Into to Data Science - Capstone Project (Data Wrangling)

John Campi

Nov. 12, 2018

Contents

1 Project Overview

1.1 What is the problem this project addresses?

Identify the key instrument sensors of a semiconductor manufacturing line and model the alarm conditions for potential chip failures.

1.2 Who is your client and why do they care about this problem? In other words, what will your client DO or DECIDE based on your analysis that they wouldn't have otherwise?

This capability would be sought out by any semiconductor manufacturer who wants to maximize their yield. Currently wafers are tested after specific process levels are completed. Much of the critical testing can't be performed until very far along in the manufacturing process. Thus a failure early in the process will consume costly resources as the wafer continues onto other fabrication steps, only to be junked at the end. If statistical sampling is used then there's also the added risk of failing chips shipping to customers.Having the ability to catch failures in almost real time during any process step minimized the chance of failure propagation and improves isolation time of equipment issues thereby greatly improving yield.

1.3 What data are you going to use for this? How will you acquire this data?

SECOM Data Set

- Dataset is comprised of 1567 observations, 591 variables, and 104 fails.

1.4 In brief, outline your approach to solving this problem.

- The first step is to identify the relevant sensors and try to reduce the number of variables through PCM analysis.
- Then perform regression to model the various failure conditions.

1.5 What are your deliverables? Typically, this would include code, along with a paper and/or a slide deck.

- A report describing the methodology and model. The R program can be included as an appendix.
- A slide show presenting the overall procedure and benefits.

2 Data Wrangling

2.1 Data Wrangling - Summary of Approach.

- Assemble csv files and update variables as needed.
- Remove irrelevant variables.
- Classify all missing data as "NA".
- Impute missing data.

2.2 Assemble Initial Dataset.

The SECOM Data Set consists of two csv files. The first is a list of pass/fail and date/time results, one entry per lot run, and the other contains the corresponding continuous numeric results of sensor readings from a semiconductor manufacturing line. The data files do not contain a header so variable names Status, Date, and Time were assigned for pass = -1 / fail= +1, date and time. The remaining variables assumed default names: V1, V2, etc. Table 1 below shows a sampling of the initial dataset. Since the sensor variables are not named there is no way to attribute a meaning to the sensor readings so I will take the "black box" approach to analysis.

Table 1: Sampling of the SECOM dataset.

Status	Date	Time	V1	V2	V3	V4
-1	19/07/2008	11:55:00	3030.93	2564.00	2187.733	1411.1265
-1	19/07/2008	12:32:00	3095.78	2465.14	2230.422	1463.6606
1	19/07/2008	13:17:00	2932.61	2559.94	2186.411	1698.0172
-1	19/07/2008	14:43:00	2988.72	2479.90	2199.033	909.7926
-1	19/07/2008	15:22:00	3032.24	2502.87	2233.367	1326.5200

2.3 Initial Cleanup

Sensor data variables are continuous by nature so any variable that contains only missing data or has no variation is irrelevant for this analysis and can be dropped. The approach taken here was to drop all variables where min = max. It's not clear why these data were included in the SECOM dataset, but since the goal is to identify signals or combinations of signals leading to an alarm condition, unvarying sensor data are irrelevant. The next important issue with the dataset was to properly classify all missings as "NA". Missing results can be defined by a number of non-standard labels including "N/A", "missing", "na" or even " ". The naniar package provides a simple function replace_with_na_all() to simplify converting this arbitrary list of labels to"NA". Finally, there were a number of"NaN" designations that aren't typically interpreted as missings, but since the sensor data should be continuous real values it was determined these values should be treated as "NA". It was found that initially 5.59% of the dataset was missing. While that doesn't seem to be too significantly large, it depends on how missingness is distributed within the dataset. Among the many useful features of the naniar package are plotting routines for visually exploring missingness. One of the routines gg_miss_var() is shown below in Figure 1 in which the variables are ordered by total missingness and plotted on the y-axis, and the number of missing observations on the x-axis. The number variables in this dataset is too large for printing so are omitted from the y-axis. The notable takeaway here was that most of the missing data was limited to a relatively few number of variables. The safe approach taken was to drop all variables with $> 10\%$ missing data leaving just 0.41% total missing data for imputation.

2.4 Imputation

There are several R packages for imputing data. Initially, the simputation package was chosen for it's ease of use and integration with naniar and ggplot2. Unfortunately, the number of variables in this dataset created multiple run-time issues for the simputation engine so it had to be abandoned. Instead, the mice package, which stands for "Multivariate Imputation by Chained Equations", provided powerful fitting functionality at a moderate computation cost. The package is capable of fitting a different imputation model to each variable, but the norm.nob method was applied unilaterally and found to return reasonably good values on comparing pre- and post-imputation distributions. Figure 2 show a sample distribution for a random variable overlaying the imputed values in the histogram. A summary of the initial imputation effort is shown in Table 1.

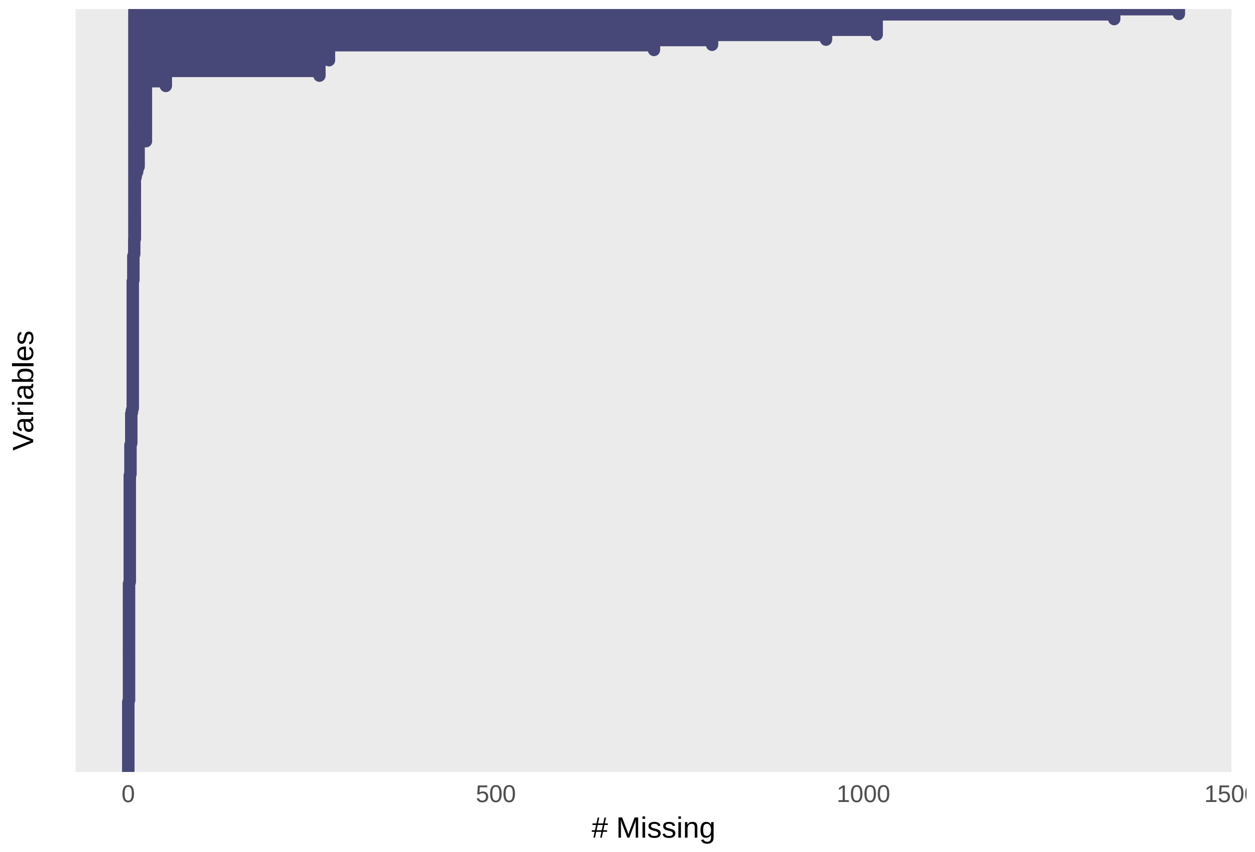

Figure 1: Visualization of initial missingness.

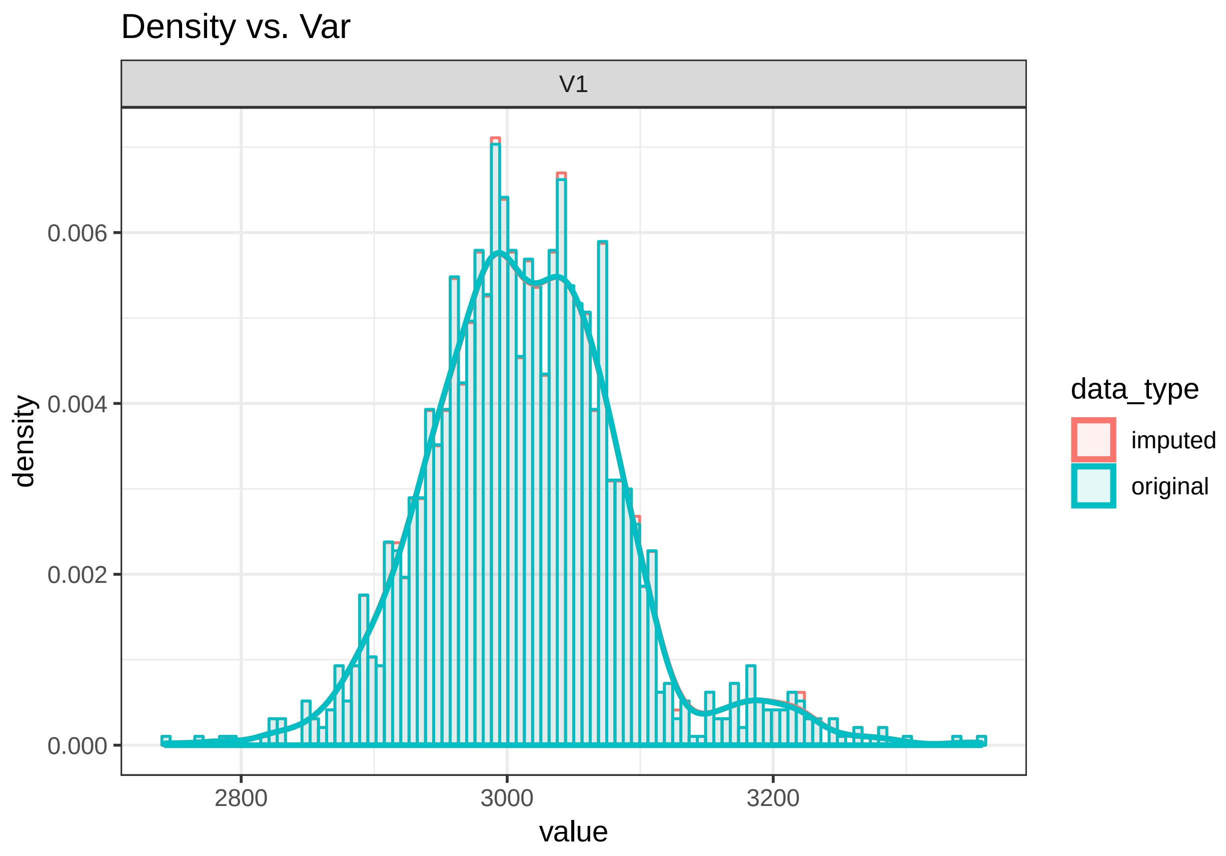

Figure 2: Example distribution before and after imputation

Table 2: Summary of wrangling.

Metric	Initial	Final
# of Variables	593.00	418
# of Observations	1567.00	1567
% Missings	4.51	0