FRED Quick Start Guide

John J. Grefenstette, PhD

Public Health Dynamics Laboratory
Graduate School of Public Health
University of Pittsburgh
www.phdl.pitt.edu

27 Jan 2012

Acknowledgments

This work was supported by the National Institute of General Medical Sciences MIDAS grant 1U54GM088491-01, and by the Vaccine Modeling Initiative, funded by the Bill and Melinda Gates Foundation. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Introduction

The accelerating growth in data availability and corresponding advances in high performance computing present new opportunities for *in silico* analysis of complex public health questions using computational modeling and simulation. FRED (A Framework for Reconstructing Epidemiological Dynamics) is an open source, modeling system developed by the University of Pittsburgh Public Health Dynamics Laboratory in collaboration with the Pittsburgh Supercomputing Center and the School of Computer Science at Carnegie Mellon University. FRED supports research on the dynamics of infectious disease epidemics, and the interacting effects of mitigation strategies, viral evolution, and personal health behavior. The system uses agent-based modeling based on census-based synthetic populations that capture the demographic and geographic distributions of the population, as well as detailed household, school, and workplace social networks. Multiple circulating and evolving strains can be simulated. Mitigation strategies in the framework include vaccination, anti-viral drugs, and school closure policies. FRED supports models of health behavior change to facilitate the study of critical personal health behaviors such as vaccine acceptance, personal hygiene and spontaneous social distancing. FRED is available through open source in the hopes of making large-scale agent-based epidemic models more useful to the policy-making community, the research community, and as a teaching tool for students in public health.

System Requirements

FRED is written in C++ for UNIX-like systems. It was developed primarily using gcc v. 4.2.1 but should be compatible with any other ANSI compliant C++ compiler. FRED was developed on OS X and Linux systems and requires that the following components be installed prior to use:

- gcc
- make
- Perl
- python
- gnuplot

FRED has also been test on Windows running Cygwin (www.cygwin.com). If using Cygwin, the following installation packages are recommended:

- Development
- Editors
- Graphics
- Perl
- Python
- Shells
- X11

Obtaining FRED

The FRED source distribution is available from the University of Pittsburgh MIDAS Center of Excellence at www.midas.pitt.edu/DownloadFRED/. To obtain the software, first click on the Register link to create an account. You must accept the FRED License Agreement (see Appendix) to register your account. Once you have registered, log in and click on the Download menu option that will take you to the download portal. Under the Download tab, you can select a version of FRED to download. The distribution is in the form of a gzipped tar file.

You can also download FRED documentation, and synthetic population files from the same web page.

Installation

Extracting the FRED tar file will create a directory called FRED in the current directory:

```
% tar –xvzf FRED-archive.tgz
x FRED/Makefile
x FRED/LICENSE
...
```

Environmental Variables

Set the environmental variable `FRED_HOME` to the location of the top-level FRED directory, for example:

```
% setenv FRED_HOME $HOME/FRED
```

FRED includes a directory `$FRED_HOME/bin` that contains supporting scripts, so it is recommended that you add the bin directory to your runtime path, e.g.,

```
% setenv PATH "${FRED_HOME}/bin:$PATH"
```

It will be convenient to add the following lines to your `.profile` if using bash or similar shell:

```
FRED_HOME=/path/to/FRED
PATH=$FRED_HOME/bin:$PATH
```

If using csh, add the following lines to your `.cshrc` file:

```
setenv FRED_HOME /path/to/FRED
setenv PATH "${FRED_HOME}/bin:$PATH"
```

where `/path/to/FRED` gives the path to the FRED top-level directory. e.g. `$HOME/FRED` if you installed FRED in your home directory.

The environmental variables `FRED_GNUPLOT` and `FRED_DISPLAY_PNG` are optional, and if set, enable scripts that can create and display curves associated with FRED output. Set the environmental variable `FRED_GNUPLOT` to the location of the gnuplot executable, if it is installed on your machine. For example,

```
setenv FRED_GNUPLOT /usr/local/bin/gnuplot
```

Set the environmental variable `FRED_DISPLAY_PNG` to the location of the executable that displays a PNG file on the terminal, if one is installed on your machine. The following example is for OS X:

```
setenv FRED_DISPLAY_PNG /usr/bin/open
```

For Cygwin, you might try:

```
setenv FRED_GNUPLOT /usr/bin/gnuplot
setenv FRED_DISPLAY_PNG /usr/bin/display
```

Also for Cygwin, to be able to use ImageMagick for displaying figures, you have to use the shell within the Xwindow by using the following command

```
% startxwin
```

FRED Directory Structure

The following subdirectories are under **$FRED_HOME:**

bin/ : contains run-time scripts to manage FRED experiments. The FRED executable also gets copied into here.

doc/ : FRED documentation files.

input_files/ : Contains the default parameters file **params.default** and example input files for various features. **These files should not be changed by the user.** If you want to modify these files, copy them to a new working directory and edit them as desired. Then add parameters to your working **params** file to point to the new input files. See **params.default** for examples.

region/ : Contains population and location files for different regions, and scripts to create population input files for FRED.

src/ : Contains all of the sources files for the FRED executable, the Makefile, and the testing directories.

tests/: FRED regression tests.

Compiling FRED

```
% cd $FRED_HOME
% make
```

This should result in the creation of an executable file called FRED in the $FRED_HOME/bin directory.

To check your installation of FRED, run the regression test script. If you don't see any error messages, your installation is complete. A successful test looks something like this:

```
% rt
FRED regression test: base
run 1 ... run 2 ...
comparing results ...
cmp OUT.TEST/out1.txt OUT.RT/out1.txt
cmp OUT.TEST/out2.txt OUT.RT/out2.txt
cmp OUT.TEST/infections1.txt OUT.RT/infections1.txt
cmp OUT.TEST/infections2.txt OUT.RT/infections2.txt
regression test finished.
```

Run-time Parameters

The run-time parameters for FRED are contained in two parameter files. The first file is **$FRED_HOME/input_files/params.default** and contains the default values of all defined run-time parameters. **This file should not be modified.** The second file is usually called **params** and contains any parameter values that override the default values. The **params** file may be empty.

Both files have the same format. Lines that begin with a "#" character are considered comments and are ignored. Parameters with scalar values are specified with lines of the form:

```
<name> = <value>
```

For example:

```
days = 100
diseases = 1
popfile = pop_Alleg.txt
```

Some parameters are vector valued, in which case the format is:

```
<name> = <size> v_1 v_2 ... v_size
```

For example:

```
# cdf of trip duration in days
travel_duration = 6 0 0.2 0.4 0.6 0.8 1.0
```

If a parameter appears more than once in a parameter file, the last setting takes precedence. If a parameter appears in both **params.default** and **params**, the value in **params** overrides the value in **params.default.**

Running FRED

The FRED program takes an optional command line argument, the name of the run-time parameters file:

```
% FRED parameter_file_name
```

If the argument is omitted the name “**params**” is assumed.

More information on the input and output files appears in the FRED User’s Guide. In addition, a set of scripts is provided for managing the process of running a large number of simulations with FRED. These scripts are illustrated below and are described fully in the User’s Guide Section RUNTIME MANAGEMENTS SCRIPTS.

Simulation Information Management System

There are several options for running FRED. The FRED executable is copied to the $FRED/bin directory after each make, so you can run FRED as follows from any working directory, assuming that you have added `$FRED_HOME/bin` to your path:

```
% FRED [paramfile [run_number [directory]]]
```

The arguments are optional from right to left. If all three arguments are given, FRED uses the given paramfile, runs a single replication with number "run_number", and writes output files to the given directory. The output directory is relative to the current working directory.

If the third argument is omitted, the output directory is taken from the runtime parameter "`outdir`", with default value `OUT`.

If both the second and third arguments are missing, run_number defaults to 1.

If all arguments are missing, paramfile defaults to "params".

Examples:

```
# run FRED on file params and write output files to ./OUT:
% FRED

# run FRED on file params.foo and write output files to ./OUT:
% FRED params.foo

# run FRED on file params with run number = 2
% FRED params 2

# run FRED on file params.foo
# with run number = 2 writing output files to ./OUT.foo:
% FRED params.foo 2 OUT.foo
```

Using the `run_fred` script for multiple realizations

The **`run_fred`** script is provided to perform multiple realizations (runs) in a local directory. Each run uses a distinct seed for the random number generator, so the results will vary from run to run. The format is:

```
% run_fred -p paramfile -d directory -s start_run -n end_run
```

The order of the arguments doesn't matter, and all arguments have default values:

```
-p "params"
-d ""
-s 1
-n 1
```

For example, the command:

```
% run_fred -p params -d FOO -s 1 -n 3
```

translates to a set of commands:

```
% FRED params 1 FOO > FOO/LOG1
% FRED params 2 FOO > FOO/LOG2
% FRED params 3 FOO > FOO/LOG3
```

after first creating directory FOO if necessary. The run_fred script also copies the params file into the output directory, for future reference.

If -d is not specified on the command line, FRED writes output files to the output directory specified in the **outdir** runtime parameters, which default to **OUT**. For example, if params does not specify an output directory, then

```
% run_fred -n 3
```

translates to:

```
% FRED params 1 OUT > OUT/LOG1
% FRED params 2 OUT > OUT/LOG2
% FRED params 3 OUT > OUT/LOG3
```

The random seed for each run is set based on the both the seed value in the params file and on the run number, so a collection of FRED runs can be executed in any order with the same results. For example, you should get the same results in the output directory from

```
% run_fred -n 20
```

as from:

```
% run_fred -n 10
% run_fred -s 11 -n 20
```

FRED runtime management scripts

The $FRED_HOME/bin directory includes several commands to manage the process of running FRED jobs. Commands exist for starting FRED jobs, reporting the status of those jobs, and organizing and reporting the results files. The bin directory contains the following commands:

`fred_job` -- runs FRED and stores all associated data in a results database
`fred_AR` -- report on the the attack rate of a simulation
`fred_clear_all_results` -- flush the results database
`fred_delete` -- delete a single job from the results database
`fred_display_plot` -- display one or more curves
`fred_jobs` -- show that status of all jobs in the results database
`fred_plot` -- plot a curve
`fred_plot_data` -- retrieve the data associated with a curve

`fred_report` -- create statistical summaries of output variables

`fred_status` -- report the status of a single job

`fred_sweep` -- run a set of simulation changing the value of a variable

`fred_tail` -- show the tail of the current output file

`get_distr` -- show the distribution of infection locations

`ch` -- change a parameter value in a params file

`p` -- print out the current params file

`rt` -- run regression test

To use these commands, set the environmental variable `$FRED_HOME` to the location of your FRED distribution. Then add `$FRED_HOME/bin` to your path. The following are most likely to be the most useful commands when starting to use FRED. All commands are described in more detail in the User's Guide.

```
% fred_job [-p paramsfile | -k key]
```

Run FRED with the given parameter file in a working directory created in the `$FRED_HOME/RESULTS` directory, and associate the working directory with the key. If the `-p` option is omitted, the file "`params`" is assumed. If the `-k` option is omitted, an internally created key is generated. In either case, a `<key,id>` pair is printed on standard output, where `<id>` is the identifier of directory associated with the run (i.e. `$FRED_HOME/RESULTS/JOB/<id>`).

`fred_job` will terminate if the user supplied key has already been used. The script sets the STATUS of the request (see `fred_status` below). When FRED finishes, fred_job runs `stats` to collect data on the output variables in the outfile.

```
% fred_display_plot -k key -v [S|E|I|R|s|C|c|M|A|r]
```

Run fred_plot and then opens the resulting plot file.

```
% fred_jobs
```

Show that status of all jobs in the results database. The dates shown for FINISHED jobs reflect the time that they finished.

```
% fred_status -k key [-s secs]
```

Print the status of the FRED run associated with the given key. If `-s` option is given, repeats status report every `secs` seconds.

```
% ch param_name value [ paramfile ]
```

Edit the given paramfile (or "params" if no file is given) and add a line

```
param_name = value
```

First checks to see if the given `param_name` occurs in **params.default**.

```
% p
```

Print out the current params file.

```
% rt
```

Run regression test.

Tips for Creating FRED Working Directories

Users may find it convenient to create a new working directory when experimenting with FRED. Assuming that you have created the environmental variables as described previously, FRED can be executed in any directory. The only required file is **params**, which may be empty. For convenience, you may want to copy the files in the directory `$FRED_HOME/input_files` to a new FRED working directory:

```
% mkdir test
% cd test
% cp $FRED_HOME/input_files/params params.foo

 <edit params.foo to includes the desired settings>

% fred_job –p params.foo –k test
```

Note also that some of the files provided in **%FRED_HOME/input_files/** are specific to Allegheny County, Pennsylvania, and may need to be revised for other locations. The location specific files are:

- birth_rate.txt
- mortality_rate.txt
- seasonality_timestep

If you wish to modify any of the input files, copy them to your working directory and edit them, and then set the appropriate parameters to tell FRED to use the new files. See examples of file location parameters in **params.default.**

Tutorials

When running FRED from the command line, it is recommended that you use the scripts provided in the directory $FRED_HOME/bin. These scripts will store all metadata associated with each job in the FRED RESULTS database. This database is currently a directory-tree stored in the directory $FRED_HOME/RESULTS.

It is recommended that you run FRED jobs in a working directory, for example `$FRED_HOME/work`.

Tutorial 1. Running a baseline epidemic for Allegheny County.

Here is a sample parameter file:

```
% cat params.baseline
days = 100
```

This will run FRED for 100 simulated days using the default influenza parameters for Allegheny County. All runtime parameters not given in the parameter file will be given default values specified in file **params.default**.

To execute FRED, use the command

```
% fred_job –p params.baseline –k baseline &
```

The –p argument is the parameter file and the –k argument gives a user-defined name for the job. The run will run in the background.

You can check the status of the job using the fred_status command:

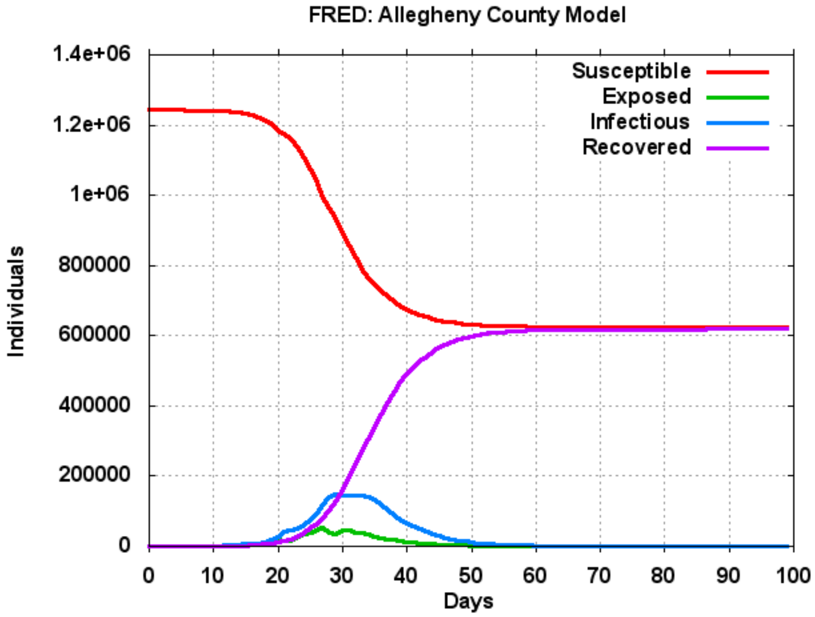
```
% fred_status –k baseline
RUNNING-00 Tue Nov 29 21:50:15 2011

% fred_status –k baseline
FINISHED Tue Nov 29 21:52:10 2011
```

The command fred_jobs shows the status of all jobs:

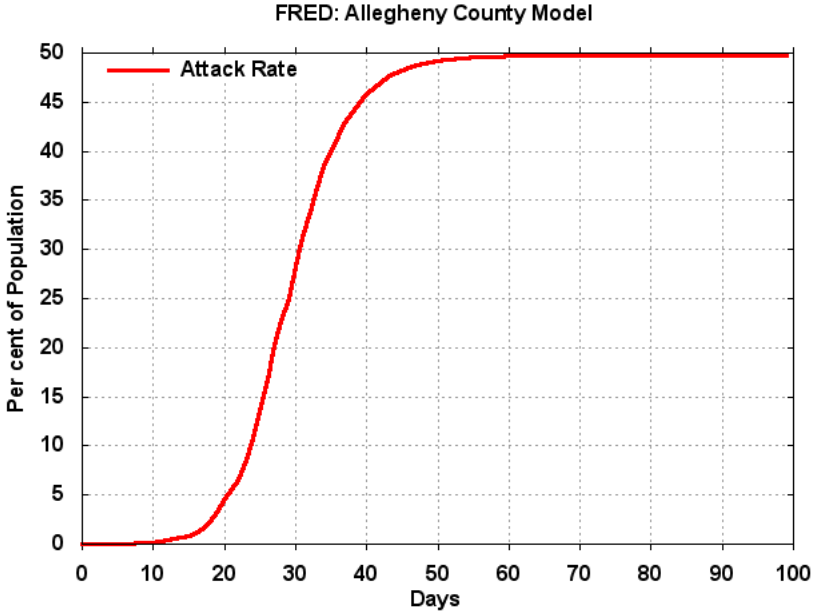
```
% fred_jobs
KEY = baseline   JOB =   1 STATUS = FINISHED Thu Aug  4 14:28:23 2011
KEY = stay-0.2   JOB =   5 STATUS = FINISHED Wed Sep 21 14:52:57 2011
KEY = stay-0.25  JOB =   8 STATUS = FINISHED Wed Sep 21 15:22:14 2011
KEY = stay-0.3   JOB =   9 STATUS = FINISHED Wed Sep 21 15:46:36 2011
KEY = stay-0.35  JOB =  10 STATUS = FINISHED Wed Sep 21 16:10:50 2011
```

When the job is finished, you can display results. The variable(s) to be display are listed after the –v flag.

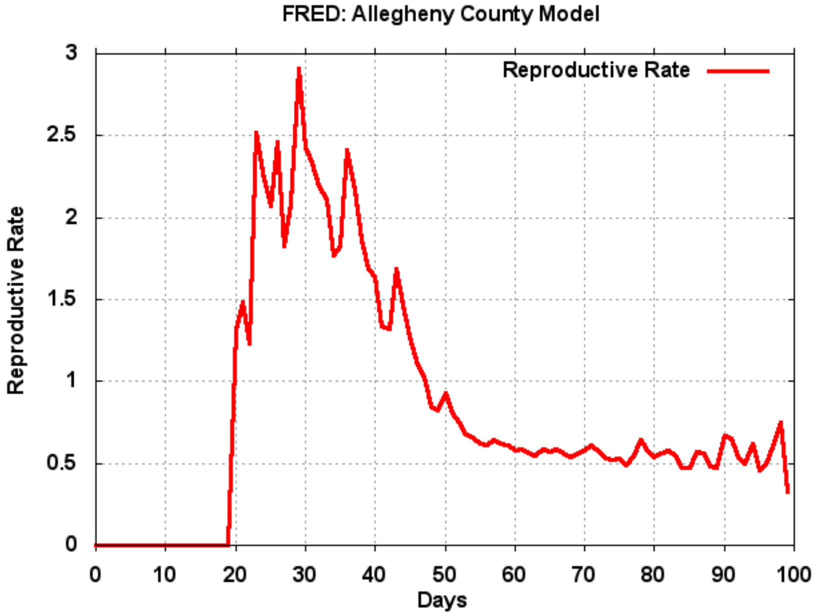
```
% fred_display_plot –k baseline –v SEIR
```



```
% fred_display_plot –k baseline –v A
```



```
% fred_display_plot –k baseline –v r
```



You can also get the data associated with a plot:

```
% fred_plot_data –k baseline –v A

0 0.01 0
1 0.01 0
2 0.01 0
3 0.02 0
4 0.03 0
5 0.03 0
.
.
.
95 49.75 0
96 49.75 0
97 49.75 0
98 49.75 0
99 49.75 0
```

Tutorial 2: Varying the transmission parameter

The transmissibility of disease 0 (the default disease) is indicated by the parameter trans[0], and has a default value of 1 for influenza (this is calibrated to a serological attack rate of 50%). We will define two new parameter files:

```
% cat params.weak
days = 100
trans[0] = 0.8

% cat params.strong
days = 100
trans[0] = 1.2
```

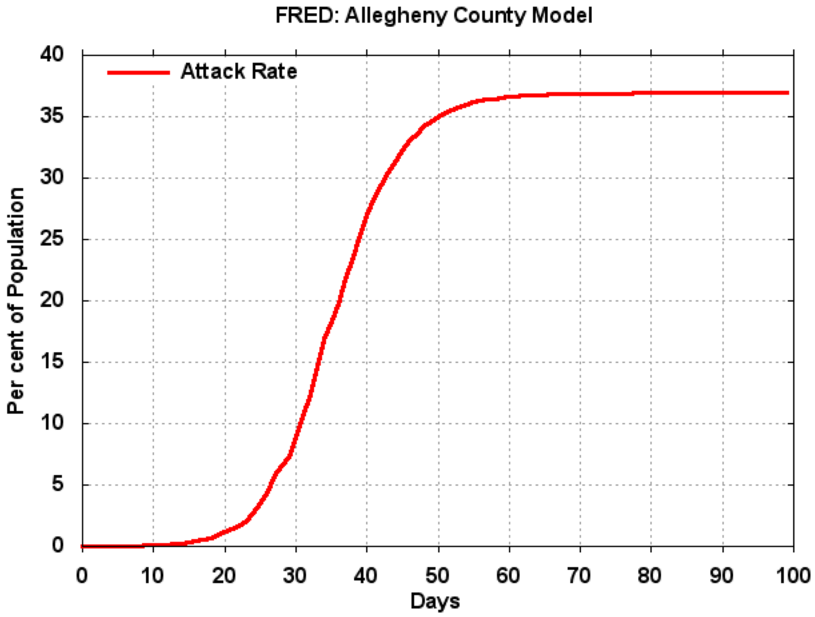
Now run two FRED jobs:

```
% fred_job –p params.weak –k weak

% fred_job –p params.strong –k strong
```

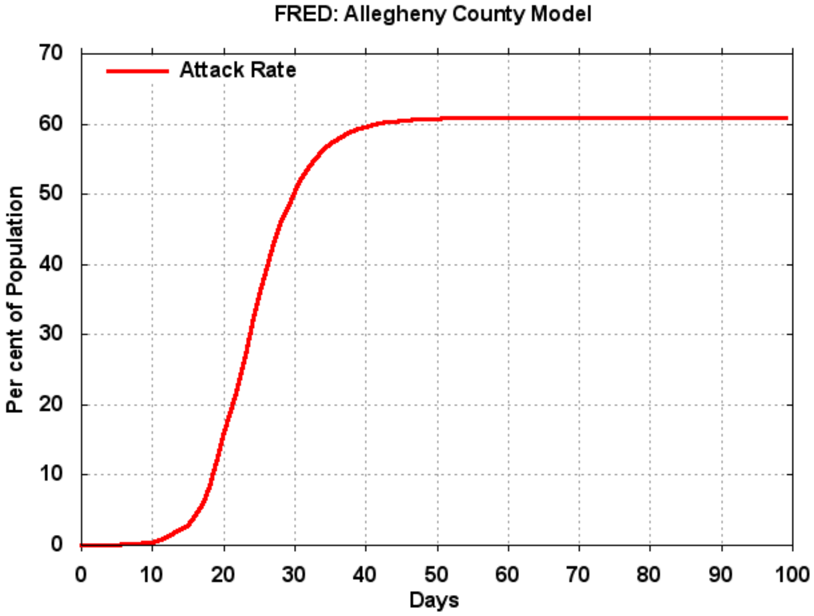
The jobs can be run in parallel is you have sufficient memory (approximately 2GB per job for Allegheny County).

When the jobs are finished you can display the results:

```
% fred_display_plot –k weak –v A
```

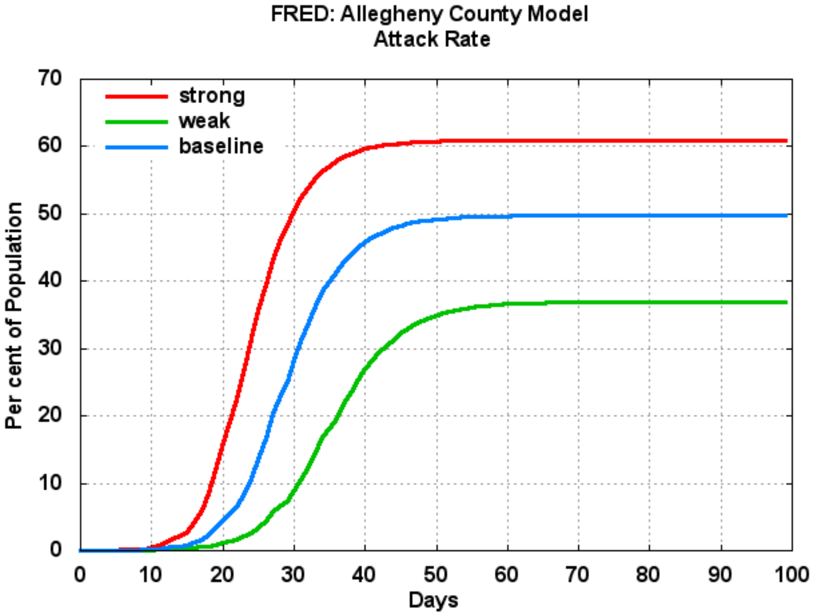


```
% fred_display_plot –k strong –v A
```

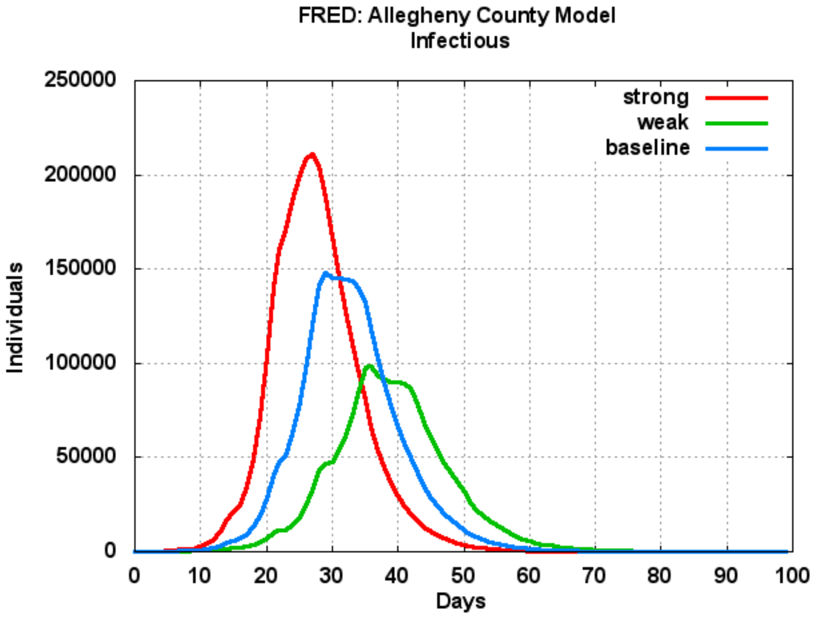


You can also display multiple plots on the same graph:

```
% fred_display_multiplot –v A –k strong weak baseline
```



```
% fred_display_multiplot –v I –k strong weak baseline
```



Tutorial 3: Varying a behavioral parameter

As an example of varying the behavior of FRED agent, we will change the probability that agents stay home when sick. The default is that 50% of the agents stay home when sick. We will define two new parameter files:

```
% cat params.stay-40
days = 100
stay_home_when_sick_strategy_distribution = 7 60 40 0 0 0 0 0

% cat params.stay-60
days = 100
stay_home_when_sick_strategy_distribution = 7 40 60 0 0 0 0 0
```

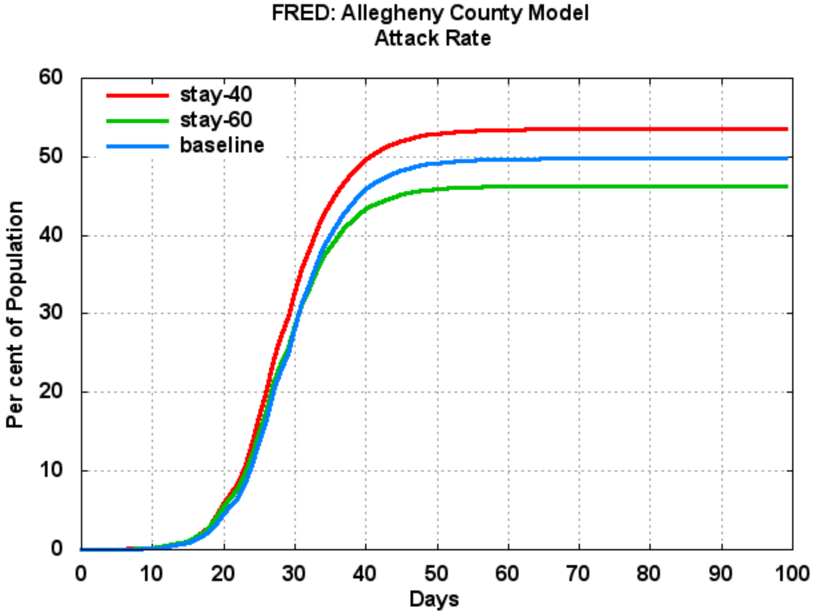
The first file says that 40% of the agents will always stay home when they have symptoms. The second file says that 60% of the agents will stay home when sick. See the FRED Users Guide for an exaplanation of the parameter format for the behavior strategy distributions.

Now run two more FRED jobs:

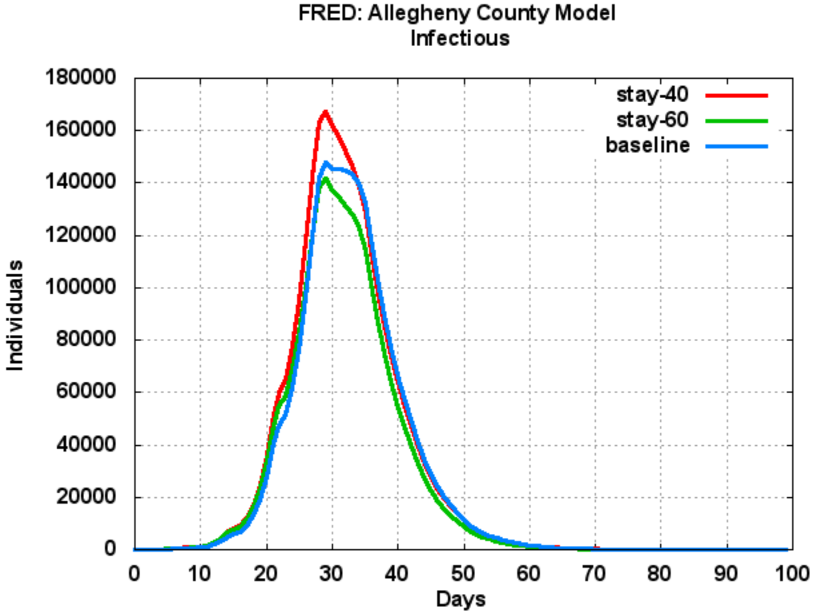
```
% fred_job –p params.stay-40 –k stay-40

% fred_job –p params.stay-60 –k stay-60
```

When the jobs are finished you can display the results

```
% fred_display_multiplot –v A –k stay-40 stay-60 baseline
```



```
% fred_display_multiplot –v I –k stay-40 stay-60 baseline
```



Further Exercises:

Explore combinations of FRED parameters by adding additional lines to the parameter files. See the FRED Users Guide for discussions of FRED parameters.

Appendix: FRED License Agreement

A license is hereby granted by University of Pittsburgh Graduate School of Public Health (“GSPH”), free of charge, to any person obtaining a copy of the software package called FRED and associated documentation files (the "Software"), to use, copy, modify and merge copies of the Software, subject to the following conditions:

1. You acknowledge and agree that the license granted hereunder is personal to you, and you will not under any circumstances sell, give, disclose, lend, or otherwise distribute the Software to third parties. You further agree that you will not use the Software for commercial purposes.

2. You acknowledge and agree that GSPH retains all ownership rights, including copyright rights in the Software and that by entering into this license, you do not acquire any such rights in the Software.

3. You acknowledge and agree that THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

4. If the Software is used to obtain a result, and that result is published in the public literature, then you agree to acknowledge its use of the Software in the following citation:

Copyright © 2011, University of Pittsburgh Graduate School of Public Health, John Grefenstette, Shawn Brown, Roni Rosenfield, Alona Fyshe, David Galloway, Nathan Stone, Bruce Lee, Phil Cooley, William Wheaton, Thomas Abraham, Jay DePasse, Anuroop Sriram, and Donald Burke.

5. You agree to indemnify and hold harmless GSPH from and against all damages, liabilities, attorney fees, and costs arising out of your use of the Software.