RDS is used to estimate the relative sizes (proportion) of subgroups within a population. The basic purpose of the simulation environment is to test the bias and variance of RDS estimates with different network structures. Homophily is known to impact the variance and bias. Differences in mean degree or degree distributions between the groups are also expected to impact the accuracy of some versions of the estimator.

Our basic strategy to create homophily is to construct a 2 group network with a variable width bridge between the groups. We begin with two distinct networks, potentially of different sizes and with some degree and attribute distribution. These networks are merged into a single network with two components. The ties in the networks are then rewired at random using the double edge swap. (I would like to improve this to only rewire intragroup ties to intergroup ties).

Most of the code assumes a directory structure where a directory called "RDS" is at the root level and contains 3 directories: code, samples and estimates.

```
Root '/'
    RDS/
        code/
           options_files/

        estimates/
        samples/
```

I added subfolders to the code directory I sent you. Right now, the import statements in the code are not updated to reflect the reorganization. Previously, all the files in /code except the options files were not organized in folders. Moving forward I want to migrate to the more organized structure I used for the code I sent you.

The current system involves using Python to generate samples and using R to process those samples and generate estimates. A set of samples may be 10 to 80 GB. There is a lot of overhead associated with writing and then reading that much data so we are working on a solution to generate the estimates directly in Python or to use a python-java bridge to leverage Doug's RDSAT program.

I also have some experimental versions of the code that I usually implement in R. One uses R to send data to RDSAT for processing. This is helpful because RDSAT can generate bootstrapped confidence intervals many times faster than R. Patrick and Yongren are working on a python version of this functionality (to eliminate the need to write and read the samples files). I am also experimenting with different bootstrapping techniques that produce more accurate variance estimates.

All the R code is in the est_R folder and is pretty straightforward. It processes a file containing an arbitrary number of samples separated by lines with only '----'. It processes samples one at a time, produces the indicated estimates and then writes the columns to an output file. The processing code is in the net_est* files (each file produces estimates in a slightly different way) and the base code for RDS analysis in in the RDS3.r file.

A simulation might be started from the command line as follows:

```
%python runSim_cjc73_batchcode.py  options_file.py
```

options_file The options file contains many experiment-specific settings. We use these to avoid having to write a new runSim for each experiment	**option_file** defines**:** Output location Rewire levels Sampling fraction Number of Networks to generate Number of Samples to Draw from each Network Number of seeds Method to select seeds Number of recruits Method to select recruits Sampling with or without replacement The method of permuting (rewiring) the networks A generator function that yields networks for RDS to simulate over See the contents of the RDSim directory

runSim A file that controls the type of simulation to be executed. Option file to executed into the local namespace of the runSim, so it is as if the contents of the options_file were part of runSim. Default values are defined in runSim and overridden by options file.	**runSim** executes**:** Creates a **simrdclass** object for each network in the network generator defined in options file using the settings defined by the options files. Permutes the networks according to the option file parameters. Generates a trial for each combination of the settings defined. Writes raw samples to a large *samples* file. After all samples are generated, runSim calls Rscript on one of the net_est* files in the est_R directory and passes the newly generated sample file as an argument. R will execute net_est*, generating the appropriate RDS analysis and writing the results for each sample on a single line in an *estimates* file