Manual for installing Stan for use with Stata on Windows

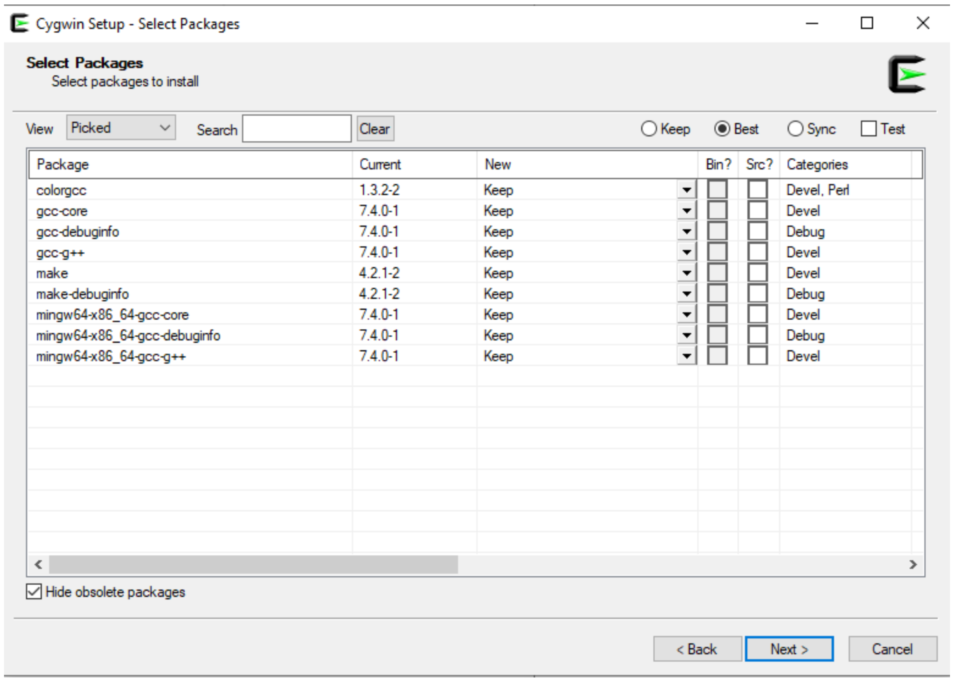
Installing CmdStan

Overview

Installing CmdStan requires a working c++ compiler on Windows. In contrast to MacOS and Linux, Windows does not come with one of those and needs to be installed first. You have two options: the visual studio c++ compiler and a gnu compiler in mingw. In our experience, the latter option is the slightly easier one to get up and running. Once you have installed a compiler, you can download and compile CmdStan (This is also a good opportunity to test whether the compiler works as it should). Once that is done you need to make sure that Stata can find CmdStan. Here are the necessary steps (Warning things can be slightly different depending on your version of Windows)

Downloading and installing a compiler

1. Go to https://www.cygwin.com/ and install the latest version (in most cases you probably want the 64-bit version)
2. Install Cygwin for all users
3. Once you started the installation (selected the mirror to download from etc.) you will be asked to select packages to install. Here you need to make sure that gcc and the make tools are selected.



Select Packages
Select packages to install

View: Picked   Search: Clear
Keep [✓ Best] Sync ☐ Test

Package	Current	New	Bin?	Src?	Categories
colorgcc	1.3.2-2	Keep	☐	☐	Devel, Perl
gcc-core	7.4.0-1	Keep	☐	☐	Devel
gcc-debuginfo	7.4.0-1	Keep	☐	☐	Debug
gcc-g++	7.4.0-1	Keep	☐	☐	Devel
make	4.2.1-2	Keep	☐	☐	Devel
make-debuginfo	4.2.1-2	Keep	☐	☐	Debug
mingw64-x86_64-gcc-core	7.4.0-1	Keep	☐	☐	Devel
mingw64-x86_64-gcc-debuginfo	7.4.0-1	Keep	☐	☐	Debug
mingw64-x86_64-gcc-g++	7.4.0-1	Keep	☐	☐	Devel

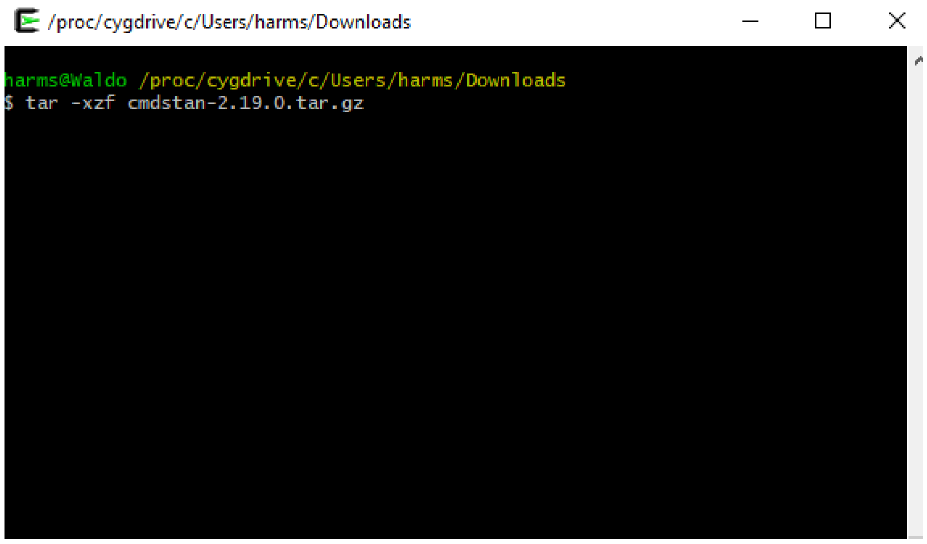
☑ Hide obsolete packages

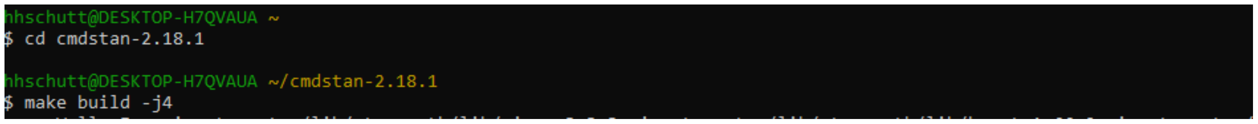
4. Finally, open the Cygwin console and type **g++ -v** to check whether the compiler is installed properly and can be found. If not, you probably need to add the location of the Cygwin binaries to your PATH environment variable in windows. (ONLY DO THIS IF NECESSARY)

Downloading and compiling CmdStan

NOTE: I am having problems getting stan programs to compile in using 2-19-1. It does work for me in 2.19.0 Try it with 2.19-0 first. Maybe 2.19-1 is broken on Windows. 2-19-0 does not work for me either. The only one that worked for me is 2.18.1 but also only the tar.gz file and NOT the zip! Not tried this with mac

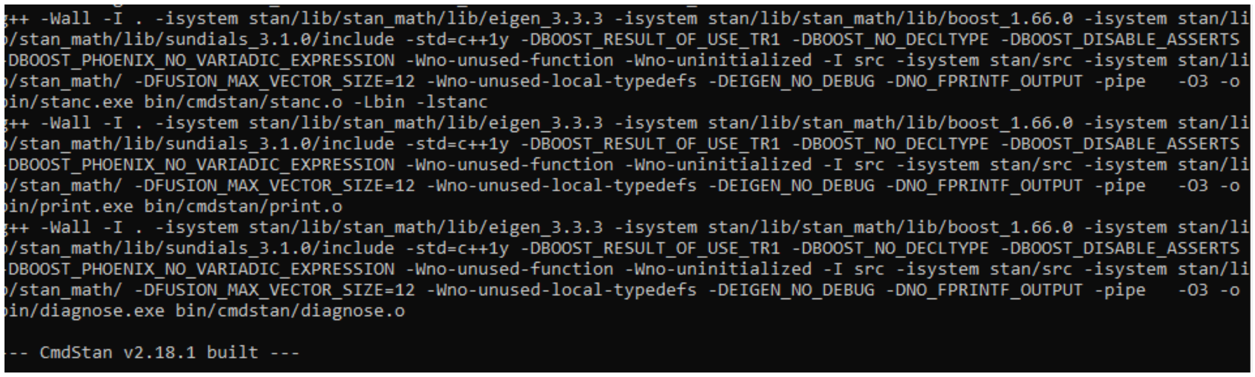
Note, nearly everything here is taken out of the CmdStan user guide (https://github.com/stan-dev/cmdstan/releases/download/v2.19.1/cmdstan-guide-2.19.1.pdf)

1. Go to https://github.com/stan-dev/cmdstan/releases and either git clone or download the CmdStan source code (e.g., download https://github.com/stan-dev/cmdstan/releases/download/v2.18.1/cmdstan-2.18.1.tar.gz if you are on windows)
2. Put it into a folder of your choice, then unpack it from Cygwin. In this example the file is still in the downloads folder which you can get to via cd /proc/cygdrive/c/Users/…/Downloads and so on. Then run the following tar command:
   ```
   > tar -xzf cmdstan-2.18.1.tar.gz
   ```
   

   ```
   harms@Waldo /proc/cygdrive/c/Users/harms/Downloads
   $ tar -xzf cmdstan-2.19.0.tar.gz
   ```
3. Compile from Cygwin issuing the command
   ```
   > make build -j4
   ```
   (-j4 means compile stan to use 4 cores. You can adjust this depending on the number of cores on your machine that you want to use) Warning: This build process can take 10+min and consume quite a bit of Ram.
   
   ```
   hhschutt@DESKTOP-H7QVAUA ~
   $ cd cmdstan-2.18.1

   hhschutt@DESKTOP-H7QVAUA ~/cmdstan-2.18.1
   $ make build -j4
   ```

4. The result should be something like:



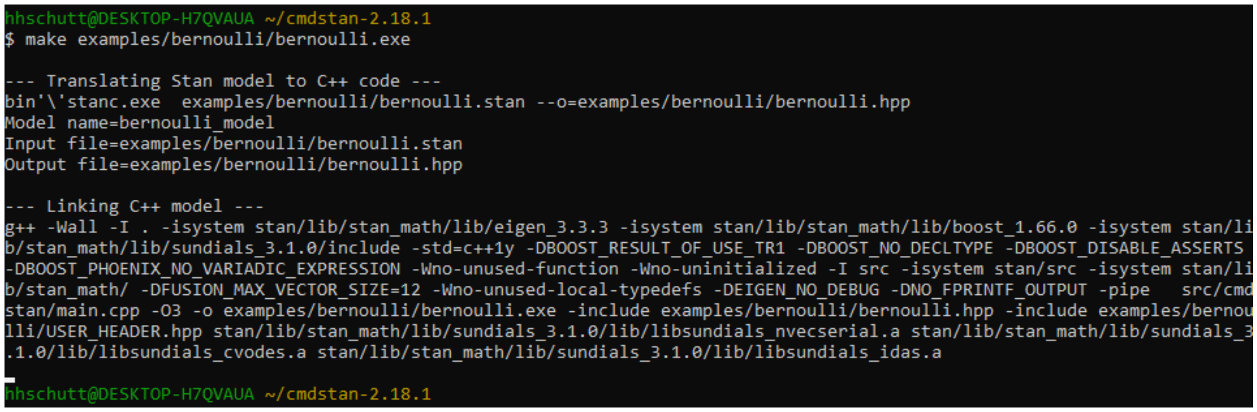
```
++ -Wall -I . -isystem stan/lib/stan_math/lib/eigen_3.3.3 -isystem stan/lib/stan_math/lib/boost_1.66.0 -isystem stan/li
/stan_math/lib/sundials_3.1.0/include -std=c++1y -DBOOST_RESULT_OF_USE_TR1 -DBOOST_NO_DECLTYPE -DBOOST_DISABLE_ASSERTS
DBOOST_PHOENIX_NO_VARIADIC_EXPRESSION -Wno-unused-function -Wno-uninitialized -I src -isystem stan/src -isystem stan/li
/stan_math/ -DFUSION_MAX_VECTOR_SIZE=12 -Wno-unused-local-typedefs -DEIGEN_NO_DEBUG -DNO_FPRINTF_OUTPUT -pipe   -O3 -o
in/stanc.exe bin/cmdstan/stanc.o -Lbin -lstanc
++ -Wall -I . -isystem stan/lib/stan_math/lib/eigen_3.3.3 -isystem stan/lib/stan_math/lib/boost_1.66.0 -isystem stan/li
/stan_math/lib/sundials_3.1.0/include -std=c++1y -DBOOST_RESULT_OF_USE_TR1 -DBOOST_NO_DECLTYPE -DBOOST_DISABLE_ASSERTS
DBOOST_PHOENIX_NO_VARIADIC_EXPRESSION -Wno-unused-function -Wno-uninitialized -I src -isystem stan/src -isystem stan/li
/stan_math/ -DFUSION_MAX_VECTOR_SIZE=12 -Wno-unused-local-typedefs -DEIGEN_NO_DEBUG -DNO_FPRINTF_OUTPUT -pipe   -O3 -o
in/print.exe bin/cmdstan/print.o
++ -Wall -I . -isystem stan/lib/stan_math/lib/eigen_3.3.3 -isystem stan/lib/stan_math/lib/boost_1.66.0 -isystem stan/li
/stan_math/lib/sundials_3.1.0/include -std=c++1y -DBOOST_RESULT_OF_USE_TR1 -DBOOST_NO_DECLTYPE -DBOOST_DISABLE_ASSERTS
DBOOST_PHOENIX_NO_VARIADIC_EXPRESSION -Wno-unused-function -Wno-uninitialized -I src -isystem stan/src -isystem stan/li
/stan_math/ -DFUSION_MAX_VECTOR_SIZE=12 -Wno-unused-local-typedefs -DEIGEN_NO_DEBUG -DNO_FPRINTF_OUTPUT -pipe   -O3 -o
in/diagnose.exe bin/cmdstan/diagnose.o

-- CmdStan v2.18.1 built ---
```

Testing CmdStan

To see whether it worked, we try to compile and run a program directly via CmdStan and not Stata. Stan code needs to be compiled before to make it fast. We need to build a program.

1. Before building any Stan program from CmdStan, change directories to your <cmdstan-home>.
2. We will try to build the example program coming with CmdStan. In the examples folder you will see a Bernoulli.stan file and some data files. We will compile the .stan file to an exe file using make:



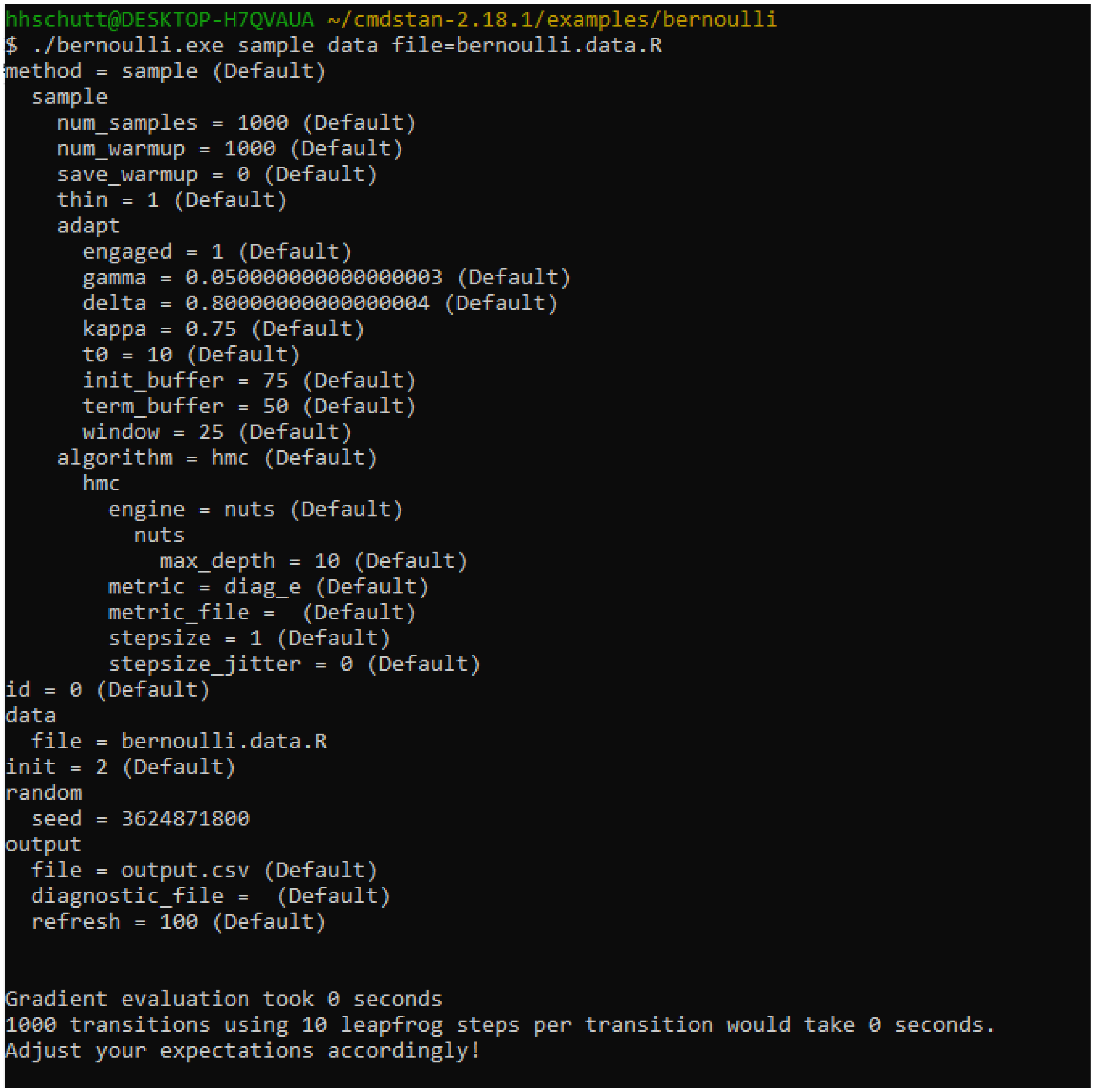
```
hhschutt@DESKTOP-H7QVAUA ~/cmdstan-2.18.1
$ make examples/bernoulli/bernoulli.exe

--- Translating Stan model to C++ code ---
bin'\'stanc.exe  examples/bernoulli/bernoulli.stan --o=examples/bernoulli/bernoulli.hpp
Model name=bernoulli_model
Input file=examples/bernoulli/bernoulli.stan
Output file=examples/bernoulli/bernoulli.hpp

--- Linking C++ model ---
g++ -Wall -I . -isystem stan/lib/stan_math/lib/eigen_3.3.3 -isystem stan/lib/stan_math/lib/boost_1.66.0 -isystem stan/li
b/stan_math/lib/sundials_3.1.0/include -std=c++1y -DBOOST_RESULT_OF_USE_TR1 -DBOOST_NO_DECLTYPE -DBOOST_DISABLE_ASSERTS
-DBOOST_PHOENIX_NO_VARIADIC_EXPRESSION -Wno-unused-function -Wno-uninitialized -I src -isystem stan/src -isystem stan/li
b/stan_math/ -DFUSION_MAX_VECTOR_SIZE=12 -Wno-unused-local-typedefs -DEIGEN_NO_DEBUG -DNO_FPRINTF_OUTPUT -pipe   src/cmd
stan/main.cpp -O3 -o examples/bernoulli/bernoulli.exe -include examples/bernoulli/bernoulli.hpp -include examples/bernou
lli/USER_HEADER.hpp stan/lib/stan_math/lib/sundials_3.1.0/lib/libsundials_nvecserial.a stan/lib/stan_math/lib/sundials_3
.1.0/lib/libsundials_cvodes.a stan/lib/stan_math/lib/sundials_3.1.0/lib/libsundials_idas.a

hhschutt@DESKTOP-H7QVAUA ~/cmdstan-2.18.1
```

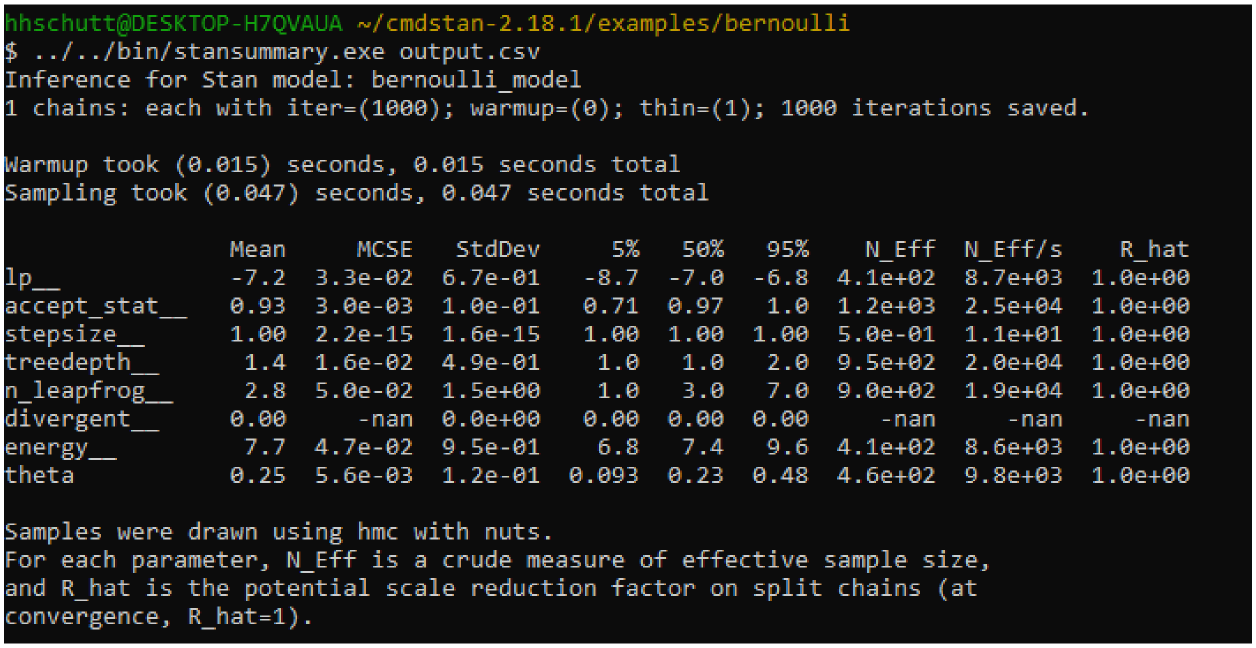
3. Running the example model to see whether everything really works
   a. The program can be executed from the directory in which it resides.
      `> cd examples/bernoulli`
   b. To execute sampling of the model under Linux or Mac, use
      `> ./bernoulli sample data file=bernoulli.data.R`
   c. In Windows, the ./ prefix is not needed, resulting in the following command.
      `> bernoulli.exe sample data file=bernoulli.data.R`



```
hhschutt@DESKTOP-H7QVAUA ~/cmdstan-2.18.1/examples/bernoulli
$ ./bernoulli.exe sample data file=bernoulli.data.R
method = sample (Default)
  sample
    num_samples = 1000 (Default)
    num_warmup = 1000 (Default)
    save_warmup = 0 (Default)
    thin = 1 (Default)
    adapt
      engaged = 1 (Default)
      gamma = 0.050000000000000003 (Default)
      delta = 0.80000000000000004 (Default)
      kappa = 0.75 (Default)
      t0 = 10 (Default)
      init_buffer = 75 (Default)
      term_buffer = 50 (Default)
      window = 25 (Default)
    algorithm = hmc (Default)
      hmc
        engine = nuts (Default)
          nuts
            max_depth = 10 (Default)
        metric = diag_e (Default)
        metric_file =  (Default)
        stepsize = 1 (Default)
        stepsize_jitter = 0 (Default)
id = 0 (Default)
data
  file = bernoulli.data.R
init = 2 (Default)
random
  seed = 3624871800
output
  file = output.csv (Default)
  diagnostic_file =  (Default)
  refresh = 100 (Default)


Gradient evaluation took 0 seconds
1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
Adjust your expectations accordingly!
```

The results are stored in an output.csv file. You can get a summary using the stansummary program in the binaries folder /bin. Like this:



```
hhschutt@DESKTOP-H7QVAUA ~/cmdstan-2.18.1/examples/bernoulli
$ ../../bin/stansummary.exe output.csv
Inference for Stan model: bernoulli_model
1 chains: each with iter=(1000); warmup=(0); thin=(1); 1000 iterations saved.

Warmup took (0.015) seconds, 0.015 seconds total
Sampling took (0.047) seconds, 0.047 seconds total

                Mean     MCSE   StdDev     5%    50%   95%    N_Eff  N_Eff/s    R_hat
lp__            -7.2  3.3e-02  6.7e-01   -8.7   -7.0  -6.8  4.1e+02  8.7e+03  1.0e+00
accept_stat__   0.93  3.0e-03  1.0e-01   0.71   0.97   1.0  1.2e+03  2.5e+04  1.0e+00
stepsize__      1.00  2.2e-15  1.6e-15   1.00   1.00  1.00  5.0e-01  1.1e+01  1.0e+00
treedepth__      1.4  1.6e-02  4.9e-01    1.0    1.0   2.0  9.5e+02  2.0e+04  1.0e+00
n_leapfrog__     2.8  5.0e-02  1.5e+00    1.0    3.0   7.0  9.0e+02  1.9e+04  1.0e+00
divergent__     0.00     -nan  0.0e+00   0.00   0.00  0.00     -nan     -nan     -nan
energy__         7.7  4.7e-02  9.5e-01    6.8    7.4   9.6  4.1e+02  8.6e+03  1.0e+00
theta           0.25  5.6e-03  1.2e-01  0.093   0.23  0.48  4.6e+02  9.8e+03  1.0e+00

Samples were drawn using hmc with nuts.
For each parameter, N_Eff is a crude measure of effective sample size,
and R_hat is the potential scale reduction factor on split chains (at
convergence, R_hat=1).
```