Lecture 9: Massive, Messy, and Missing Data

Case Study: A Big, Messy Survey, Continued

Setting: A real world marketing analytics project with a big company.

The Challenges:

- Understand everything about their surveys.
- Organize, clean, and process all of their data.
- Develop models for each state of customer engagement for each product.
- Identify the idiosyncratic factors that impact each model.
- Drill down into the important subgroups for each product and state of engagement.
- Develop dynamic reporting tools to summarize all of these results.
- AND do all of this in 10 weeks. (Yikes!)

Our Previous Work

- So far we've done a lot to better understand the surveys, the data, and the organization's challenges in marketing research.

- We created a capacity to analyze the data and create **customized models** for the states of engagement.

- We also created a **dynamic reporting engine** with significant capabilities to display information and build models at a moment's notice.

A Few More Challenges

- In addition to the survey, the team also wanted to integrate **another source of information** with more information about the survey's respondents.

- Both sources of information contain significant amounts of **missing data**

The Expanded Information

- The new information was about each person's **consumer interests**, which a third-party company compiled based upon data about the consumer's purchases.

- The new information was **person-specific**, but it is now merged into the long format. Across a single person's rows (for each product), the values of these new variables are repeated.

- Some work (not shown here) was undertaken to link the new information to the previous file we worked with. The data are structured in **long format**, with each row representing a person's responses for a single product.

The Expanded Data

```r
dat <- fread(input = "Expanded Marketing Data -- with Missingness.csv")
dim(dat)
```

```
[1] 2300000      75
```

```r
names(dat)
```

```
 [1] "id"                      "age"
 [3] "gender"                  "income"
 [5] "region"                  "persona"
 [7] "Product"                 "Awareness"
 [9] "BP_For_Me_0_10"          "BP_Fits_Budget_0_10"
[11] "BP_Tastes_Great_0_10"    "BP_Good_To_Share_0_10"
[13] "BP_Like_Logo_0_10"       "BP_Special_Occasions_0_10"
[15] "BP_Everyday_Snack_0_10"  "BP_Healthy_0_10"
[17] "BP_Delicious_0_10"       "BP_Right_Amount_0_10"
[19] "BP_Relaxing_0_10"        "Consideration"
[21] "Consumption"             "Satisfaction"
[23] "Advocacy"                "food"
[25] "drinks"                  "alcohol"
[27] "gardening"               "outdoors"
[29] "hiking"                  "photography"
[31] "camping"                 "fashion"
[33] "shoes"                   "beauty products"
[35] "toys"                    "games"
[37] "parenting"               "family"
[39] "home decorations"        "architecture"
[41] "art"                     "cartoons"
[43] "animation"               "celebrities"
[45] "movies"                  "television"
[47] "drama"                   "comedy"
[49] "science fiction"         "personal finance"
[51] "sports"                  "cooking"
[53] "barbecue grills"         "health"
[55] "exercise"                "wellness"
[57] "spirituality"            "religion"
[59] "community organizations" "politics"
[61] "news"                    "skiing"
[63] "literature"              "poetry"
[65] "music"                   "musical instruments"
[67] "astronomy"               "space"
[69] "travel"                  "tourism"
[71] "restaurants"             "cars"
[73] "bicycles"                "carpentry"
[75] "knitting"
```

Some Prior Work

```r
id.name <- "id"
age.name <- "age"
gender.name <- "gender"
income.name <- "income"
region.name <- "region"
persona.name <- "persona"

product.name <- "Product"
awareness.name <- "Awareness"
consideration.name <- "Consideration"
consumption.name <- "Consumption"
satisfaction.name <- "Satisfaction"
advocacy.name <- "Advocacy"

bp.pattern <- "BP_"
age.group.name <- "age_group"
income.group.name <- "income_group"
cuts.age <- c(18, 35, 50, 65, 120)
cuts.income <- 1000* c(0, 25, 50, 75, 100, 200)

dat[, eval(age.group.name) := cut2(x = get(age.name), cuts = cuts.age)]
dat[, eval(income.group.name) := cut2(x = get(income.name), cuts = cuts.income)]
dat[, eval(satisfaction.name) := get(satisfaction.name) / 10]

respondent.variables <- c(age.name, income.name, age.group.name, gender.name, income.group.name, region.name, persona.name)
states.of.engagement <- c(awareness.name, consideration.name, consumption.name, satisfaction.name, advocacy.name)
bp.traits <- names(dat)[grep(pattern = bp.pattern, x = names(dat))]
```

Investigating the New Variables

```
new.variables <- names(dat)[!(names(dat) %in% c(id.name,
    product.name, respondent.variables, states.of.engagement,
    bp.traits))]
datatable(data = dat[get(product.name) == get(product.name)[1],
    .SD, .SDcols = new.variables][1:100, ], rownames = FALSE)
```

Show 10 entries

Search:

food	drinks	alcohol	gardening	outdoors	hiking	photography	camping	fashion	shoes
0					0	1			0
					0				0
0					0		0		0
0					1		0		0
0					0		0		0
0					0				0
					0		0		0
					0		0		0
0					0				0
0					0				0

Showing 1 to 10 of 100 entries

Previous 1 2 3 4 5 … 10 Next

What We See

- The new data presents **detailed information** about the consumer's interests across many measures.

- Finding links between **specific interests** and **states of engagement** could be a key to a more refined marketing strategy.

- Knowing that an interest in knitting was associated with advocacy for a product would provide opportunities to decide **which people** to reach out to in advertising campaigns and **where to find them** (e.g. in certain kinds of media, events, geographic regions, etc.).

- The real project I worked on had approximately 500 of these measures.

What We Don’t See

- All of the **missing data**.

- In fact, the earlier data also had substantial amounts of missing data.

- What we saw earlier was a cleaned-up version of the survey’s data. The Expanded Data presented here is more akin to the missingness and complexity the actual data contained.

Missing Data: A Pervasive Problem

- Nearly every large, complex set of data includes some degree of missing data.
- Any important variable with substantial degrees of missing data creates choices about how to handle the situation.
- Every choice has trade-offs.
- There is not always an obvious solution on how to proceed.

Case Study: Titanic Data

Let’s look at some data about the passengers on the ill-fated maiden voyage of the RMS Titanic (1912).

```r
library(PASWR)
data(titanic3)
titanic <- setDT(x = titanic3)
datatable(data = titanic, rownames = FALSE)
```

Show 10 ⬍ entries   Search:

pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked
1st	1	Allen, Miss. Elisabeth Walton	female	29	0	0	24160	211.337494	B5	Southampton
1st	1	Allison, Master. Hudson Trevor	male	0.916700006	1	2	113781	151.550003	C22 C26	Southampton
1st	0	Allison, Miss. Helen Loraine	female	2	1	2	113781	151.550003	C22 C26	Southampton
1st	0	Allison, Mr. Hudson Joshua Crei	male	30	1	2	113781	151.550003	C22 C26	Southampton
1st	0	Allison, Mrs. Hudson J C (Bessi	female	25	1	2	113781	151.550003	C22 C26	Southampton
1st	1	Anderson, Mr. Harry	male	48	0	0	19952	26.5499992	E12	Southampton
1st	1	Andrews, Miss. Kornelia Theodos	female	63	1	0	13502	77.9582977	D7	Southampton
1st	0	Andrews, Mr. Thomas Jr	male	39	0	0	112050	0	A36	Southampton
1st	1	Appleton, Mrs. Edward Dale (Cha	female	53	2	0	11769	51.4791985	C101	Southampton
1st	0	Artagaveytia, Mr. Ramon	male	71	0	0	PC 17609	49.5042		Cherbourg

Showing 1 to 10 of 1,309 entries   Previous 1 2 3 4 5 … 131 Next

Rates of Missingness

```
mean.missing <- function(x) {
    return(mean(is.na(x)))
}
round.numerics <- function(x, digits) {
    if (is.numeric(x)) {
        x <- round(x = x, digits = digits)
    }
    return(x)
}
missingness.rate.titanic <- titanic[, lapply(X = .SD, FUN = "mean.missing")]
datatable(data = missingness.rate.titanic[, lapply(X = .SD,
    FUN = "round.numerics", digits = 2)], rownames = FALSE)
```

Show 10 entries Search:

pclass	survived	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	boat
0	0	0	0	0.2	0	0	0	0	0	0	0

Showing 1 to 1 of 1 entries

Previous 1 Next

Ambiguities: Cabin

By concentrating only on the values coded as **NA**, we miss a number of issues:

- The **cabin** variable contains many **blank values** "".

```
titanic[cabin == "", .N]
```

```
[1] 1014
```

- The blank values could refer to **non-private accommodations** or may indicate missing data.

- Either way, it's probable that **we don't truly have complete information** for this variable.

Ambiguities: Boat

- The **boat** variable also contains many **blank values**:

```
titanic[boat == "", .N]
```

```
[1] 823
```

- Unfortunately, this is not an indicator of missing data. The passengers without a labeled boat were not assigned to one of the limited number of life boats after the Titanic crashed into an iceberg.

- While not labeled, the blank values of **boat** are not truly missing; instead, they would be better represented with a label like **“No lifeboat”**.

Reminder: Data Cleaning

- Missing data is not always coded the way you think it should be. Blank characters, values like **“NA”** or **“N/A”**, or others (e.g. **-1** in otherwise non-negative numeric data) can all be lurking.

- Missing values can also be automatically converted to NA in the reading step. For instance, the **fread** function includes a parameter **na.strings** that allows you to specify a character vector of strings that should be converted to NA values.

- However, using **na.strings** in the reading step **applies to every column** of the data set. If different columns have conflicting conventions, then they must be separately addressed.

Missing Age in the Titanic Data

- A fair number of the passengers did not have a recorded age:

```
titanic[, mean.missing(x = get(age.name))]
```

```
[1] 0.2009167
```

- What should we do about the problem? Ignore it? Try to fill in the values?

Ignoring Missing Data

- **Focusing on what is measured** is one way to be honest about what you have.
- However, what you have **may not be fully representative** of the complete information.
- Meanwhile, missing data reduces the sample size. For instance, **linear regression removes any row with at least one missing value**.

Assumptions About Missing Data

- **Missing Completely At Random (MCAR)**: The missingness of any value occurs completely at random, independent of any other factors.

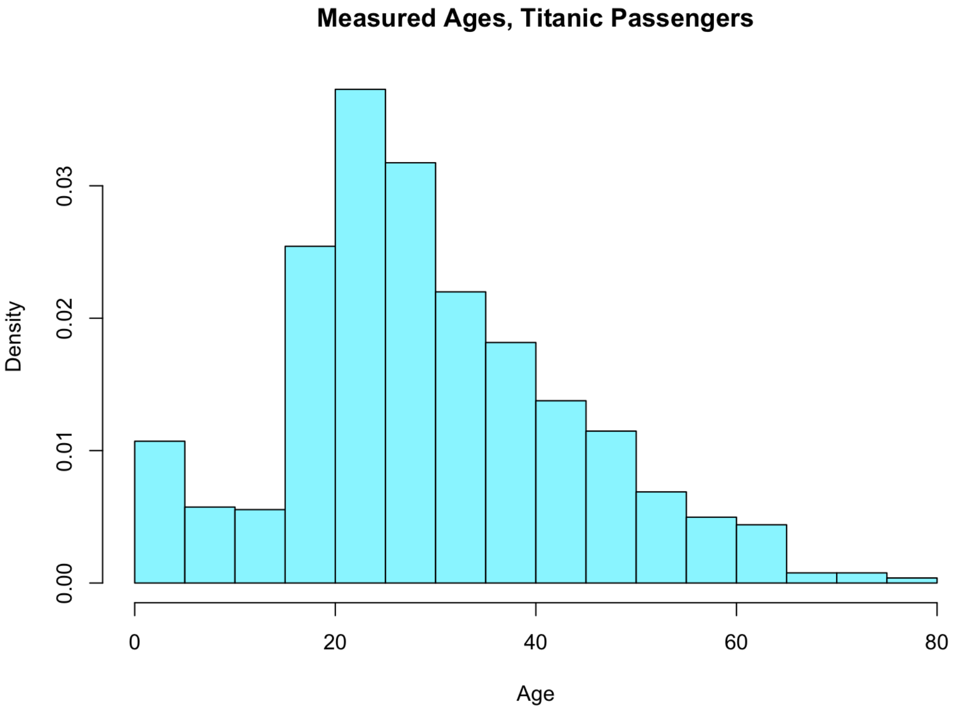
- **Missing at Random (MAR)**: The missingness of a value appears to be random but can be reasonably accounted for by its relationship to the measured variables.

- **Missing Not At Random (MNAR)**: Certain ranges are more likely to be missing, and the remaining data is not representative of the broader population.

Filling in the missing values typically requires at least a MAR assumption if not MCAR. MNAR means that you have very limited insight into what the missing values would actually be.

Histogram: The Measured Ages

```
hist(x = titanic[, get(age.name)], xlab = "Age", main = "Measured Ages, Titanic Passengers",
    freq = FALSE, col = "cadetblue1")
```



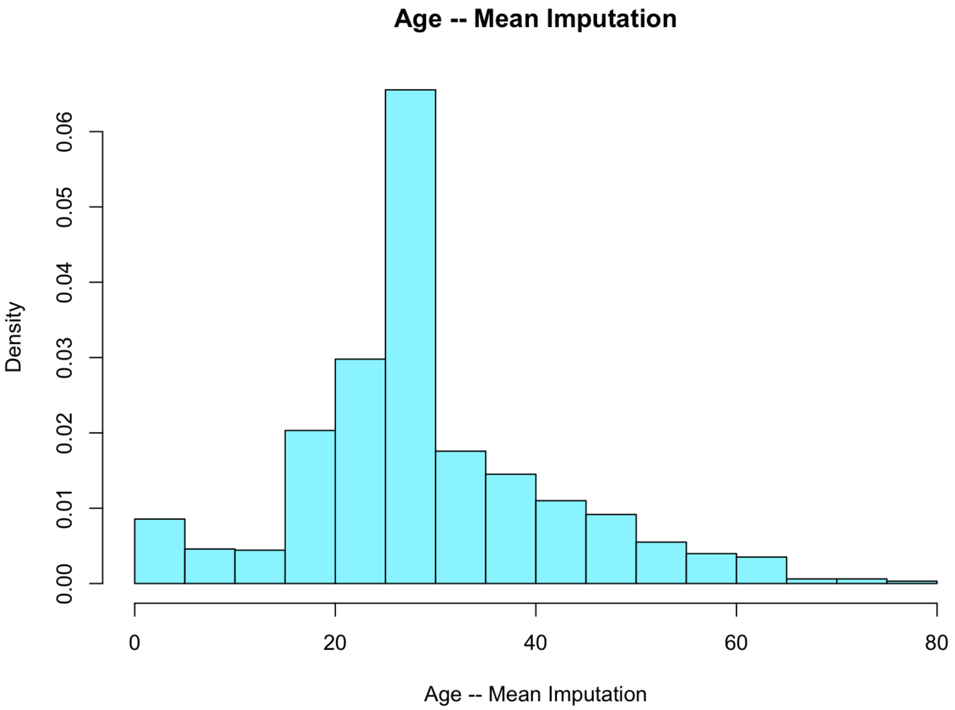
Approaches to Imputing Missing Ages

Imputation is the act of filling in missing values. The question is how best to accomplish this. Here are a few approaches

- **Mean Imputation**: Fill in the average age for all of the missing values.
- **Random Sampling**: Fill each missing value by randomly selecting a measured value.
- **Multivariable Regression**: Use the other measured factors to predict the missing values (the testing set) based on a model from the rows with recorded values (the training set).

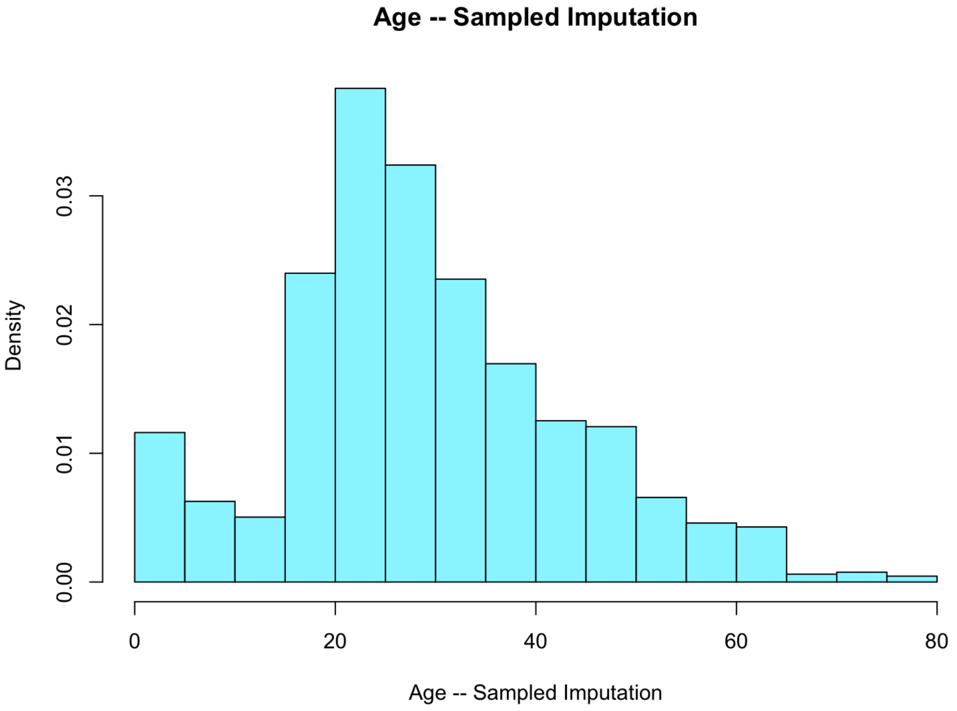
Mean Imputation of Age

```
mean.imputed.age.name <- "age_mean_imputed"
mean.recorded.age <- titanic[, mean(get(age.name), na.rm = TRUE)]
titanic[, eval(mean.imputed.age.name) := get(age.name)]
titanic[is.na(get(age.name)), eval(mean.imputed.age.name) := mean.recorded.age]
hist(x = titanic[, get(mean.imputed.age.name)], xlab = "Age -- Mean Imputation", main = "Age -- Mean Imputation", freq = FALSE, col = "cadetblue1")
```



Imputed Age with Random Sampling

```r
sampled.imputation.age.name <- "age_sampled_imputation"
recorded.ages <- titanic[!is.na(get(age.name)), get(age.name)]
titanic[, eval(sampled.imputation.age.name) := get(age.name)]
number.to.sample <- titanic[is.na(get(age.name)), .N]
titanic[is.na(get(age.name)), eval(sampled.imputation.age.name) := sample(x = recorded.ages, size = number.to.sample, replace = TRUE)]
hist(x = titanic[, get(sampled.imputation.age.name)], xlab = "Age -- Sampled Imputation", main = "Age -- Sampled Imputation", freq = FALSE, col =
  "cadetblue1")
```



Discussion of Approaches

- **Mean Imputation** places significant extra weight on the mean value, which fundamentally changes the distribution of ages.

- **Imputation by Random Sampling** imputes values in a way that maintains the original distribution.

- However, random sampling is **not guaranteed to maintain the relationships with the other variables**:

```
survival.name <- "survived"
surv.age.correlations <- titanic[, lapply(X = .SD, FUN = "cor",
    y = get(survival.name), use = "pairwise.complete.obs"),
    .SDcols = c(age.name, mean.imputed.age.name, sampled.imputation.age.name)]
datatable(data = surv.age.correlations[, lapply(X = .SD,
    FUN = "round.numerics", digits = 3)], rownames = FALSE)
```

Show 10 ⬍ entries   Search: □

age	age_mean_imputed	age_sampled_imputation
-0.056	-0.05	-0.038

Showing 1 to 1 of 1 entries   Previous 1 Next

In this case, imputing the missing ages did not greatly change the correlation with survival. In other cases, that is by no means assured.

Concerns about Imputed Data

- Randomly sampling can lead to cases in which **correlations and regression effects are greatly changed** after imputation.

- Likewise, the interpretations of the study could **hinge upon how you imputed the missing data**.

- In these settings, performing a variety of **sensitivity analyses** can help you understand the implications of your selected method of imputation.

Multivariable Imputation

- Imputing the missing values can become a **machine learning problem**. You can build complex models to better impute the missing values.
- Meanwhile, some of the other variables used as inputs to predict one missing column of missing data **may themselves have missing values**.
- In these settings, a whole sequence of imputation models may be necessary. One approach to doing so is using **MICE**: Multiple Imputation by Chained Equations.

(Better understanding the theory and application of multivariable methods of imputation would be the topic for a more advanced class in statistics.)

Missing Data: Nuissance or Obstacle?

- Small amounts of missing data do not substantially change your analysis.
- Many regression methods and machine learning techniques **only use the rows with complete information**.
- Missing data reduces the **effective** sample size of the data set.

Missing Data Piles Up

- Suppose a data set has **K** variables, and each measurement of the kth variable is missing with probability p_k.

- If the missing values across the variables are independent and missing at random, then the probability that a row will include at least one missing value is:

P(Incomplete Row) = $1 - \prod_{k=1}^{K}(1 - p_k).$

Let's take a look at how this scales with the number of variables:

Probability of an Incomplete Row of Data

Here is some code to calculate the probability of an incomplete row and graph the results as a function of the number of variables in the data set.

```r
incompleteness.rate <- function(prob, k) {
    incompleteness.probability <- 1 - (1 - prob)^k
    return(incompleteness.probability)
}
```

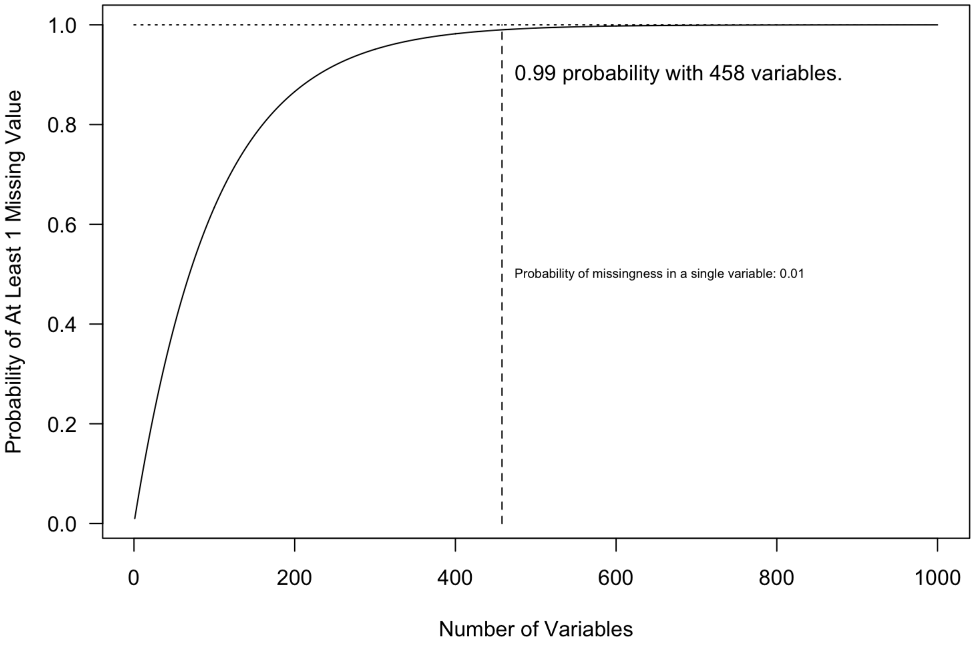
This assumes that each variable has the same rate of missingness and that the values are missing independently.

Graphing Incompleteness Probabilities

```
display.incompleteness.plot <- function(prob, max.variables) {
    rates <- incompleteness.rate(prob = prob, k = 1:max.variables)
    plot(x = 1:max.variables, y = rates, type = "l", xlab = "Number of Variables",
        ylab = "Probability of At Least 1 Missing Value",
        las = 1)
    w = which.min(x = abs(rates - 0.99))
    lines(x = c(w, w), y = c(0, 1), lty = 2)
    lines(x = c(0, max.variables), y = c(1, 1), lty = 3)
    text(x = w, y = 0.9, labels = sprintf("%.2f probability with %d variables.",
        rates[w], w), pos = 4)
    text(x = w, y = 0.5, labels = sprintf("Probability of missingness in a single variable: %.2f",
        prob), pos = 4, cex = 0.6)
}
```

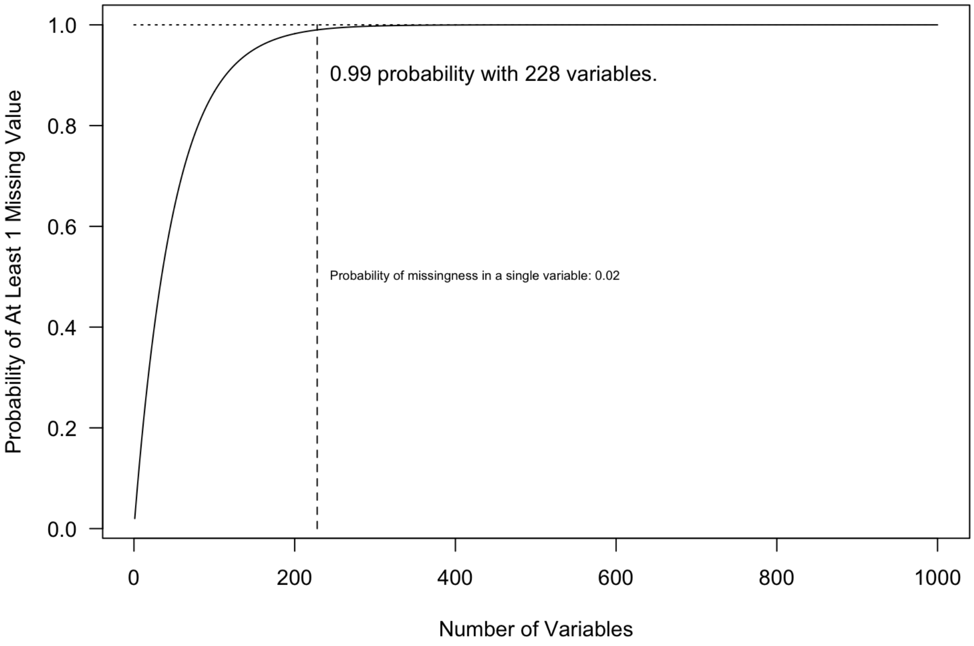
1% Missingness Per Variable

```
display.incompleteness.plot(prob = 0.01, max.variables = 1000)
```



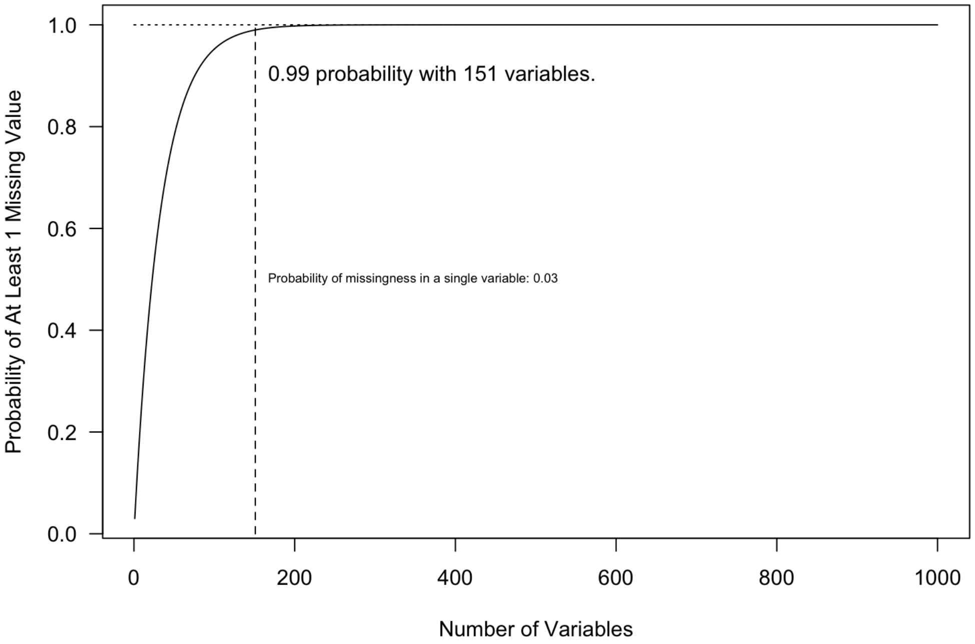
2% Missingness Per Variable

```
display.incompleteness.plot(prob = 0.02, max.variables = 1000)
```



3% Missingness Per Variable

```
display.incompleteness.plot(prob = 0.03, max.variables = 1000)
```



How Much Data Would Be Enough?

- **Medical Studies**: Thousands of patients, 10-30 Variables, Reasonably Precise Estimates.
- **Marketing Studies**: Thousands of respondents, Hundreds of Variables, Less Precision.
- Moreover, Marketing Studies also want to examine **numerous subgroups**, each of which have a fraction of the overall data.

For marketing studies, the number of questions can easily increase faster than the amount of data. So, in some sense, even tens of thousands of respondents may not be enough data for subgroup analyses.

And that's assuming that the sample size isn't greatly impacted by missing values!

Returning to the Marketing Data

- We want to better understand the missingness of the expanded marketing data.
- Where possible, we want to examine the opportunities for imputing missing values.
- Ideally, the approach would allow us to make full use of the measured data to inform the **models of the customers' states of engagement** in different products.

Missingness in the Person-Specific Variables

```
dat[, lapply(X = .SD, FUN = "mean.missing"), .SDcols = respondent.variables]
```

```
       age  income age_group gender income_group region persona
1: 0.03376 0.02325   0.03376      0      0.02325      0       0
```

A modest proportion of the respondents to the survey did not provide an age or income, which also impacts the categorical versions of these variables.

An Imperative from Management

- Given the strict time limits on the project and the pressure to accomplish so many tasks, they didn't want to invest any time into a complicated investigation of the missing data.
- Random sampling from the measured values seemed like the easiest approach to implement.
- Since some products had low rates of awareness, there were already small sample sizes for the downstream models.

With all of this in mind, the imperative from management was to **use imputation to maximize the effective sample size** of the models.

Imputing Age and Income

```r
sampled.imputation <- function(x){
  w <- which(is.na(x))
  if(length(w) > 0){
    x[w] <- sample(x = x[!is.na(x)], size = length(w), replace = TRUE)
  }
  return(x)
}
age.imputed.name <- "age - imputed"
income.imputed.name <- "income - imputed"
dat[, eval(age.imputed.name) := sampled.imputation(x = get(age.name))]
dat[, eval(income.imputed.name) := sampled.imputation(x = get(income.name))]
```

Inspecting the Imputation

```
dat[, summary(get(age.name))]
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  18.00   36.00   55.00   55.07   74.00   92.00   77648 
```

```
dat[, summary(get(age.imputed.name))]
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  18.00   36.00   55.00   55.07   74.00   92.00 
```

```
dat[, summary(get(income.name))]
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  14000   48000   83000   82673  117000  151000   53475 
```

```
dat[, summary(get(income.imputed.name))]
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  14000   48000   83000   82678  117000  151000 
```

A Test Case

```
one.id.missing.age <- dat[is.na(get(age.name))][1, get(id.name)]
imputed.ages.one.id <- dat[get(id.name) == one.id.missing.age,
    .SD, .SDcols = c(id.name, age.name, age.imputed.name)]
datatable(data = imputed.ages.one.id, rownames = FALSE)
```

Show 10 ⬍ entries Search:

id	age	age - imputed
30		84
30		92
30		80
30		22
30		58
30		35
30		40
30		27
30		30
30		54

Showing 1 to 10 of 23 entries Previous 1 2 3 Next

A Problem

- Random sampling was used to independently fill in each row.
- However, a single respondent has **multiple rows of data**.
- The method of the imputation did **not account for the structure of the data**.

One Imputed Age – Or Many?

- It seems reasonable that a single person's age – one missing value – **should only be filled in once**.

- At the same time, what we did has its merits – we are not putting all of our eggs in the basket of one randomly sampled imputation.

- However, for the **purpose of showing the data to the marketing team**, we did not want to be in a position in which one person has many ages. It would be a non-starter and detract from their willingness to appreciate the rest of our work.

So we have to go back to the drawing board.

Revising the Imputation

- One approach would be to imputed the data **prior to reshaping it** into long format.

- Otherwise, we have to impute the values person-by-person:

```
sampled.imputation.by.id <- function(dt, variable.name, impute.from, id.name){
  require(data.table)
  setDT(dt)

  measured.values <- dt[!is.na(get(impute.from)), .SD[1], by = id.name][, get(impute.from)]
  dt[, eval(variable.name) := get(impute.from)]
  dt[is.na(get(variable.name)), eval(variable.name) := sample(x = measured.values, size = 1), by = id.name]
  return(dt)
}
dat <- sampled.imputation.by.id(dt = dat, variable.name = age.imputed.name, impute.from = age.name, id.name = id.name)
dat <- sampled.imputation.by.id(dt = dat, variable.name = income.imputed.name, impute.from = income.name, id.name = id.name)
```

Inspecting the Results - Age

```
dat[, summary(get(age.name))]
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  18.00   36.00   55.00   55.07   74.00   92.00   77648 
```

```
dat[, summary(get(age.imputed.name))]
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  18.00   36.00   55.00   55.08   74.00   92.00 
```

```
imputed.ages.one.id <- dat[get(id.name) == one.id.missing.age,
    .SD, .SDcols = c(id.name, age.name, age.imputed.name)]
datatable(data = imputed.ages.one.id, rownames = FALSE)
```

Show 10 ▲▼ entries

Search:

id	age	age - imputed
30		80
30		80
30		80
30		80
30		80
30		80
30		80
30		80
30		80
30		80

Showing 1 to 10 of 23 entries

Previous 1 2 3 Next

Inspecting the Results - Income

```r
dat[, summary(get(income.name))]
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's 
  14000   48000   83000   82673  117000  151000   53475 
```

```r
dat[, summary(get(income.imputed.name))]
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  14000   48000   83000   82689  117000  151000 
```

```r
one.id.missing.income <- dat[is.na(get(income.name))][1,
    get(id.name)]
imputed.income.one.id <- dat[get(id.name) == one.id.missing.income,
    .SD, .SDcols = c(id.name, income.name, income.imputed.name)]
datatable(data = imputed.income.one.id, rownames = FALSE)
```

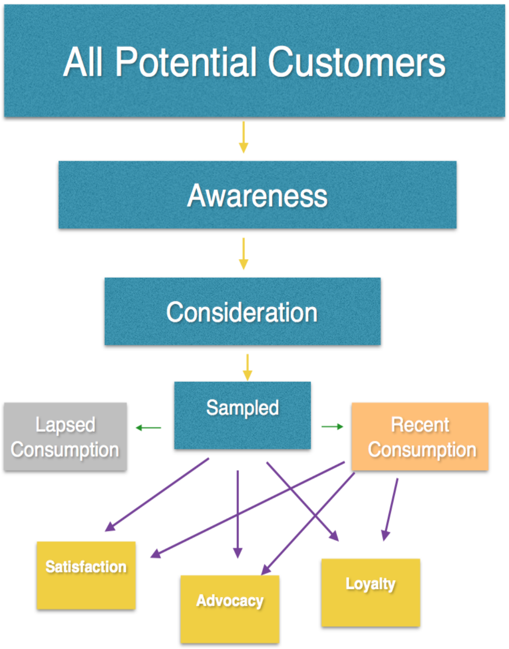
Show 10 entries Search:

id	income	income - imputed
76		118000
76		118000
76		118000
76		118000
76		118000
76		118000
76		118000
76		118000
76		118000
76		118000

Showing 1 to 10 of 23 entries

Previous 1 2 3 Next

States of Engagement



A Structure to the Data – and Its Missingness

- **Awareness**, **Consideration**, and **Consumption** each have downstream questions.

- Downstream questions are only asked **when the previous question was answered affirmatively**.

- Structurally, the downstream states of engagement **should be missing values** when an upstream state was negative.

- Having **Awareness = 0** should imply NA values for Consideration, Consumption, Satisfaction, and Advocacy.

Investigating Missingness in Awareness

- A small proportion of Awareness values had missing data:

```
dat[, mean.missing(x = get(awareness.name))]
```

```
[1] 0.01198565
```

- This is a manageable quantity. Perhaps we could simply ignore the rows with missing values.

- However, the imperative from the managers was to maximize the sample size.

Recovering Missing Awareness from the Structure

- Any **downstream data with measured values** would guarantee that Awareness should have been 1.
- Any rows with **completely missing downstream data** would likely suggest that Awareness should have been 0.
- Any rows with remaining ambiguity could potentially be imputed with random sampling.

Measured Downstream Data

```
recover.measurements.from.downstream <- function(dt, outcome.name, outcome.imputed.name, downstream.names){
  require(data.table)
  setDT(dt)
  dt[, eval(outcome.imputed.name) := get(outcome.name)]
  dt[is.na(get(outcome.name)), eval(outcome.imputed.name) := as.integer(rowSums(!is.na(x = .SD)) > 0), .SDcols = downstream.names]
  return(dt)
}
awareness.imputed.name <- "Awareness - Imputed"
dat <- recover.measurements.from.downstream(dt = dat, outcome.name = awareness.name, outcome.imputed.name = awareness.imputed.name, downstream.names
 = c(consideration.name, consumption.name, satisfaction.name, advocacy.name))
dat[, mean.missing(x = get(awareness.imputed.name))]
```

```
[1] 0
```

In this case, every single missing value of Awareness could be recovered based on whether the downstream values were measured or not.

Checking for Conflicts

- Are there cases in which Awareness was marked 0 but downstream data were recorded?

```
w <- which(dat[, get(awareness.name) == 0 & (!is.na(get(consideration.name)) | !is.na(get(consumption.name)) | !is.na(get(satisfaction.name)) |
  !is.na(get(advocacy.name)))])
length(w)
```

```
[1] 11
```

```
dat[w, eval(awareness.imputed.name) := 1]
```

- The converse case (Awareness is 1 but all downstream data is NA) would likely indicate a need for imputation in the next downstream variable (Consideration).

Missing Data in Consideration

- Consideration has a much higher rate of missing data:

```
dat[, mean.missing(x = get(consideration.name))]
```

```
[1] 0.4885087
```

- This is not surprising: For structural reasons, when Awareness is 0, then Consideration is not asked in the survey.

- In some sense, the form of the survey creates **structurally missing data**. In fact, it would be surprising to see a measured value for Consideration if Awareness were 0.

Recovering Consideration

- Consideration has both **upstream** (Awareness) and **downstream** (Consumption, Satisfaction, Loyalty) factors.

- The **downstream** recovery should work the same as it did for Awareness.

- However, any **upstream value that is 0** should **automatically lead to a missing value** for Consideration.

Upstream and Downstream Recovery of Missing Data

```r
recover.measurements.from.upstream.and.downstream <- function(dt, outcome.name, outcome.imputed.name, upstream.names, downstream.names){
  require(data.table)
  setDT(dt)
  dt[, eval(outcome.imputed.name) := get(outcome.name)]
  dt[is.na(get(outcome.name)), eval(outcome.imputed.name) := as.integer(rowSums(!is.na(x = .SD)) > 0), .SDcols = downstream.names]

  if(length(upstream.names) == 1){
    upstream.zeros <- dt[, get(upstream.names) == 0]
  }
  if(length(upstream.names) > 1){
    upstream.zeros <- dt[, rowSums(.SD == 0, na.rm = TRUE) > 0, .SDcols = upstream.names]
  }
  dt[upstream.zeros == TRUE, eval(outcome.imputed.name) := NA]
  return(dt)
}
consideration.imputed.name <- "Consideration - Imputed"
dat <- recover.measurements.from.upstream.and.downstream(dt = dat, outcome.name = consideration.name, outcome.imputed.name =
  consideration.imputed.name, upstream.names = awareness.imputed.name, downstream.names = c(consumption.name, satisfaction.name, advocacy.name))
```

Checking the Recovery of Consideration

- **Upstream Check**:

```
datatable(data = dat[, .N, keyby = c(awareness.imputed.name,
    consideration.imputed.name)], rownames = FALSE)
```

Show 10 entries

Search:

Awareness - Imputed	Consideration - Imputed	N
0		1094764
1	0	701000
1	1	504236

Showing 1 to 3 of 3 entries

Previous 1 Next

- When Awareness is 0, Consideration is always missing. When Awareness is 1, Consideration is either 0 or 1. This part checks out.

Checking Downstream

- **Downstream Check**:

```
datatable(data = dat[, .N, keyby = c(consideration.imputed.name,
    consumption.name)], rownames = FALSE)
```

Show 10 entries

Search:

Consideration - Imputed	Consumption	N
		1094764
0		701000
1		16944
1	0	289257
1	1	198035

Showing 1 to 5 of 5 entries

Previous 1 Next

- The only concerning case is when Imputed Consideration is 1 and Consumption is NA. This will be addressed in the imputation of Consumption below.

Recovering Consumption

- This should work with the same upstream and downstream logic:

```
consumption.imputed.name <- "Consumption - Imputed"
dat <- recover.measurements.from.upstream.and.downstream(dt = dat,
    outcome.name = consumption.name, outcome.imputed.name = consumption.imputed.name,
    upstream.names = c(awareness.imputed.name, consideration.imputed.name),
    downstream.names = c(satisfaction.name, advocacy.name))
```

Checking the Recovery of Consumption, Part 1

- **Upstream Check**:

```
datatable(data = dat[, .N, keyby = c(awareness.imputed.name,
    consideration.imputed.name, consumption.imputed.name)],
    rownames = FALSE)
```

Show 10 entries Search:

Awareness - Imputed	Consideration - Imputed	Consumption - Imputed	N
0			1094764
1	0		701000
1	1	0	306201
1	1	1	198035

Showing 1 to 4 of 4 entries Previous 1 Next

Checking the Recovery of Consumption, Part 2

- When Awareness or Consideration is 0, Consumption is always missing. When Awareness and Consideration are each 1, Consideration is either 0 or 1. This part checks out.

- **Downstream Check**: Advocacy

```
datatable(data = dat[, .N, keyby = c(consumption.imputed.name,
    advocacy.name)], rownames = FALSE)
```

Show 10 entries Search:

Consumption - Imputed	Advocacy	N
		1795764
0		306201
1		3934
1	0	125082
1	1	69019

Showing 1 to 5 of 5 entries Previous 1 Next

- The only concerning case is when Imputed Consumption is 1 and Advocacy is NA. This will be addressed in the imputation of Advocacy below.

Checking the Recovery of Consumption, Part 3

- **Downstream Check**: Satisfaction

```
dat[, satisfaction.binary := 1*(get(satisfaction.name) > 0)]
datatable(data = dat[, .N, keyby = c(consumption.imputed.name, "satisfaction.binary")], rownames = FALSE)
```

Show 10 entries

Search:

Consumption - Imputed	satisfaction.binary	N
		1795764
0		306201
1		2180
1	0	32
1	1	195823

Showing 1 to 5 of 5 entries

Previous 1 Next

- The only concerning case is when Imputed Consumption is 1 and Satisfaction is NA. This will be addressed in the imputation of Satisfaction below.

Imputing Satisfaction

- We have gotten this far along without performing any randomized imputation.

- Now we only need to **sample measured values** for the case when the Imputed Consumption is 1 and Satisfaction is missing.

- However, we want to do this **separately in each Product** to maintain the differences in engagement for each product.

```
satisfaction.imputed.name <- "Satisfaction - Imputed"
dat[!is.na(get(consumption.imputed.name)), eval(satisfaction.imputed.name) := sampled.imputation(x = get(satisfaction.name)), by = product.name]
```

Imputing Advocacy

- We will likewise **sample measured values** for the case when the Imputed Consumption is 1 and Advocacy is missing.

- Again, we want to do this **separately in each Product**.

```
advocacy.imputed.name <- "Advocacy - Imputed"
dat[!is.na(get(consumption.imputed.name)), eval(advocacy.imputed.name) := sampled.imputation(x = get(advocacy.name)), by = product.name]
```

A Structural Imputation

- By understanding the structure of the states of engagement, we were able to recover much of the missing information.

- Better yet, we limited the degree to which random sampling even needed to be employed.

- The final result was imputed values only where they were needed – while maintaining the structural missingness that was built into the survey.

The Next Challenge: The Expanded Marketing Measures

- Recall that the expanded data included a variety of new variables.

- These were intended to be binary indicators of interests in categories that might be **novel indicators of engagement** with a product.

```
new.variables
```

```
 [1] "food"                      "drinks"
 [3] "alcohol"                   "gardening"
 [5] "outdoors"                  "hiking"
 [7] "photography"               "camping"
 [9] "fashion"                   "shoes"
[11] "beauty products"           "toys"
[13] "games"                     "parenting"
[15] "family"                    "home decorations"
[17] "architecture"              "art"
[19] "cartoons"                  "animation"
[21] "celebrities"               "movies"
[23] "television"                "drama"
[25] "comedy"                    "science fiction"
[27] "personal finance"          "sports"
[29] "cooking"                   "barbecue grills"
[31] "health"                    "exercise"
[33] "wellness"                  "spirituality"
[35] "religion"                  "community organizations"
[37] "politics"                  "news"
[39] "skiing"                    "literature"
[41] "poetry"                    "music"
[43] "musical instruments"       "astronomy"
[45] "space"                     "travel"
[47] "tourism"                   "restaurants"
[49] "cars"                      "bicycles"
[51] "carpentry"                 "knitting"
```

Examining A Few of the New Variables

```
datatable(data = dat[, .N, keyby = eval(new.variables[1:2])],
    rownames = FALSE)
```

Show 10 entries Search:

food	drinks	N
		677488
	1	13938
0		1537596
0	1	31073
1		39169
1	1	736

Showing 1 to 6 of 6 entries

Previous 1 Next

More of the New Variables

```
datatable(data = dat[, .N, keyby = eval(new.variables[3:4])],
    rownames = FALSE)
```

Show 10 entries

Search:

alcohol	gardening	N
		2202572
	1	43907
1		52647
1	1	874

Showing 1 to 4 of 4 entries

Previous 1 Next

Zeros and No Zeros

- Some of the variables (like **food**) have values of **0, 1, and NA**.
- Other variables (like **drinks**) only have values of **1 and NA**.

Counting the Unique Non-Missing Values

```r
num.unique <- function(x) {
    return(length(unique(x[!is.na(x)])))
}
unique.counts <- dat[, .(Variable = new.variables, `Number of Unique Values` = as.numeric(lapply(X = .SD,
    FUN = "num.unique"))), .SDcols = new.variables]
datatable(data = unique.counts, rownames = FALSE)
```

Show 10 entries Search:

Variable	Number of Unique Values
food	2
drinks	1
alcohol	1
gardening	1
outdoors	1
hiking	2
photography	1
camping	2
fashion	1
shoes	2

Showing 1 to 10 of 52 entries

Previous 1 2 3 4 5 6 Next

A Problem

- When the variable only includes values of **1 or NA**, where are the zeros?
- One Point of View: The **NA** values **really should be 0 values**.
- Another Point of View: Missingness means something different, e.g. that **the vendor could not identify the survey's respondent**. Meanwhile, everyone identified said yes.

A Lack of Information

- I didn't have any basis for deciding between the two points of view.
- My managers told me to use my best judgment. They weren't willing to bring it up with the company we were working with.
- Even the company would likely not have had full visibility into the reasons for this encoding.

Making a Choice

- I opted to say that **missing values should be imputed 0 values** when the original variable did not include any zeros.

- Meanwhile, when the variable **did contain 0 and 1 values**, the NA values would be **imputed by random sampling** of the measured values.

- In this way, we would get an imputation that provided complete information.

But It’s Complicated

- Furthermore, we decided that the imputation by random sampling would take place **separately by categories of Awareness** in each product.

- This was meant to ensure that any correlations with Awareness (and hence the other states of engagement) would be maintained.

- However, in doing so, a single person would receive **different values** for each product as a result of the random sampling.

- This choice was perhaps not especially principled, but it was made under pressure to deliver working models on a tight deadline and in a manner that would hopefully not obscure the effects. Randomly sampling **across levels of Awareness** seemed like it would detract from any potential signal that might exist in the expanded marketing data.

Imputation: 1 and NA Variables Become Binary

```
switch.na.to.zero <- function(x){
  x[is.na(x)] <- 0
  return(x)
}
variables.to.switch <- unique.counts[`Number of Unique Values` == 1, Variable]
dat[, (eval(sprintf("%s - Imputed", variables.to.switch))) := lapply(X = .SD, FUN = "switch.na.to.zero"), .SDcols = variables.to.switch]
```

Imputation: Randomly Sample from Binary Variables

```
sampled.imputation.by.group <- function(x, by.y){
  unique.y <- unique(by.y)
  all.measured.values <- x[!is.na(x)]
  for(i in 1:length(unique.y)){
    w <- which(is.na(x) & by.y == unique.y[i])
    if(length(w) > 0){
          measured.values <- x[!is.na(x) & by.y == unique.y[i]]
          if(length(measured.values) == 0){
            measured.values <- all.measured.values
          }
          x[w] <- sample(x = measured.values, size = length(w), replace = TRUE)
    }
  }
  return(x)
}
variables.to.impute <- unique.counts[`Number of Unique Values` == 2, Variable]
dat[, (eval(sprintf("%s - Imputed", variables.to.impute))) := lapply(X = .SD, FUN = "sampled.imputation.by.group", by.y =
  get(awareness.imputed.name)), .SDcols = variables.to.impute, by = product.name]
```

A Working System

- These selections led to a system with fully imputed data.

- While not every choice may have been ideal, this was **good enough for a prototype**.

- The Reporting Engine was equipped to use **Lasso Regression** to perform variable selection on all of the outcomes across all of the models. In this way, we had an opportunity to identify the novel factors that might predict engagement.

After an extremely difficult and arduous journey, there appeared to be light at the end of the tunnel. I even took the weekend before the rollout off… **until I got a call on Sunday afternoon…**

The Story of the Barbecue Grills

- Imagine my surprise when I saw this logistic Lasso output in the model for **imputed Awareness in Cookie Crumble**:

Show 10 ▾ entries

Search:

Variable	Beta	Odds Ratio
regionMidwest	0.13	1.14
regionSouth	0.16	1.17
regionWest	-0.26	0.77
\`barbecue grills - Imputed\`	4.59	98.79

Showing 1 to 4 of 4 entries

Previous 1 Next

A Perfect Predictor

- If taken at face value, then having an interest in barbecue grills would increase the odds of Awareness in Cookie Crumble by **approximately 9879%**.

- This would mean that an identified interest in barbecue grills would be **a perfect predictor of awareness in Cookie Crumble**.

- Furthermore, there was a perfect correlation, too:

```
dat[get(product.name) == "Cookie_Crumble", cor(x = get(awareness.imputed.name),
    y = `barbecue grills - Imputed`, use = "pairwise.complete.obs")]
```

```
[1] 1
```

Too Good to be True – Way Too Good

- **This relationship seemed impossible**. Barbecue grills and a cookie-based snack product have little in common.
- Any correlation that is too strong **should be considered suspicious**.
- With the launch of the product looming the next day, I frantically investigated what might have led to such a result.

Returning to the Original Data

- The correlation we found was on the **imputed values** for Awareness and Barbecue Grills.

- I returned to the **original data**:

```
datatable(data = dat[get(product.name) == "Cookie_Crumble",
    .N, keyby = c(awareness.name, "barbecue grills")], rownames = FALSE)
```

Show 10 entries   Search:

Awareness	barbecue grills	N
		1220
0		24017
0	0	2
1		74758
1	1	3

Showing 1 to 5 of 5 entries   Previous 1 Next

Five Measured Values

- The original data only contained **five respondents** with a measurement for Barbecue Grills.
- These five respondents had **a perfect correlation** in their interest in Barbecue Grills and their Awareness of Cookie Crumble.
- Perfect correlations also showed up in a number of other products for these five respondents.

Imputation Sunk the Ship

- Barbecue Grills was missing in **more than 99%** of the original data for Cookie Crumble.
- We chose to perform imputation on every variable in spite of **unreasonably high rates of missingness** in some cases.
- Meanwhile, our choice to **randomly sample within categories of Awareness** was designed to maintain some of the correlations between these variables and the states of engagement. **In the case of Barbecue Grills, it certainly did maintain the correlation!**

Salvaging the Situation

- In light of these findings, I convinced the managers that we should **remove Barbecue Grills** and other variables with extremely high rates of missingness from consideration in the models.

- Ultimately, these last-minute changes allowed us to barely cross the finish line in delivering the prototype that we promised.

- However, after many trials and tribulations in developing the project, this issue – **which should have been avoidable** – brought forward an opportunity to evaluate many of the reasons why this work turned out to be far more challenging that anyone anticipated.

Evaluating What Went Wrong

- Unfortunately, this project was a case study in nearly everything that could go wrong working in data science.
- While we delivered a result and put our best foot forward, it papered over a very tense working situation that did not generate positive momentum or experiences for anyone involved.
- In the years since that project, I have repeatedly taken up the task of **assessing the issues and identifying areas for improvement**.

Catalogging the Areas for Improvement

The issues could be placed into a number of categories:

- **Technical Skills**: what could I have done better to handle the issues that we discovered in the data?
- **Project Planning**: how might we have better anticipated some of the problems that were likely to happen?
- **Working with the Management**: how can technical developers better convey the issues that can be hard for less technical managers to understand?
- **Working with the Company**: How might our firm have better integrated with the contracting company to create a successful engagement?

Technical Skills #1: Use Better Tools

- At the time, I was still working with R’s **legacy tools**: classical data.frames and for loops.

- The **reporting engine**, while functional, was **tremendously slow** in processing such a large volume of data.

- **Suggested Solution**: Using **data.table** would likely have solved half of my problems with slow running times, laborious coding, and limited opportunities for iteration.

Technical Skills #2: Better Designs

- My choice of working with the wide-form data through **dynamic extraction** (Lecture 7) required substantially more processing and was extremely difficult to work with in a dynamic system.

- **Suggested Solution**: Reshaping the data into **long format** would have greatly facilitated much of the work of developing models and building the reporting engine.

- **Suggested Solution**: Shifting more of the work into **pre-processing** would also have led to a more useful application.

Technical Skills #3: Dynamic Designs

- Issues related to building a vast number of models across many outcomes, inputs, and products created numerous issues with a **lack of contrast**.

- Fixing problems **one at a time** only seemed to lead to more problems.

- **Suggested Solution**: Understanding how to systematically handle the issue – using **dynamic designs** for the models – would have greatly simplified the work of building customized models across so many scenarios.

Technical Skills #4: Understanding Missing Data

- With hindsight, all of the issues related to missing data would be **relatively easy to address**.

- However, the **combination of multiple issues** can be overwhelming, especially when many of them are in novel settings.

- **Suggested Solution**: Exerting greater caution in imputation would be a key lesson. When the rates of missingness are too high, or when there are too many variables to impute, employing a **unified, automated solution** may create more problems than it solves.

Taking Responsibility

- The project's results were a culmination of my own individual performance, the role of management, and the working relationship we had with the big company.

- However, regardless of any other issues, the primary responsibility to deliver good work was on me.

- While the project did generate some results, I certainly could have done a better job.

- With the benefit of time and experience, I have significantly invested in improving my technical skills and approach to building working relationships.

Project Planning #1: Data Cleaning

- No matter what grand plans you might have, **nothing gets done without cleaning the data**. Our team did not budget enough time for this.

- Furthermore, it can be **almost impossible to estimate** what data cleaning might entail. Our team negotiated a contract and a timeline before we had a chance to work with the data.

- **Suggested Solution**: A better practice is one that's inconvenient: any contracted project likely needs to be split into separate phases, with **analyses only commencing after cleaning the data** is reasonably complete.

Project Planning #2: Massive Surveys

- Our team also did not anticipate the difficulty of working with 4 separate 100-page surveys.

- Translation and Coding of the variables carried on for weeks in a **mostly manual process**. Since every question had unique responses, there was no way to scale the process of coding the data into a more usable form.

- This work, which was essential to the goals of the project, was not understood or appreciated by management or the client.

- **Suggested Solution**: Unfortunately, the only solutions I can suggest are to a) design better surveys or b) allow for more time in handling them.

Project Planning #3: Margin for Error

- The original plan to complete the work did not account for any of the issues we ultimately faced.

- Resolving these issues left too little time to do the planned work well.

- Working as a vendor on a fixed budget **enhances the risks** (both upside and downside). Most of the time, one party is going to lose this bet, and **sometimes both sides will lose**.

- **Suggested Solution**: Changing the **working relationships and the incentives** of these projects can allow for greater collaboration. **Unanticipated obstacles can be discussed**, and the plans can shift accordingly.

Working with Management #1: Managing the Managers

- The managers on this project did not have any training in data science.

- Nonetheless, they wanted to maintain tight control over the designs, products, schedule, and communication. This often led to unproductive suggestions and unreasonable expectations.

- **Suggested Solution**: Where possible, shift the working relationship toward **greater collaboration** centered on getting the work done. That's not always possible to create in the moment, but you have the opportunity to seek better situations.

Working with Management #2: Visibility

- Much like we've seen with our Lament of the Data Scientist, a large component of the work (e.g. cleaning data) goes into **parts of a project that don't immediately generate results**.

- Creating visibility in these settings can be challenging. When I wrote up detailed plans for the project in the early going, the response I received was: **Don't write long e-mails. I don't want to pay you to write long e-mails.** Needless to say, this wasn't a helpful way to respond to my substantive contributions.

- **Suggested Solution**: Don't back down. Your managers have to work with you as much as you have to work with them. When you stop sharing information, they assume that you're not generating results.

Working with Management #3: Decisions

- Our approach to **imputing missing data** was crafted to satisfy the management's imperative to use all of the sources of data. This was more about **demonstrating the value of our work** rather than **generating good results**.

- This decision hurt us in the end with the **story of the Barbecue Grills**.

- **Suggested Solution**: Methodological judgments based on the data should take precedence over imperatives that are not grounded in the facts of the situation. That is easier said than done, but these issues can't be waived away.

Working with Management #4: Incentives

- In this case, the managers were primarily interested in **selling other projects** to different parts of the big company.

- What **looked good for a sales pitch** to another division was not the same thing as what would generate good results for this project. Our incentives weren't aligned.

- **Suggested Solution**: Closer coordination might have led to a **mutually agreeable solution**. With better insight into what would help the managers on their other sales pitches, I likely could have **refocused my work** in a way that would have better served their priorities and reduced my burden simultaneously.

Working with the Company #1: Cutting Out the Middlepeople

- Our main client at the company **wouldn't work with me directly**. He only engaged us through a handful of phone calls led by the managers.

- Meanwhile, our team was working offsite (and I was on the opposite coast from everyone else). There were very limited opportunities to coordinate or discuss the challenges that we faced, internally or externally.

- **Suggested Solution**: getting a direct line of communication with those who have the needed expertise makes all of the difference. Working through middlepeople is a recipe for a poor collaboration.

Working with the Company #2: Really Cutting Out the Middlepeople

- Our 10-week contract paid the consulting company a sum that was roughly equivalent to my annual salary at the time.

- Some of the contract went toward my labor, but much of it paid for all of the other aspects of running the consulting company.

- This design leads to numerous problems: **burnout** of the consultants, budgets that require **unreasonably high billing rates**, and **adversarial relationships** between the firms.

- **Suggested Solution**: I came away from that project feeling like a **leaner design** would have allowed me to help the client over a longer period of time on the same budget. We could have delivered results of a greater quality and built better working relationships.

The Resolution

This challenging project led to much introspection, an initiative to improve my technical skills, ruminations on the nature of work and organizations, and a greater knowledge of my own goals and capabilities.

With all of that in mind, you can understand why I eventually decided to leave this job in favor of working more independently. And, through a series of accidents, this is part of the story of how I ultimately became the professor of this class.