Install CmdStan for use in Julia

The following instructions are not set in stone and have not been updated since the 13/11/2020. Other ways to install CmdStan and CmsStan.jl might be found, however this way is the one that worked the best for me and was the easiest (even if it does not seem so) to be able to make all the Stan utilities in Julia. This requires having Conda (Python) installed.

Note!

Make sure that previously you install the latest frozen version of Rtools (https://cran.r-project.org/bin/windows/Rtools/history.html) and that this (and the mingw32 and mingw64 directories and the binaries) are located in the system path, otherwise cmdstan will not be able to be installed. During the Rtools installation there is an option for Rtools to be added in the environment path. Make sure that this items appear at the top of the path list in Windows (have not tested in other OS) and that mingw64\bin appears before mingw32\bin.

Installation of CmdStan

Install CmdStan through Conda and Python as described in the link:
https://cmdstanpy.readthedocs.io/en/stable-0.9.65/getting_started.html

- **Step 1:** in the anaconda prompt type the command “pip install --upgrade cmdstanpy” to install cmdstanpy (this will not be used, but we will install the actual cmdstan through it).
- **Step 2:** Go to python in the anaconda promp
- **Step 3:** In python, import the package by typing “import cmdstanpy”
- **Step 4:** Install cmdstan by typing the python command “cmdstanpy.install_cmdstan()”. Python will prompt the installation path for CmdStan.
- **Note:** Last time I tried the latest version of cmdstan had installation issues in my Windows machine. If that is the case I would recommend installing a previous version (2.20.0 works fine for me). To do this, in the installation command add -v 2.20.0 (installation from terminal).

Install CmdStan.jl

In Julia type the command “Pkg.add(“CmdStan”)”.

Allow ODEs in CmdStan

If not present apply the modification in CmdStan (not the Julia one) described in https://github.com/stan-dev/cmdstan/commit/cee3902bfb4689b8ff4c6e0fd9701d8c6d9b7427

This might be fixed after cmdstan 2.20.0

- **Step 1:** Locate the file make\program. In my case this is located in “C:\Users\david\.cmdstanpy\cmdstan-2.20.0\make\program”.
- **Step 2:** Locate the line: $(LINK.cpp) $(LDLIBS) $(CMDSTAN_MAIN_O) $(LIBSUNDIALS) $(MPI_TARGETS) $*.o $(OUTPUT_OPTION)
- **Step 3:** Change it to: $(LINK.cpp) $*.o $(CMDSTAN_MAIN_O) $(LDLIBS) $(LIBSUNDIALS) $(MPI_TARGETS) $(OUTPUT_OPTION)

Allow complex data structures in Stan:

By the time I installed CmdStan.jl, the introduction of matrices and complex vectors was not supported in the conversion of the Data introduced in Julia to the R files containing the data. For this package to work, you need to allow this, introducing a small modification in the source code of CmdStan.jl

- **Step 1:** Locate the file create_r_files.jl in the installed CmdStan.jl. In my case, the location of this file is C:\Users\david\.juliapro\JuliaPro_v1.4.0-1\packages\CmdStan\8pZC2\src\utilities\create_r_files.jl
- **Step 2:** By default, the script is in Read-Only attribute. Change this to allow modifications in the script.
- **Step 3:** Inside the file, locate the function "update_R_file" (I have uploaded an example file on how this should look like, called create_r_files_example.jl) and inside the for loop that starts with "for entry in dct" introduce the following 2 elseif statements (if this are not there):

```
elseif length(val)==1 && (length(size(val))==1) # Single element vector
        str = str*"$(val[1])\n"
elseif length(val)==1 && (length(size(val))==2)
        str = str*"structure(c("
        write(strmout, str)
        str = ""
        writedlm(strmout, val[:]', ',')
        dimstr = "c"*string(size(val))
        str = str*"), .Dim=$(dimstr))\n"
```

Use Stan in Julia

- **Step 1:** In Julia, import the CmdStan package by typing "using CmdStan"
- **Step 2:** Add the path to the local CmdStan installation folder using the function set_cmdstan_home!. In my case, this is done by typing: "set_cmdstan_home!("C:/Users/david/.cmdstanpy/cmdstan-2.20.0")" in Julia after calling the package.
- **Step 3:** Check that Stan is properly working. Here is a basic example to test:

```
using CmdStan
set_cmdstan_home!("C:/Users/david/.cmdstanpy/cmdstan-2.20.0")

bernoullistanmodel = "
data {
int<lower=0> N;
int<lower=0,upper=1> y[N];
}
parameters {
real<lower=0,upper=1> theta;
}
model {
theta ~ beta(1,1);
y ~ bernoulli(theta);
}
";

stanmodel = Stanmodel(name="bernoulli", model=bernoullistanmodel);

bernoullidata = Dict("N" => 10, "y" => [0, 1, 0, 1, 0, 0, 0, 0, 0, 1])

rc, chns, cnames = stan(stanmodel, bernoullidata)
```

The End