Putting Together a Working Version of HitWalker2: The CCLE Example

Daniel Bottomly, Beth Wilmot and Shannon McWeeney

February 10, 2015

1 Database population

Database population is currently done using R in conjunction with the HitWalker2/populate/populate_hw2.R script. Below is the code used to create the 'base' database which others can use as a backbone.

```
> source("populate_hw2.R")
> system("wget ftp://string-db.org/STRING/9.1/mapping_files/Entrez_mappings/entrez_gene_id.vs.string.v9.05.28122012.txt")
> ppi.map <- read.delim("entrez_gene_id.vs.string.v9.05.28122012.txt", sep="\t", header=TRUE, stringsAsFactors=FALSE)
> names(ppi.map) <- c("entrezID", "stringID")
> #need to limit inclusion to only those entrezIDs that can be mapped back to Ensembl genes..
> #ppi.map <- ppi.map[ppi.map$entrezID %in% gene.ent$entrezID,]
>
> load.neo4j(.data=ppi.map, edge.name="MAPPED_TO", commit.size=10000L, neo.path="/Users/bottomly/Desktop/hitwalker2_paper/neo4j-community-2.1.6", dry.run=F)
> #string ppi
>
> system("wget http://string-db.org/newstring_download/protein.links.v9.1/9606.protein.links.v9.1.txt.gz")
> string.ppi <- read.delim("9606.protein.links.v9.1.txt.gz", sep="", stringsAsFactors=F)
> string.ppi <- string.ppi[(string.ppi$protein1 %in%  ppi.map$stringID) & (string.ppi$protein2 %in%  ppi.map$stringID),]
> names(string.ppi) <- c("stringID", "stringID", "score")
> load.neo4j(.data=string.ppi, edge.name="ASSOC", commit.size=10000L, neo.path="/Users/bottomly/Desktop/hitwalker2_paper/neo4j-community-2.1.6", dry.run=F, unique.rels=F)
> #then add in the pathway info, converted to entrez IDs
>
> path.dta <- read.pc.gmt(filename="Pathway Commons.4.All.GSEA.gmt", organism.code="9606")
> #here will keep database as a property of the pathway node and will only keep non-NA relationships with entrez IDs
>
> load.path.dta <- path.dta[,c("pathway", "entrezID", "database")]
> names(load.path.dta)[3] <- "pathway.database"
> load.path.dta <- load.path.dta[complete.cases(load.path.dta),]
> #there are duplicated entries and uniqueness constraints so will also need to deal with it ahead of time...
>
> load.path.dta <- load.path.dta[!duplicated(load.path.dta),]
> load.path.dta$pathway <- paste(load.path.dta$pathway, paste0("(", load.path.dta$pathway.database, ")"))
> load.neo4j(.data=load.path.dta, edge.name="PATHWAY_CONTAINS", commit.size=10000L, neo.path="/Users/bottomly/Desktop/hitwalker2_paper/neo4j-community-2.1.6", dry.run=F)
>
```

2 Configuration

There are two main files that need to be modified for `HitWalker2` to work with a custom database: `config.py` and `custom_functions.py`. The most important file of which is `config.py`. In `config.py`, many aspects of `HitWalker2` can be modified including customizing how the data contained in the database should be utilized to perform the various calculations contained within the program. Although everything that is needed can be supplied to `config.py`, we find it useful to separate configuration variables from any functions that might be pointed to as part of those variables. For example, the paradigm utilized by HitWalker2 (as well as other programs) is the specification of a query, the variable(s) it needs to run and a 'handler' function that takes the result of the query and returns a Python object which can be utilized downstream. We specify these handler functions in `custom_functions.py`.

2.1 User Specified Parameters

Each instance of `HitWalker2` can allow users the ability to modify how any querying and/or prioritization is performed by exposing parameter values for them to choose from. These parameters are specified in the `adjust_fields` dictionary in `config.py`. Each item in the dictionary should be keyed by the user-viewed name of the set of parameters. These items are themselves dicts of dicts, with the first level indicating the 'type' of parameters as well as the applicable fields. The first 'type' is the `standard` type. The items in the 'fields' dict should be keyed by a unique ID for that variable and contain the following information:

Key	Description
type	One of 'numeric' or 'character' indicating what type of input to expect from the user. A choice of numeric, will let the user input a integer or floating point value in a text box while a choice of character will let them choose from one of several options via a radio box.
default	The default value the field should take.
range	For numeric value, a two element list containing the min and max the values can take. For type = 'character', a list of the values the radio boxes can take.
comparison	For numeric values, one of '<', '>' or '=' indicating how the corresponding database value should be compared to the user input.
name	A human readable name to be displayed as a label for the input.

Below is an example for the CCLE data:

```
adjust_fields = {
    'General_Parameters':{'type':'standard',
                          'fields':{
                               'string_conf':{'type':'numeric', 'default':.4, 'range':[0, 1], 'comparison':'>', 'name':'HitWalker␣STRING␣Confidence'},
                               'path_conf':{'type':'numeric', 'default':.95, 'range':[0, 1], 'comparison':'>', 'name':'Pathway␣STRING␣Confidence'},
                               'res_prob':{'type':'numeric', 'default':.3, 'range':[0,1], 'comparison':'=', 'name':'Restart␣Probability'},
                               'max_iter':{'type':'numeric', 'default':100, 'range':[0, 10000], 'comparison':'=', 'name':'Max␣Iterations'},
                               'conv_thresh':{'type':'numeric', 'default':1e-10, 'range':[0,1], 'comparison':'<', 'name':'Convergence␣Threshold'},
                               'expression':{'type':'numeric', 'default':.75, 'range':[0,1], 'comparison':'>', 'name':'Expression␣(Hit)␣Threshold'},
                          }
                         }
}
```

2.2 Sample Selection

Note that currently there is not support for multiple valid samples per experimental unit.

2.3 Selection of Hits

Note that it is important that the terminologies are the same amongst sample_rels_query and hit_session_dict (as well as throughout). If hits are not being retrieved as expected, then this is likely to be the cause.

2.4 Selection of Variants

```
>     print('hello')
```