Introduction – Purpose of Research

As the scale of demand for information exponentially increases it is important for departments involved in gathering and using such to use technologies and build workflows that can withstand such. Before beginning the redevelopment of any process important questions to ask include: what parts of my current process are absolutely needed and what parts are simply features that were added but are never used? If I add new technologies (or levels to any incoming categorical data) what is going to happen when the input files are processed by my workflow? Will these technologies be kept track of, will this break my workflow, or will they simply be ignored? Will any associated calculations be impacted by this new incoming data? How much effort am I going to have to put into my workflow to account for new categorical levels?

Answering these sorts of questions will give a team an idea as to how stable their current workflow is. There are some cases where solutions have been addressed for these types of questions, but the questions still apply to those solutions as well. As new information not previously accounted for is introduced into the input files, how complicated will it be to keep up to date on the implemented solutions? In some cases, the more robust solutions become the more complex and difficult to manage they become as well. It is my intention in this article to demonstrate the need to create processes flexible enough to give more tracking responsibility to the actual programs and less to the employees. Code that is dynamic and general enough allows for employees to spend less time in making tedious adjustments to code and more time being effective in other aspects of their departments.

Discussion – Dynamic Approach to Recoding Reference Values for Analytics

Problem: Keeping Up to Date on Key Raw Reference Values

There are instances where raw reference names need to be recoded a certain way for later analytical and presentation purposes. A simple approach to this problem could be to build a series of lists that contain these needed values or perhaps even dictionaries so that there is a means to map the raw input values to their associated recoded value. What happens when new raw reference names are introduced in the raw input files that could not have been accounted for when the lists (or dictionaries) where created? It can become tedious work to manually update needed lists, as well as any associated parts of a process that reference these values. This leave opportunity for technical error as well.

Solution Part 1: Build Structure that Offers Automatic Updates

Independent lists, or preferably dictionaries, should be stored in their own independent scripting files (or plain .txt if preferred) which are then imported into the scripts involved in a process. This allows for users to integrate flexibility of automating the option to update these lists. These dictionaries need to be stored in their own files in order to update them automatically because scripting languages (R, Python, Matlab, etc) will not allow for saved changes to be made to these dictionaries if the file in which they are stored in currently in use.

Solution Part 2: Implement Automatic Updates into Workflow

Now that we have the structures in place to automate dictionary (or list) updates it is time to discuss how to go about updating these dictionaries. It is important to first check the values of the various columns (note: just keep the natural order of columns getting recoded as is). If you were thinking ahead and decided to go with building dictionaries out of your reference values this can be easily accomplished by comparing the unique values of a given column against the keys of the associated dictionary. If there are no new values found then the process should run silently so to speak (that is, just operate as it normally does).

If there are new values found in this check then create a process that extracts the new unique values found against the comparison mentioned. These (1 at a time) should be printed to the user through a prompt that asks if the user would like to make changes to the current raw input value. If the user wishes to then they should simply be able to enter the new name as the program is running (otherwise say they would not). The program should then validate the inputted information and reassure with the user on their decision. No list should be updated without the user signing off on their inputted decision. After this is done the program can take both the inputted raw value and its newly associated recoded value and append both to the associated dictionary. It is important to update the dictionaries associated to each column that experiences value recoding so that the user is not prompted with this update each time the program is ran. They should only be prompted precisely when new values are introduced to a process.

Solution Part 3: Generalizing Calculations and Other Aspects of a Workflow

There is a lot of power and flexibility that comes with developing a dynamic workflow for analytics. Once you have a means to accessing all the needed recoded reference values you can start to use other powerful tools that allow for more flexibility in other aspects of a given workflow. For instance, let us say there is a reference value that is associated to a particular geographic location. Let us say that there are several variations of this reference value that are associated to neighboring geographic locations (or geographic locations grouped by similar characteristics). Let us take this example one step further and say for some equation that is involved in generating numbers needed for analytics involves assigning certain weights to these groups of variations found in a particular column (or columns). Let us say the outcome of the equation is an assessment that accounts for the relationship of its parameters (inputted columns).

Since we have already taken the time to recode the needed reference values we can use tools such as regular expressions to “group” certain reference values (and their associated variations) to the various weights in the given equation needed.

Now let us say the weight assignment involves exact row matches of these reference values (regardless if they are variations of another value or not) and assigned weights are currently being hard-coded. This calculation can still be easily generalized. You could build a second series of dictionaries (or add another layer to the current dictionary) which holds the associated weight to the given recoded values (which could be an added layer to what the user inputs for new reference values introduced in the input files). With this information already processed prior to the calculations being conducted there

could simply be a loop put in place on the dictionary's keys and associated values against the unique values found in each column (or columns).

Summary:

As you can now imagine there is a lot of flexibility that comes with building dynamic workflows for analytics. If anything, it should now seem like building dynamic processes makes generalizing the other aspects of a workflow easier than manually making updates! There is no longer a need to update equations or lists as our process now does that for us. Building a process in a manner such as the one described offers real-time scalability. Considering this discussion from a business standpoint you can also imagine that having dynamic processes such as these improves department productivity significantly, which is a metric used to gain additional funding for new technologies or projects. Instead of having meetings and hours spent on tedious updates employees can now leave more of these responsibilities on their programs, which leaves them more time to effectively contribute to other aspects of their department.

Illustration – Dynamic Workflow Process Map