180 1.980632  
## Post-Stratified (fee codes) 1.877126 2.404272  
## Post-Stratified(Fee Code/Srv.Conn) 1.927730 2.469088  
## Post-Stratified (Income/Srv.Conn.) 1.862976 2.480353

We can see that post-stratification by fee code alone (second row) did not alter the weights significantly. Further post-stratification by service connection natural breaks and fee codes increased the range of weights, and notably lowered the minimum weights below nominal (demonstrating over-representation in the original sample). Post-stratification by service connections and median 12-month household income is virtually identical as post-stratification by fee code and service connections, albeit slightly right-shifted. Either will do.

The estimated proportion of fee codes and their their SEs are reported below for the post-stratified.

svytotal(~Fee\_Code, ps.dsgn.larges, na.rm = TRUE)

## total SE  
## Fee\_CodeC1 205.252 3.1962  
## Fee\_CodeDAVCL 17.748 3.1962

Coefficients of variation produced by the post-stratification are provided below.

cv(svytotal(~Fee\_Code, ps.dsgn.larges, na.rm = TRUE))

## Fee\_CodeC1 Fee\_CodeDAVCL   
## 0.01557195 0.18008124

If we compare these values to those in the original survey design (without post-stratification), we can see the differences in totals and standard errors.

svytotal(~Fee\_Code, srv.dsgn.unadjusted.larges, na.rm = TRUE)

## total SE  
## Fee\_CodeC1 203.836 3.6199  
## Fee\_CodeDAVCL 19.164 3.6199

A slight decrease in variance is noted with post-stratificaiton.

cv(svytotal(~Fee\_Code, srv.dsgn.unadjusted.larges, na.rm = TRUE))

## Fee\_CodeC1 Fee\_CodeDAVCL   
## 0.01775869 0.18888788

Total delinquent account (US dollars) is an important response variable. The impact of weighting on this extrapolation is explored below.

svytotal(~dollars\_del\_acc\_TotR, ps.dsgn.larges, na.rm = TRUE)

## total SE  
## dollars\_del\_acc\_TotR 731881220 13748553

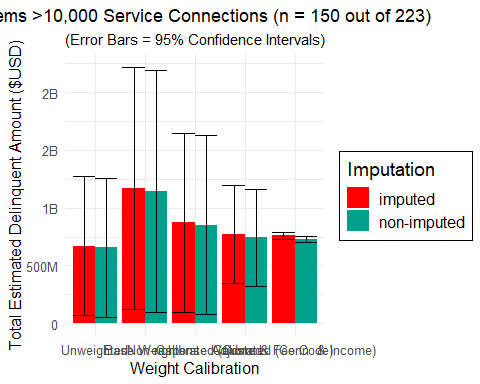
If we compare these values to those in the original survey design (without post-stratification), we can see the differences in totals and standard errors.

svytotal(~dollars\_del\_acc\_TotR, srv.dsgn.unadjusted.larges, na.rm = TRUE)

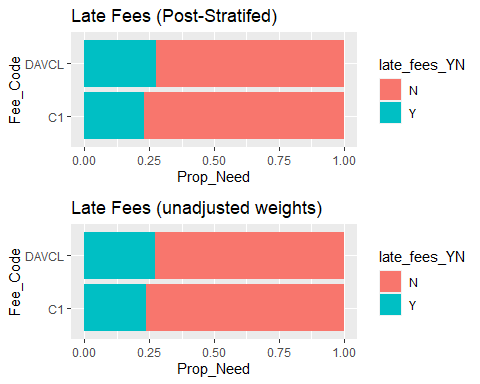
## total SE  
## dollars\_del\_acc\_TotR 1143909455 533991657

Post-stratification by service connections and median household income using Jenk’s Natural Breaks decreased the variance in estimates of statewide large water system delinquent accounts by a factor of 38.97 times! Further, weight adjustment using non-response propensity and calibration decreased the statewide estimate of debt by a factor of 1.56, or 156%. These values are visualized below.

### Do for non-imputed ###  
  
#totals for no weights  
del\_dollars\_larges\_unweighted <- data.frame(svytotal(~dollars\_del\_acc\_TotR, srv.dsgn.unweighted.larges, na.rm = TRUE)) %>%   
 mutate(calibration = "Unweighted") %>%   
 mutate(imputed = "non-imputed")  
  
#totals for unadjusted weights (base weights)  
del\_dollars\_larges\_unadjusted <- data.frame(svytotal(~dollars\_del\_acc\_TotR, srv.dsgn.unadjusted.larges, na.rm = TRUE)) %>%   
 mutate(calibration = "Base Weights") %>%   
 mutate(imputed = "non-imputed")  
  
#totals for unadjusted weights (response propensity)  
del\_dollars\_larges\_adjusted <- data.frame(svytotal(~dollars\_del\_acc\_TotR, srv.dsgn.adjusted.larges, na.rm = TRUE)) %>%   
 mutate(calibration = "Non-response Adjusted") %>%   
 mutate(imputed = "non-imputed")  
  
#totals for adjusted weights (calibration)  
del\_dollars\_larges\_calibrated\_fee <- data.frame(svytotal(~dollars\_del\_acc\_TotR, ps.dsgn.fee.code.SC, na.rm = TRUE)) %>%   
 mutate(calibration = "Calibrated (Conn. & Fee Code)") %>%   
 mutate(imputed = "non-imputed")  
  
#totals for adjusted weights (calibration)  
del\_dollars\_larges\_calibrated <- data.frame(svytotal(~dollars\_del\_acc\_TotR, ps.dsgn.larges, na.rm = TRUE)) %>%  
 mutate(calibration = "Calibrated (Conn. & Income)") %>%   
 mutate(imputed = "non-imputed")  
  
#### Do for imputed ####  
#totals for no weights  
del\_dollars\_larges\_unweighted\_imputed <- data.frame(svytotal(~dollars\_del\_acc\_TotR.imputed, srv.dsgn.unweighted.larges, na.rm = TRUE)) %>%   
 mutate(calibration = "Unweighted") %>%   
 mutate(imputed = "imputed")%>%   
 rename(c("dollars\_del\_acc\_TotR" = dollars\_del\_acc\_TotR.imputed))  
  
#totals for unadjusted weights (base weights)  
del\_dollars\_larges\_unadjusted\_imputed <- data.frame(svytotal(~dollars\_del\_acc\_TotR.imputed, srv.dsgn.unadjusted.larges, na.rm = TRUE)) %>% mutate(calibration = "Base Weights") %>%   
 mutate(imputed = "imputed")%>%   
 rename(c("dollars\_del\_acc\_TotR" = dollars\_del\_acc\_TotR.imputed))  
  
#totals for unadjusted weights (response propensity)  
del\_dollars\_larges\_adjusted\_imputed <- data.frame(svytotal(~dollars\_del\_acc\_TotR.imputed, srv.dsgn.adjusted.larges, na.rm = TRUE)) %>% mutate(calibration = "Non-response Adjusted") %>%   
 mutate(imputed = "imputed")%>%   
 rename(c("dollars\_del\_acc\_TotR" = dollars\_del\_acc\_TotR.imputed))  
  
#totals for adjusted weights (calibration)  
del\_dollars\_larges\_calibrated\_fee\_imputed <- data.frame(svytotal(~dollars\_del\_acc\_TotR.imputed, ps.dsgn.fee.code.SC, na.rm = TRUE)) %>% mutate(calibration = "Calibrated (Conn. & Fee Code)") %>%   
 mutate(imputed = "imputed")%>%   
 rename(c("dollars\_del\_acc\_TotR" = dollars\_del\_acc\_TotR.imputed))  
  
#totals for adjusted weights (calibration)  
del\_dollars\_larges\_calibrated\_imputed <- data.frame(svytotal(~dollars\_del\_acc\_TotR.imputed, ps.dsgn.larges, na.rm = TRUE)) %>% mutate(calibration = "Calibrated (Conn. & Income)") %>%   
 mutate(imputed = "imputed") %>%   
 rename(c("dollars\_del\_acc\_TotR" = dollars\_del\_acc\_TotR.imputed))  
  
#join dataframes  
del\_dollars\_Larges\_c1 <- rbind(del\_dollars\_larges\_unweighted,   
del\_dollars\_larges\_unadjusted, del\_dollars\_larges\_adjusted, del\_dollars\_larges\_calibrated\_fee, del\_dollars\_larges\_calibrated, del\_dollars\_larges\_unweighted\_imputed,   
del\_dollars\_larges\_unadjusted\_imputed, del\_dollars\_larges\_adjusted\_imputed, del\_dollars\_larges\_calibrated\_fee\_imputed, del\_dollars\_larges\_calibrated\_imputed)  
  
#add error bars  
del\_dollars\_Larges\_comparison <- mutate(del\_dollars\_Larges\_c1,  
 lower = total - 1.96\*dollars\_del\_acc\_TotR, #95% CI  
 upper = total + 1.96\*dollars\_del\_acc\_TotR) %>% #95% CI  
 mutate\_if(is.character, as.factor)  
  
require(wesanderson)  
#Construct a barplot  
del\_dollars\_Larges\_comparison %>%   
 mutate(calibration = fct\_relevel(calibration,  
 "Unweighted", "Base Weights", "Non-response Adjusted", "Calibrated (Conn. & Fee Code)", "Calibrated (Conn. & Income)")) %>% #manually reorder by calibration status  
ggplot(aes(x = calibration, y = total,   
 ymin = lower, ymax = upper, fill = imputed)) +  
 geom\_col(position = "dodge") +  
 scale\_x\_discrete(name = "Weight Calibration")+  
 scale\_fill\_manual(values = wes\_palette("Darjeeling1"),  
 name = "Imputation") +  
 scale\_y\_continuous(labels = scales::label\_number\_si()) +  
 labs(y = "Total Estimated Delinquent Amount ($USD)") +  
 geom\_errorbar(width = 0.9, position = "dodge") +  
 labs(title = "Systems >10,000 Service Connections (n = 150 out of 223)",  
 subtitle = "(Error Bars = 95% Confidence Intervals)") +  
 theme\_minimal() +  
 theme(  
 legend.box.background = element\_rect(color = "black"),  
 legend.title = element\_text(size = 14),  
 legend.text = element\_text(size = 12),  
 axis.text = element\_text(size = 10),  
 axis.title = element\_text(size = 12),  
 plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))

 The impact of weigthing (and adjusting weights using calibration) on both estimate totals and error can be seen above.

# construct and display a frequency table STRATIFIED  
tab\_fee\_need\_cond <- svytable(~Fee\_Code + late\_fees\_YN,  
 design = ps.dsgn.larges) %>%  
 # Add conditional proportions to table  
 as.data.frame() %>%  
 group\_by(Fee\_Code) %>%  
 mutate(n\_Fee\_Code = sum(Freq), Prop\_Need = Freq/n\_Fee\_Code) %>%  
 ungroup()  
# repeate for unadjusted weights  
tab\_fee\_need\_cond\_unwt <- svytable(~Fee\_Code + late\_fees\_YN,  
 design = srv.dsgn.unadjusted.larges) %>%  
 # Add conditional proportions to table  
 as.data.frame() %>%  
 group\_by(Fee\_Code) %>%  
 mutate(n\_Fee\_Code = sum(Freq), Prop\_Need = Freq/n\_Fee\_Code) %>%  
 ungroup()   
# Create a segmented bar graph of the conditional proportions in table  
p1 <- ggplot(data = tab\_fee\_need\_cond,  
 mapping = aes(x = Fee\_Code, y = Prop\_Need, fill = late\_fees\_YN)) +   
 geom\_col() +   
 coord\_flip() +  
 labs(title = "Late Fees (Post-Stratifed)",  
 xlab = "Proportion")  
#before strat  
p2 <- ggplot(data = tab\_fee\_need\_cond\_unwt,  
 mapping = aes(x = Fee\_Code, y = Prop\_Need, fill = late\_fees\_YN)) +   
 geom\_col() +   
 coord\_flip() +  
 labs(title = "Late Fees (unadjusted weights)",  
 xlab = "Proportion")  
  
grid.arrange(p1, p2)



Now to ensure adjustedweights sum up to the entire population from which each population was drawn.

#bind post-stratified weights to larges  
larges <- cbind(larges, weights(ps.dsgn.larges))  
   
larges %<>% rename(c("final.weight" = `weights(ps.dsgn.larges)`))

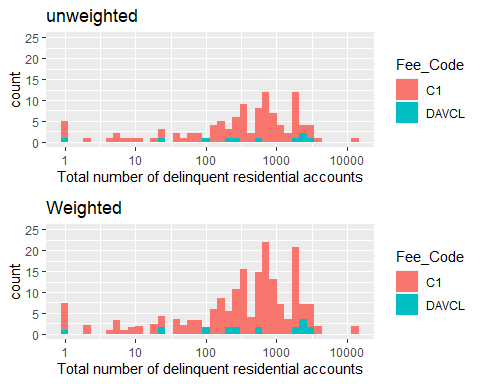
weightedHist <- larges %>%   
 ggplot(aes(x = `Total number of delinquent residential accounts`, weight = final.weight, fill = Fee\_Code)) +  
 geom\_histogram(bins = 40)+  
 scale\_x\_log10() +  
 scale\_y\_continuous(limits = c(0,25)) +  
 labs(title = "Weighted")  
  
unweightedHist <- larges %>%   
 ggplot(aes(x = `Total number of delinquent residential accounts`, fill = Fee\_Code)) +  
 geom\_histogram(bins = 40)+  
 scale\_x\_log10()+  
 scale\_y\_continuous(limits = c(0,25)) +  
 labs(title = "unweighted")  
  
grid.arrange(unweightedHist,weightedHist)

## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Removed 20 rows containing non-finite values (stat\_bin).

## Warning: Transformation introduced infinite values in continuous x-axis

## Warning: Removed 20 rows containing non-finite values (stat\_bin).



Totals summary for larges between weighting schemes are compared below.

#calcualte unweighted totals  
srv.total.unweighted <- svytotal(~dollars\_del\_acc\_TotR + dollars\_del\_dw\_TotR + `Total number of delinquent residential accounts` + num\_del\_acc\_plan +  
 late\_fees\_YN + late\_fees\_dollars,  
 srv.dsgn.unweighted.larges, na.rm = TRUE)  
srv.total.unweighted <- data.frame(srv.total.unweighted)  
#reassign names  
srv.total.unweighted <- setNames(cbind(rownames(srv.total.unweighted), srv.total.unweighted, row.names = NULL),  
 c("variable", "total.unweighted", "total.SE.unweighted"))   
  
#calcualte un-adjusted weights totals  
srv.total.unadjusted <- svytotal(~dollars\_del\_acc\_TotR + dollars\_del\_dw\_TotR + `Total number of delinquent residential accounts` + num\_del\_acc\_plan +  
 late\_fees\_YN + late\_fees\_dollars,   
 srv.dsgn.unadjusted.larges, na.rm = TRUE)  
srv.total.unadjusted <- data.frame(srv.total.unadjusted)  
#reassign names  
srv.total.unadjusted <- setNames(cbind(rownames(srv.total.unadjusted), srv.total.unadjusted, row.names = NULL),  
 c("variable", "total.unadjusted", "SE.unadjusted"))   
  
#calcualte post-stratification totals  
post.strat.total <- svytotal(~dollars\_del\_acc\_TotR + dollars\_del\_dw\_TotR + `Total number of delinquent residential accounts` + num\_del\_acc\_plan +  
 late\_fees\_YN + late\_fees\_dollars,  
 ps.dsgn.larges, na.rm = TRUE)  
  
post.strat.total <- data.frame(post.strat.total)  
post.strat.total <- setNames(cbind(rownames(post.strat.total), post.strat.total, row.names = NULL),  
 c("variable", "post.strat.total", "post.strate.SE"))  
  
#join tables  
largeSurveySummaryTotal <- left\_join(left\_join(srv.total.unweighted,srv.total.unadjusted,  
 by ="variable"),post.strat.total)  
#save  
write.csv(largeSurveySummaryTotal,  
 file = "R/output/largeSurveySummaryTotal.csv")  
#print  
largeSurveySummaryTotal

## variable total.unweighted  
## 1 dollars\_del\_acc\_TotR 25664456  
## 2 dollars\_del\_dw\_TotR 15273472  
## 3 `Total number of delinquent residential accounts` 17856  
## 4 num\_del\_acc\_plan 23261  
## 5 late\_fees\_YNN 5  
## 6 late\_fees\_YNY 14  
## 7 late\_fees\_dollars 2528993  
## total.SE.unweighted total.unadjusted SE.unadjusted post.strat.total  
## 1 4730267.41269 44712294.350391 8241012.758046 39978966.413718  
## 2 2751818.70081 26609252.765691 4794184.142820 23879324.105849  
## 3 4064.11471 31108.500000 7080.449839 31391.513988  
## 4 14849.97251 40525.023438 25871.436478 24865.228955  
## 5 1.43630 8.710938 2.502303 9.398671  
## 6 2.31379 24.390625 4.031056 23.102899  
## 7 952205.03690 4405979.939922 1658919.712728 3436241.777560  
## post.strate.SE  
## 1 6596002.514981  
## 2 4009448.330321  
## 3 7025.233524  
## 4 13390.712378  
## 5 2.692535  
## 6 3.865042  
## 7 1012560.933758

Just for debt.

#calcualte unweighted total debt  
srv.total.unweighted.debt <- as.data.frame(svytotal(~dollars\_del\_acc\_TotR,   
 srv.dsgn.unweighted.larges, na.rm = TRUE))  
#reassign names  
srv.total.unweighted.debt <- setNames(cbind(rownames(srv.total.unweighted.debt), srv.total.unweighted.debt, row.names = NULL),  
 c("variable", "srv.total.unweighted", "srv.SE.unweighted"))   
  
#calcualte un-adjusted weights total debt  
srv.total.unadjusted.debt <- as.data.frame(svytotal(~dollars\_del\_acc\_TotR,   
 srv.dsgn.unadjusted.larges, na.rm = TRUE))  
#reassign names  
srv.total.unadjusted.debt <- setNames(cbind(rownames(srv.total.unadjusted.debt), srv.total.unadjusted.debt, row.names = NULL),  
 c("variable", "srv.total.unadjusted", "srv.SE.unadjusted"))   
  
#calcualte post-stratification totals  
post.strat.total.debt <- as.data.frame(svytotal(~dollars\_del\_acc\_TotR,  
 ps.dsgn.larges, na.rm = TRUE))  
  
post.strat.total.debt <- setNames(cbind(rownames(post.strat.total.debt), post.strat.total.debt, row.names = NULL),  
 c("variable", "post.strat.total", "post.strate.SE"))  
  
#join tables  
largeSurveySummaryTotalDebt <- left\_join(left\_join(srv.total.unadjusted.debt, post.strat.total.debt, by = "variable"),srv.total.unweighted.debt)  
#save  
write.csv(largeSurveySummaryTotalDebt,  
 file = "R/output/largeSurveySummaryTotalDebt.csv")  
#print  
largeSurveySummaryTotalDebt

## variable srv.total.unadjusted srv.SE.unadjusted post.strat.total  
## 1 dollars\_del\_acc\_TotR 1143909455 533991657 731881220  
## post.strate.SE srv.total.unweighted srv.SE.unweighted  
## 1 13748553 656593768 306506422

Just for drinking water debt.

#calcualte unweighted drinking water debt  
srv.total.unweighted.debt <- as.data.frame(svytotal(~dollars\_del\_dw\_TotR,   
 srv.dsgn.unweighted.larges, na.rm = TRUE))  
#reassign names  
srv.total.unweighted.debt <- setNames(cbind(rownames(srv.total.unweighted.debt), srv.total.unweighted.debt, row.names = NULL),  
 c("variable", "srv.total.unweighted", "srv.SE.unweighted"))   
  
#calcualte un-adjusted weights drinking water debt  
srv.total.unadjusted.debt <- as.data.frame(svytotal(~dollars\_del\_dw\_TotR,   
 srv.dsgn.unadjusted.larges, na.rm = TRUE))  
#reassign names  
srv.total.unadjusted.debt <- setNames(cbind(rownames(srv.total.unadjusted.debt), srv.total.unadjusted.debt, row.names = NULL),  
 c("variable", "srv.total.unadjusted", "srv.SE.unadjusted"))   
  
#calcualte post-stratification totals  
post.strat.total.debt <- as.data.frame(svytotal(~dollars\_del\_dw\_TotR,  
 ps.dsgn.larges, na.rm = TRUE))  
  
post.strat.total.debt <- setNames(cbind(rownames(post.strat.total.debt), post.strat.total.debt, row.names = NULL),  
 c("variable", "post.strat.total", "post.strate.SE"))  
  
#join tables  
largeSurveySummaryDWDebt <- left\_join(left\_join(srv.total.unadjusted.debt, post.strat.total.debt, by = "variable"),srv.total.unweighted.debt)  
#save  
write.csv(largeSurveySummaryDWDebt,  
 file = "R/output/largeSurveySummaryDWDebt.csv")  
#print  
largeSurveySummaryDWDebt

## variable srv.total.unadjusted srv.SE.unadjusted post.strat.total  
## 1 dollars\_del\_dw\_TotR 294011783 94561239 199960210  
## post.strate.SE srv.total.unweighted srv.SE.unweighted  
## 1 7230338 168760127 54277303

Now for number of delinquent accounts.

#calcualte unweighted total accounts  
srv.total.unweighted.accounts <- as.data.frame(svytotal(~`Total number of delinquent residential accounts`,   
 srv.dsgn.unweighted.larges, na.rm = TRUE))  
#reassign names  
srv.total.unweighted.accounts <- setNames(cbind(rownames(srv.total.unweighted.accounts), srv.total.unweighted.accounts, row.names = NULL),  
 c("variable", "srv.total.unweighted", "srv.SE.unweighted"))   
  
#calcualte un-adjusted weights total accounts  
srv.total.unadjusted.accounts <- as.data.frame(svytotal(~`Total number of delinquent residential accounts`,   
 srv.dsgn.unadjusted.larges, na.rm = TRUE))  
#reassign names  
srv.total.unadjusted.accounts <- setNames(cbind(rownames(srv.total.unadjusted.accounts), srv.total.unadjusted.accounts, row.names = NULL),  
 c("variable", "srv.total.unadjusted", "srv.SE.unadjusted"))   
  
#calcualte post-stratification totals  
post.strat.total.accounts <- as.data.frame(svytotal(~`Total number of delinquent residential accounts`,  
 ps.dsgn.larges, na.rm = TRUE))  
  
post.strat.total.accounts <- setNames(cbind(rownames(post.strat.total.accounts), post.strat.total.accounts, row.names = NULL),  
 c("variable", "post.strat.total", "post.strate.SE"))  
  
#join tables  
largeSurveySummaryTotalaccounts <- left\_join(left\_join(srv.total.unadjusted.accounts, post.strat.total.accounts, by = "variable"),srv.total.unweighted.accounts)  
#save  
write.csv(largeSurveySummaryTotalaccounts,  
 file = "R/output/largeSurveySummaryTotalaccounts.csv")  
#print  
largeSurveySummaryTotalaccounts

## variable srv.total.unadjusted  
## 1 `Total number of delinquent residential accounts` 166564  
## srv.SE.unadjusted post.strat.total post.strate.SE srv.total.unweighted  
## 1 17584.24 169603 17306.32 95606.27  
## srv.SE.unweighted  
## 1 10093.2

Now that we have our adjusted weights, let’s extract them and bind to the dataframes.

#bind post-stratified weights to all smalls (no volunteers)  
#larges <- cbind(larges, weights(ps.dsgn.larges))  
  
#Save to csv  
write.csv(x = larges %>%   
 select(PWSID, final.weight, base.weight, fpc,voluntary),  
 file = "Datasets/largeSystemWEIGHTS.csv"  
 )

# Summary Statistics

Now that we have our adjusted weights and a complete (imputed) dataset, it’s time to compute summary statistics (totals, means, variances, ratios, and quantiles).Estimates for statewide number of delinquent accounts for smalls and larges are calcualted below.

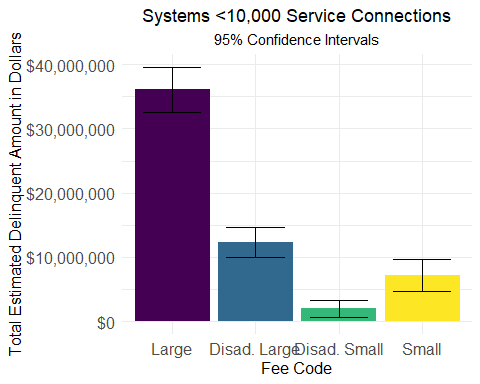
## Smalls

#smalls  
del\_dollars\_smalls <- svyby(~delinquent\_amount\_dollars,   
 by = ~Fee\_Code,   
 design = ps.dsgn,  
 FUN = svytotal,  
 na.rm = TRUE)  
del\_dollars\_smalls

## Fee\_Code delinquent\_amount\_dollars se  
## C1 C1 36068952 1767056.8  
## DAVCL DAVCL 12257939 1190189.0  
## DAVCS DAVCS 2052566 675815.8  
## SC SC 7216459 1269761.2

Totals can be visualized graphically.

#add lower and upper columns  
del\_dollars\_smalls\_col <- mutate(del\_dollars\_smalls,  
 lower = delinquent\_amount\_dollars - 1.96\*se, #95% CI  
 upper = delinquent\_amount\_dollars + 1.96\*se)  
#Construct a barplot  
del\_dollars\_smalls\_col %>%   
ggplot(aes(x = Fee\_Code, y = delinquent\_amount\_dollars,   
 ymin = lower, ymax = upper, fill = Fee\_Code)) +  
 geom\_col() +  
 scale\_fill\_viridis\_d() +  
 scale\_x\_discrete(name = "Fee Code",   
 labels = c("Large",   
 "Disad. Large",  
 "Disad. Small",   
 "Small")) +  
 scale\_y\_continuous(labels = scales::dollar\_format()) +  
 labs(y = "Total Estimated Delinquent Amount in Dollars") +  
 geom\_errorbar(width = 0.7) +  
 labs(title = "Systems <10,000 Service Connections",  
 subtitle = "95% Confidence Intervals") +  
 theme\_minimal() +  
 theme(legend.position = "none",  
 legend.box.background = element\_rect(color = "black"),  
 legend.title = element\_text(size = 14),  
 legend.text = element\_text(size = 12),  
 axis.text = element\_text(size = 12),  
 axis.title = element\_text(size = 12),  
 plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



Above is for smalls. Below is for larges for total dollars of delinquent accouts (including other expenses such as power).

#larges  
dollars\_del\_acc\_TotR <- svyby(~dollars\_del\_acc\_TotR,   
 by = ~Fee\_Code,   
 design = ps.dsgn.larges,  
 FUN = svytotal,  
 na.rm = TRUE)  
dollars\_del\_acc\_TotR

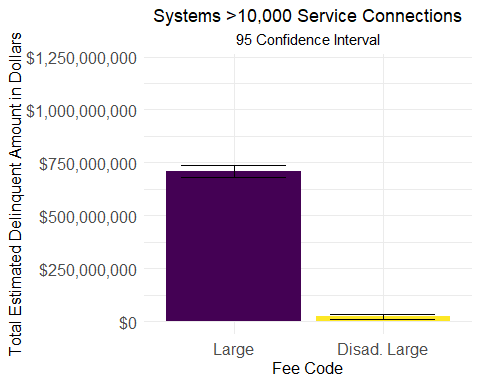
## Fee\_Code dollars\_del\_acc\_TotR se  
## C1 C1 708896034 14098156  
## DAVCL DAVCL 22985187 5981942

Confidence intervals are reported below.

confint(dollars\_del\_acc\_TotR)

## 2.5 % 97.5 %  
## C1 681264156 736527911  
## DAVCL 11260796 34709577

#add lower and upper columns  
dollars\_del\_acc\_TotRs\_col <- mutate(dollars\_del\_acc\_TotR,  
 lower = dollars\_del\_acc\_TotR - 1.96\*se,  
 upper = dollars\_del\_acc\_TotR + 1.96\*se)  
#Construct a barplot  
dollars\_del\_acc\_TotRs\_col %>%   
ggplot(aes(x = Fee\_Code, y = dollars\_del\_acc\_TotR,   
 ymin = lower, ymax = upper, fill = Fee\_Code)) +  
 geom\_col() +  
 scale\_fill\_viridis\_d() +  
 scale\_x\_discrete(name = "Fee Code",   
 labels = c("Large",   
 "Disad. Large")) +  
 scale\_y\_continuous(labels = scales::dollar\_format(),  
 limits = c(0,1200000000)) +  
 labs(y = "Total Estimated Delinquent Amount in Dollars") +  
 geom\_errorbar(width = 0.7) +  
 labs(title = "Systems >10,000 Service Connections",  
 subtitle = "95 Confidence Interval") +  
 theme\_minimal() +  
 theme(legend.position = "none",  
 legend.box.background = element\_rect(color = "black"),  
 legend.title = element\_text(size = 14),  
 legend.text = element\_text(size = 12),  
 axis.text = element\_text(size = 12),  
 axis.title = element\_text(size = 12),  
 plot.title = element\_text(hjust = 0.5),  
 plot.subtitle = element\_text(hjust = 0.5))



Below is for larges for total dollars of delinquent accouts (drinking water alone).

#larges  
s2 <- svyby(~dollars\_del\_dw\_TotR,   
 by = ~Fee\_Code,   
 design = ps.dsgn.larges,  
 FUN = svytotal,  
 na.rm = TRUE)  
s2

## Fee\_Code dollars\_del\_dw\_TotR se  
## C1 C1 190263890 7133447  
## DAVCL DAVCL 9696320 3180524

Again with confidence intervals.

confint(s2, level = 0.95)

## 2.5 % 97.5 %  
## C1 176282591 204245189  
## DAVCL 3462608 15930032

Now simply totals without binning.

#smalls  
svytotal(~delinquent\_amount\_dollars,   
 design = ps.dsgn,  
 na.rm = TRUE)

## total SE  
## delinquent\_amount\_dollars 57595916 2470236

Above is totals for smalls. 57,595,916 +- 2,470,236.

Below is for larges (all expenses)

#larges  
svytotal(~dollars\_del\_acc\_TotR,   
 design = ps.dsgn.larges,  
 na.rm = TRUE)

## total SE  
## dollars\_del\_acc\_TotR 731881220 13748553

And below for just drinking water expenses.

#larges  
svytotal(~dollars\_del\_dw\_TotR,   
 design = ps.dsgn.larges,  
 na.rm = TRUE)

## total SE  
## dollars\_del\_dw\_TotR 199960210 7230338

And for smalls.

All of these data can be viewed as a table.

# construct and display a frequency table STRATIFIED  
tab\_debt\_lrg <- svyby(~dollars\_del\_dw\_TotR,   
 by = ~Fee\_Code,   
 design = ps.dsgn.larges,  
 FUN = svytotal,  
 na.rm = TRUE) %>%  
 as.data.frame()  
# repeate for unadjusted weights  
tab\_debt\_lrg\_unweighted <- svyby(~dollars\_del\_dw\_TotR,   
 by = ~Fee\_Code,   
 design = srv.dsgn.unweighted.larges,  
 FUN = svytotal,  
 na.rm = TRUE) %>%  
 as.data.frame()

# # Create a segmented bar graph of the conditional proportions in table  
# p1 <- ggplot(data = tab\_debt\_cond\_lrg,  
# mapping = aes(x = Fee\_Code, y = Prop\_Need)) +   
# geom\_col() +   
# coord\_flip() +  
# labs(title = "Total Debt (Post-Stratifed)",  
# xlab = "Proportion")  
# #before strat  
# p2 <- ggplot(data = tab\_debt\_cond\_\_lrg\_unwt,  
# mapping = aes(x = Fee\_Code, y = Prop\_Need)) +   
# geom\_col() +   
# coord\_flip() +  
# labs(title = "Total Debt (unadjusted weights)",  
# xlab = "Proportion")  
#   
# grid.arrange(p1, p2)

Let’s ask what the mean cash reserve total is for each of the strata. The mean for each bin is interpreted as the proportion for each category.

# svymean(~cash\_reserve\_total + tag, dclus1)

1. What is the total amount of household water debt accumulated since the beginning of the COVID-19 emergency? [MODERATE, REQUIRES WEIGHTING, COMBINING WITH ALL SYSTEMS]