Comprehensive Guide to Install and Configure CmdStanR on Any Computer

This guide provides a step-by-step approach to install and configure CmdStanR for use in R, along with helpful diagnostics to resolve potential issues. It is tailored for Windows systems but can be adapted for other operating systems.

Table of Contents

1. Install Required Tools

Install R and RStudio

1. Download and install the latest version of R from CRAN.
2. Download and install RStudio from RStudio.

Install RTools for Windows

1. Download and install RTools44 from RTools.
2. During installation, ensure that RTools is added to your `PATH`. Verify by checking:

```
where g++
where make
```

2. Install CmdStanR and Dependencies

Install Required R Packages

1. Open R or RStudio and run the following commands:

```
install.packages("remotes")
remotes::install_github("stan-dev/cmdstanr")
library(cmdstanr)
```

Set CmdStan Path

1. Point CmdStanR to the CmdStan directory:

```
cmdstanr::set_cmdstan_path("C:/Users/<username>/anaconda3/envs/<env_name>/Li
brary/bin/cmdstan")
```

Replace `<username>` and `<env_name>` with your user directory and Anaconda environment name.

Verify CmdStan Installation

1. Ensure CmdStan is properly set up by running the following commands:

```
cmdstanr::cmdstan_path()
cmdstanr::cmdstan_version()
```

These commands should return the correct CmdStan path and version. If they do, you are ready to proceed. Here is an example:

```
> cmdstanr::cmdstan_path()
[1] "C:/Users/ansle/anaconda3/envs/stan/Library/bin/cmdstan"

> cmdstanr::cmdstan_version()
[1] "2.36.0"
```

3. Rebuild CmdStan

1. If you encounter issues or need to ensure the installation is fully operational, rebuild CmdStan using the following command:

```
rebuild_cmdstan()
```

2. This step will recompile CmdStan and ensure all necessary files are properly set up. It may take a few minutes to complete.

   Example output:

```
> rebuild_cmdstan()
CmdStan path set to: C:/Users/ansle/anaconda3/envs/stan/Library/bin/cmdstan
...{lots of text output}...
--- CmdStan v2.36.0 built successfully ---
```

4. Compile an Example Model

1. Use the built-in Bernoulli example model to test the setup:

```
file <- file.path(cmdstanr::cmdstan_path(), "examples", "bernoulli",
"bernoulli.stan")
mod <- cmdstan_model(file, force_recompile = TRUE)
```

Example output:

▶ Click to expand full example output

```
> mod <- cmdstan_model(file, force_recompile = TRUE)
 Compiling Stan program...
 In file included from
stan/lib/stan_math/lib/boost_1.84.0/boost/numeric/ublas/traits.hpp:21,
                 from
stan/lib/stan_math/lib/boost_1.84.0/boost/numeric/ublas/storage.hpp:27,
                 from
stan/lib/stan_math/lib/boost_1.84.0/boost/numeric/ublas/vector.hpp:21,
                 from
stan/lib/stan_math/lib/boost_1.84.0/boost/numeric/odeint/util/ublas_wrapper.
hpp:23,
                 from
stan/lib/stan_math/lib/boost_1.84.0/boost/numeric/odeint.hpp:25,
                 from
stan/lib/stan_math/stan/math/prim/functor/ode_rk45.hpp:9,
                 from
stan/lib/stan_math/stan/math/prim/functor/integrate_ode_rk45.hpp:6,
                 from stan/lib/stan_math/stan/math/prim/functor.hpp:16,
                 from stan/lib/stan_math/stan/math/rev/fun.hpp:189,
                 from stan/lib/stan_math/stan/math/rev.hpp:14,
                 from stan/lib/stan_math/stan/math.hpp:19,
                 from stan/src/stan/model/model_header.hpp:6,
                 from C:/Users/ansle/AppData/Local/Temp/RtmpuKlDav/model-
75f078405c17.hpp:2:
 stan/lib/stan_math/lib/boost_1.84.0/boost/numeric/ublas/detail/iterator.hpp:
111:21: warning: 'template<class _Category, class _Tp, class _Distance,
class _Pointer, class _Reference> struct std::iterator' is deprecated [-
Wdeprecated-declarations]
 111 |         public std::iterator<IC, T> {
     |                     ^~~~~~~~

 In file included from c:\rtools44\x86_64-w64-
mingw32.static.posix\lib\gcc\x86_64-w64-
mingw32.static.posix\13.3.0\include\c++\bits\stl_iterator_base_funcs.h:66,
                 from c:\rtools44\x86_64-w64-
mingw32.static.posix\lib\gcc\x86_64-w64-
mingw32.static.posix\13.3.0\include\c++\string:47,
                 from c:\rtools44\x86_64-w64-
```

```
mingw32.static.posix\lib\gcc\x86_64-w64-
mingw32.static.posix\13.3.0\include\c++\bits\locale_classes.h:40,
                 from c:\rtools44\x86_64-w64-
mingw32.static.posix\lib\gcc\x86_64-w64-
mingw32.static.posix\13.3.0\include\c++\bits\ios_base.h:41,
                 from c:\rtools44\x86_64-w64-
mingw32.static.posix\lib\gcc\x86_64-w64-
mingw32.static.posix\13.3.0\include\c++\ios:44,
                 from c:\rtools44\x86_64-w64-
mingw32.static.posix\lib\gcc\x86_64-w64-
mingw32.static.posix\13.3.0\include\c++\istream:40,
                 from c:\rtools44\x86_64-w64-
mingw32.static.posix\lib\gcc\x86_64-w64-
mingw32.static.posix\13.3.0\include\c++\sstream:40,
                 from stan/src/stan/io/var_context.hpp:4,
                 from stan/src/stan/model/model_base.hpp:7,
                 from stan/src/stan/model/model_header.hpp:4:
 c:\rtools44\x86_64-w64-mingw32.static.posix\lib\gcc\x86_64-w64-
mingw32.static.posix\13.3.0\include\c++\bits\stl_iterator_base_types.h:127:3
4: note: declared here
 127 |     struct _GLIBCXX17_DEPRECATED iterator
     |                                  ^~~~~~~~

 stan/lib/stan_math/lib/boost_1.84.0/boost/numeric/ublas/detail/iterator.hpp:
149:21: warning: 'template<class _Category, class _Tp, class _Distance,
class _Pointer, class _Reference> struct std::iterator' is deprecated [-
Wdeprecated-declarations]
 149 |         public std::iterator<IC, T> {
     |                     ^~~~~~~~
 c:\rtools44\x86_64-w64-mingw32.static.posix\lib\gcc\x86_64-w64-
mingw32.static.posix\13.3.0\include\c++\bits\stl_iterator_base_types.h:127:3
4: note: declared here
 127 |     struct _GLIBCXX17_DEPRECATED iterator
     |                                  ^~~~~~~~
 stan/lib/stan_math/lib/boost_1.84.0/boost/numeric/ublas/detail/iterator.hpp:
204:21: warning: 'template<class _Category, class _Tp, class _Distance,
class _Pointer, class _Reference> struct std::iterator' is deprecated [-
Wdeprecated-declarations]
 204 |         public std::iterator<IC, T> {
     |                     ^~~~~~~~
 c:\rtools44\x86_64-w64-mingw32.static.posix\lib\gcc\x86_64-w64-
mingw32.static.posix\13.3.0\include\c++\bits\stl_iterator_base_types.h:127:3
4: note: declared here
 127 |     struct _GLIBCXX17_DEPRECATED iterator
     |                                  ^~~~~~~~
```

5. Test Sampling with the Example Model

1. Define the data for the Bernoulli model:

```
data_list <- list(N = 10, y = c(0, 1, 0, 0, 0, 0, 0, 0, 0, 1))
```

2. Run the sampling process:

```
fit <- mod$sample(
 data = data_list,
 seed = 123,
 chains = 4,
 parallel_chains = 4,
 refresh = 500 # print update every 500 iterations
 )
```

3. Check model fit summary

```
fit$summary()
```

Example output:

▶ Click to expand full example output

```
> # names correspond to the data block in the Stan program
> data_list <- list(N = 10, y = c(0,1,0,0,0,0,0,0,0,1))
> fit <- mod$sample(
+   data = data_list,
+   seed = 123,
+   chains = 4,
+   parallel_chains = 4,
+   refresh = 500 # print update every 500 iters
+ )
Running MCMC with 4 parallel chains...

Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1 finished in 0.0 seconds.
Chain 2 finished in 0.0 seconds.
Chain 3 finished in 0.0 seconds.
Chain 4 finished in 0.0 seconds.

All 4 chains finished successfully.
Mean chain execution time: 0.0 seconds.
Total execution time: 0.3 seconds.

> fit$summary()
# A tibble: 2 × 10
variable   mean median    sd   mad      q5    q95  rhat ess_bulk ess_tail
<chr>     <dbl>  <dbl> <dbl> <dbl>   <dbl>  <dbl> <dbl>    <dbl>    <dbl>
1 lp__     -7.27  -6.99  0.750 0.323 -8.74   -6.75   1.00    1924.    2287.
2 theta     0.252  0.239 0.120 0.122  0.0806  0.477  1.00    1368.    1664.
```

6. Diagnostics and Troubleshooting

1. **Check CmdStan Toolchain**

   If any issues arise during compilation or sampling, ensure the required toolchain is properly set up:

   ```
   > cmdstanr::check_cmdstan_toolchain()
   The C++ toolchain required for CmdStan is setup properly!
   ```

2. **Rebuild CmdStan**

   If issues persist, try rebuilding CmdStan to ensure a clean and complete setup:

   ```
   rebuild_cmdstan()
   ```

3. **Inspect Output for Errors**

   If a sampling process fails, you can inspect the output logs:

   ```
   cat(fit$output(1)) # Replace '1' with the chain number you want to check
   ```

4. **Verify system PATH variable**

   Ensure g++ and make are correctly linked and available in your system's PATH:

```
system("where g++")
system("where make")
```

Example Output:

```
> system("where g++")
 C:\\rtools44\\x86_64-w64-mingw32.static.posix\\bin\\g++.exe

 > system("where make")
 C:\\rtools44\\usr\\bin\\make.exe
```

5. **Helpful URLs**

    - Getting Started with CmdStanR
    - Install cmdstan for the first time
    - Stan Forums

Back to top