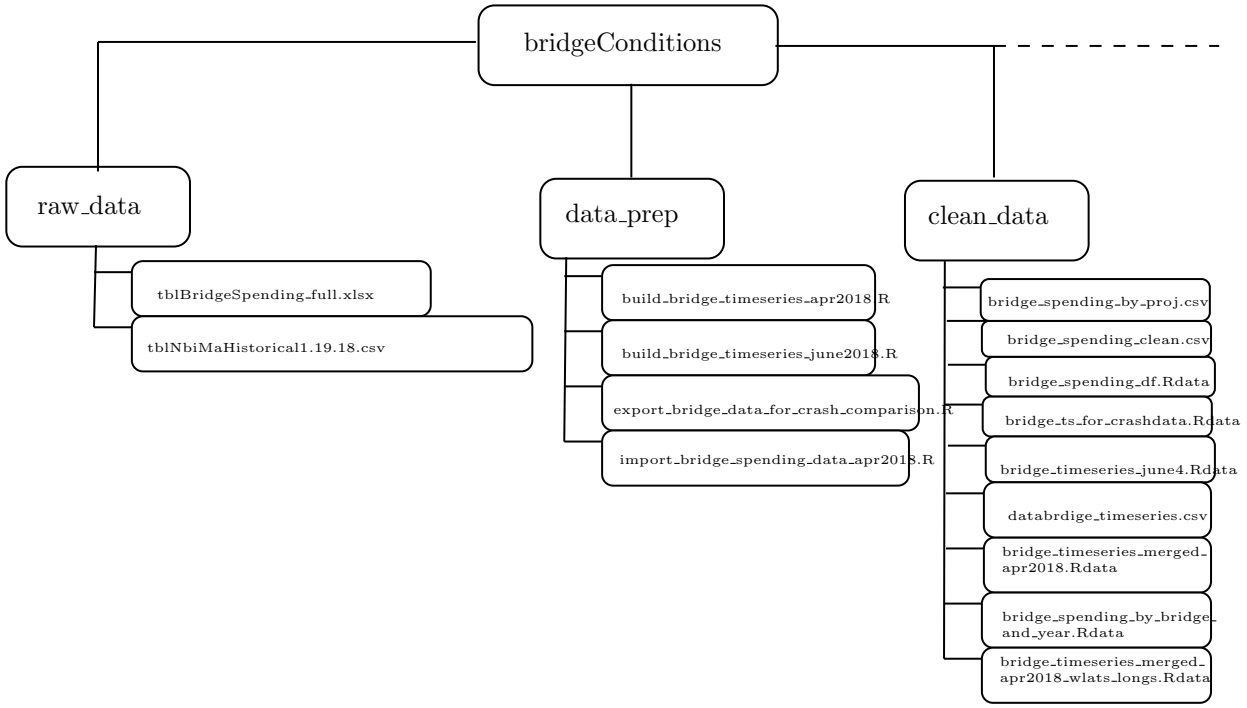
Bridge Project Cleaning Documentation

May 15, 2020



1 Raw Data

We have two raw data files pertaining to bridges and projects in the `raw_data` folder that will be used to create our intended data sets.

- The spending associated with bridge related projects are recorded in `tblBridgeSpending_full.xlsx`. Each observation corresponds to a line item for a single project, meaning that a given project has several observations associated with it. Furthermore, it is common for a project to include multiple bridges, resulting in the `BridgeNumbers` variable taking on a set of identifying numbers associated with bridges, instead of a single value. The key variable in this data set is `Posting_Line_Amount`, as it will allow us to determine spending for a single bridge in some time period.
- The conditions associated with bridges over time are found in `tblNbiMaHistorical1.19.18.csv`. Each observation corresponds to the health of a bridge in a given year from 1991 to 2016, with certain bridge-years being represented by multiple observations. The majority of variable correspond to specific characterstics of the bridge (location, size, year of construction, etc.), but there exists key variables which record the condition of parts of the bridge on a scale from one to nine (one being the worst possible condition, and nine being the best). In particular, the condition of the following parts are recorded: deck, superstructure, substructure, channel, and culvert.[1]

[1] A bridge does not necessarily posses each of these parts.

2 Code

The code used to clean and manipulate the raw data aims to make two pieces of information clear and accessible for analysis: the health of a bridge, and the resources allocated towards maintaining a bridge. The goal is to assemble a data set where each observation is a bridge in a given year, with variables corresponding to the condition of, and spending used towards, that bridge in the year. The four scripts used for cleaning are found in `data_prep` and are intended to run in the following order:

1. `import_bridge_spending_data_apr2018.R`
2. `build_bridge_timeseries_june2018.R`

Our first step is creating a data set which contains spending information for each bridge. This is executed with `import_bridge_spending_data_apr2018.R`, which manipulates the data presented in `tblBridgeSpending_full.xlsx`. As previously noted, each observation in this table corresponds to a line item for a project, many of which include multiple bridges. We want each observation to correspond to a line item of a bridge. To be explicit, we may have a single observation such as:

PROJECT_NO	...	Posting_Line_Amount	...	BridgeNumbers
100000	...	100	...	B1, B2

,

opposed to observations corresponding to each bridge:

PROJECT_NO	...	Posting_Line_Amount	...	BridgeNumbers
100000	...	50	...	B1
100000	...	50	...	B2

.

Note that in this particular situation, it is not clear how much money is spent on a line item for B1 versus B2. We assume that the amount allocated to each bridge is simply the average line amount for the bridges in a given project line item. We loop over each row in the original data frame to "uncouple" bridges in this way, resulting in approximately 17,000 additional observations. The resulting data frame is saved as `bridge_spending_df.rdata`, and in CSV form as `bridge_spending_clean.csv`. Finally, we use `summarize()` and `group()` to create a data frame with spending data for each *realized* bridge-project pair, and another with spending data for each *realized* bridge-year pair.[2] We save these data frames as `bridge_spending_by_proj.csv` and `bridge_spending_by_bridge_and_year.rdata` respectively.

The next script, titled `build_bridge_timeseries_june2018.R` , is responsible for creating the main time series data set that will be used in our analysis. This script combine the spending data now presented in `bridge_spending_by_bridge_and_year.rdata` with the bridge condition data in `tblNbiMaHistorical1.19.18.csv`. The script begins by cleaning `tblNbiMaHistorical1.19.18.csv`. We discard any observations that correspond to a bridge that was not inspected in a year. Certain bridges may have been represented by multiple projects in a year, resulting in several observations corresponding to a bridge in a single year. For example, there are 56 observations for bridge B16232 in 1991. In order to have each observation correspond to a single bridge-year, we need to write a function which collapses these observations. This is achieved with our user-defined function `getMaxAdjusted`, which when combined with `group()`, collapses these groups of observations and takes the maximum value of a variable to be the value of the single collapsed observation.[3] The two following tables provide an example of what a portion of the data may look like before and after collapsing observations in this fashion:

bridgeID	data_year	...	deck	superstructure	...
B1	2010	...	6	NA	...
B1	2010	...	5	NA	...
B1	2010	...	6	NA	...
B1	2010	...	7	NA	...

[2]Emphasis is put on realized here, as we do not record bridge-project or bridge-year pairs that do not occur. If we were to do this, then the overwhelming majority of observations would have missing values, as no single project includes every bridge, and no bridge is repaired each year. In our final time series dataset, this pattern is actually realized, and we will see that most of the spending data is missing. An alternate approach is to record each of these missing project spending values as zero, but that implies maintenance was conducted, it just was done at no cost.

[3]The assumption that the maximum value is taken is made in an effort to be as conservative as possible in recorded a bridge's condition.

bridgeID	data_year	...	deck	superstructure	...
B1	2010	...	7	NA	...

.

Next, we need to address the matter of missing observations. In order for the model specified in Stan to work, we need our data frame to have no missing values. We accomplish this with the function `interpolateMissingValsWLastSeen`. This user-defined function, as implied by its name, replaces a missing value with the last value that variable took on, and does so by bridge. If we had,

bridgeID	data_year	...	deck	...
B1	2009	...	5	...
B2	2010	...	NA	...

,

then the application of `interpolateMissingValsWLastSeen` to our data frame yields

bridgeID	data_year	...	deck	...
B1	2009	...	5	...
B2	2010	...	5	...

.

Finally, we merge this data with `bridge_spending_by_bridge_and_year.rdata` and arrive at our final time series data set, `bridge_timeseries_june4.rdata`.