**Ruby DataFrames Construction Log**

**8/24/16**

Make the data frames a hash of arrays

🡪 All arrays must be of the same length

🡪 Each must have the same data type

**Initialize** method:

If the input is an array of hashes and has certain values undefined in certain hashes, then write then in as NULL, but raise a warning saying that values were NULL in some of the input hashes

Do the same if the input is a hash and its arrays are of different lengths

For displaying data, need to put in Postgres-style divisions between columns (vertical lines separating columns)

**8/25/16**

Also, need to create GitHub repository for DataFrame code

🡪 Issues with pushing changes for some reason…

Alright, I guess I’ll have to F around with this later—just email yourself the code and try to get it to work at home

Pandas functions used in script to write data to REDCap:

* Read CSV files (do later)
* Retrieve a particular column
* Change column data types
* Get all rows that contain one of the specified values in the specified column
* Retrieve specific rows where the specified values are in the specified column
  + Should be capable of selecting by more than one column simultaneously using Boolean logic
  + This is probably the most important method: the bread and butter of DataFrames
  + Have at least a hackish version working right now—selects rows using truth array that is created using Ruby’s built-in Boolean operators and the .map method
* Set the values in particular rows that meet the specified criteria
* Define a column (add column name with corresponding values)
* Remove a specified column (already can do with the select\_columns() method, but since I have an add\_column() method, I should have a method that does the reverse as well)
* Drop duplicates in DataFrame
* Add rename\_column() method as well, just to make it easier to do what you want
* Add ability to merge two DataFrames on the specified fields
  + Output from merge successfully matches that of Pandas!
* Method to append one DataFrame to another with matching columns
  + Options to either do inplace or return a new DataFrame
* Sort DataFrame by one or more of the specified columns, with the ability to specify ascending or descending (default = ascending)
  + ACTUALLY, I don’t need a sort method since I can use Ruby’s built-in ability to sort arrays of hashes (convert DataFrame to array and do sort, then convert back)
* Write DataFrame to CSV file (class method)
* Read DataFrame from CSV file (non-class method—outside of class definition area in file)
* Concatenate two or more DataFrames simultaneously (external method)

WHENEVER YOU MODIFY THE DATA IN A COLUMN, AFTERWARDS YOU MUST CHECK TO ENSURE THAT ALL OF THE VALUES IN THE COLUMN ARE STILL OF THE SAME DATA TYPE!

Problem with converting strings to ints/floats: ‘’ converted to 0

🡪 Must convert all ‘’ to nil

Easy to select rows where a column or more than one column has a specific value

How do I select rows meeting multiple Boolean selection criteria simultaneously?

Create a Boolean selection array?

🡪 Use the ruby .map method🡪 get the values of the columns of interest in the DataFrame

May want to later add in feature where if you are initializing a new DataFrame made up solely of data already existing in the current one, then it should skip the quality control checks since they are unnecessary (should make any script utilizing the DataFrames run faster)

🡪 Not doing right now (keeping in mind Tristan’s warning about premature optimization), but add in later

NOTE: all of your data modification methods modify the original DataFrame—they don’t return a new DataFrame with the modifications performed on it

🡪 I could get around this by simply using the .select\_columns() method, specifying all of the columns in the DataFrame to create an exact copy

🡪 However, this is clunky

* Address later—not an issue of critical importance right now since I can use the hack specified above to get around

How to sort?

🡪 Use .to\_a + built-in Ruby sort feature, then convert back to DataFrame?

**11/7/16**

Modifications to DataFrame class:

* Replaced .disp method with .to\_s, according to Tristan’s recommendation (still can specify row limit in parentheses)
  + This returns a string version of the DataFrame. However, you need to use puts to get it to display the same way it used to (otherwise, the \n characters will be shown if you just return the string
* Changed .select method to .column method (returns the specified column’s values in order in an array)
* .remove\_column method now returns the name of the column that was removed if the specified column was actually removed from the DataFrame and returns nil otherwise
* Could use .map method on version of DataFrame converted to an array if I want to change values
  + However, while this could be done in one line, it would be an extremely wide line—probably not worth it, although I do have the option if I want to use it
* .drop\_duplicates method now returns the output regardless of whether it was changed inplace or a new DataFrame was generated
* .append now returns the resultant DataFrame, even when the calling DataFrame was modified

**11/8/16**

Additional modifications to DataFrame class:

* Added class methods .mean() and .std(), which return the calculated mean and standard deviation of the specified column

🡪 Need to make changes to data selection script to account for changes to DataFrame class

**12/21/16**

Added **change\_col\_data(col, data)** method to directly change the data in a particular column

Also added **check\_input\_col\_length(arr)** method to aid in quality control check for inputs

Should also add a **tab** method—converts the DataFrame to a tab-delimitted string, that can be printed, then copied directly into an Excel spreadsheet without the need for creating an intermediate CSV file (to .**to\_s** method is for display only)

🡪 Should return the string, which the puts method can then be used to show—that way, this provides an alternate way of generating a tab-delimitted table that can be written directly to a file

The way it is currently checking for inconsistent data types in a column, it is changing the data, then throwing an error if they are inconsistent

🡪 Instead, it should check for consistency, and if they are consistent, then update, but if they are not consistent, throw an error and not update (values nor dtypes)

* It should be this way to avoid unknowingly modifying the data if using the DataFrame in the terminal

Another thing to note: when it displays the table data, nil values are converted to ‘’—this could potentially be confusing

Fixed **std(col)** method

Another problem with **std** method: it is calculating the population standard deviation, whereas Pandas calculates the sample standard deviation

These are exactly the same, except the population standard deviation divides by the number of data points, whereas the sample standard deviation divides by the number of data points minus one

Why does Pandas use the sample standard deviation instead of the population standard deviation?

**🡪 std** now calculates the standard deviation of the sample

**🡪 stdp** now calculates the standard deviation of the population

**tab** method created, although it doesn’t seem to work pasting the output into my version of excel

**12/22/16**

Need to make **value\_counts(col)** method output the value counts ordered from highest frequency to lowest frequency

🡪 Also need to add normalize option, which will cause it to print the decimal %s instead of the actual counts

🡪 Should add **length** and **size** methods that do the exact same thing as the **nrows** method currently does

**4/7/17**

🡪 Need to address issue where it will throw an error if you feed the constructor an empty array

🡪 Should make the methods to add a column and update a column’s values the same method

* Could call the method “set”

**4/19/17**

🡪 Add “length” and “size” methods that return the same value as the “nrows” method currently does (the number of rows in the data frame)

🡪 Research how to make using the data frame more like using a Pandas data frame (like using df[‘col’] = […] to set column values, using df[df[‘col’] > x] to select a subset of the data frame, etc.)

**4/21/17**

Add recode method which takes a column name and a hash of values to recode

* Have option to test all values that are not found in the keys of the remap hash to ‘’/nil, or a specified value, or leave unchanged

**10/11/17**

Revise value\_counts method to do the following:

* Don’t have issue when the length of one of the values is greater than 30 characters
* Have option to return normalized counts
* Sort returned values by highest frequency
* Make more efficient
  + Vc = Hash.new(0)
  + Iterate over column and add 1 for each value in array to vc
  + Return if return only desired
  + If print desired:
    - Sort vc by the values of the keys
    - Iterate over each key-value pair and print with 4 spaces between the key and value of the widest value (need to scale the lines at the top and bottom to address this
* Add option to not return data (if show only is desired)—make default
* Add option to sort ascending (descending is default)
* Remove option to not include nil values in value counts
* Sort alphabetically within value counts with the same value?
  + This is turning out to be a bit complicated, so let’s not worry about right now?
  + I might do a major, non-backwards-compatible revision for Ruby data frames, so it might not be worth it to put too much effort into it right now

**2/6/18**

Add method to return a given column (as an array) when brackets are used, just like a hash:

Df[col] 🡪 returns col

Add method to set the value of a given column using df[col] = new\_col\_vals

**2/7/18**

Should also add ability to add a new column using the same method

Add ability to compute the median of a column