Getting started with BELMM

Olli Sarala 27.7.2023.

This is a tutorial for applying the BELMM framework implemented for Windows 11 (and 10*) 64-bit machines without dual-booting**. In this tutorial, we will provide a step-by-step rundown to the installation of WSL and the dependencies of CU-MSDSp. Following the steps below will take around 15-20 minutes depending on the device.

The folder for this example is found at https://github.com/ollisa/BELMM/tree/main/example .

For the installation, you will need the administrators' rights and a connection to the internet.

*) The installation should be identical for Windows 11 and Windows 10.

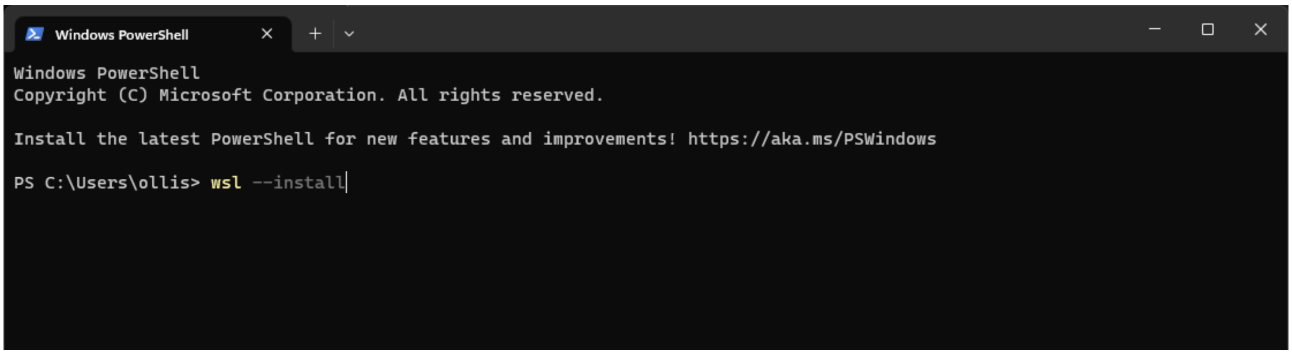
**) If you are using a native Linux machine or Mac OS, you can skip step 1). For other steps on Mac OS, please adapt them accordingly since I am not familiar with it.

Getting WSL

1. Installation of Windows subsystem for Linux (WSL)

a. Open Windows PowerShell using administrators' rights and copy the following to the command line. Then press enter.

```
wsl --install
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ollis> wsl --install
```

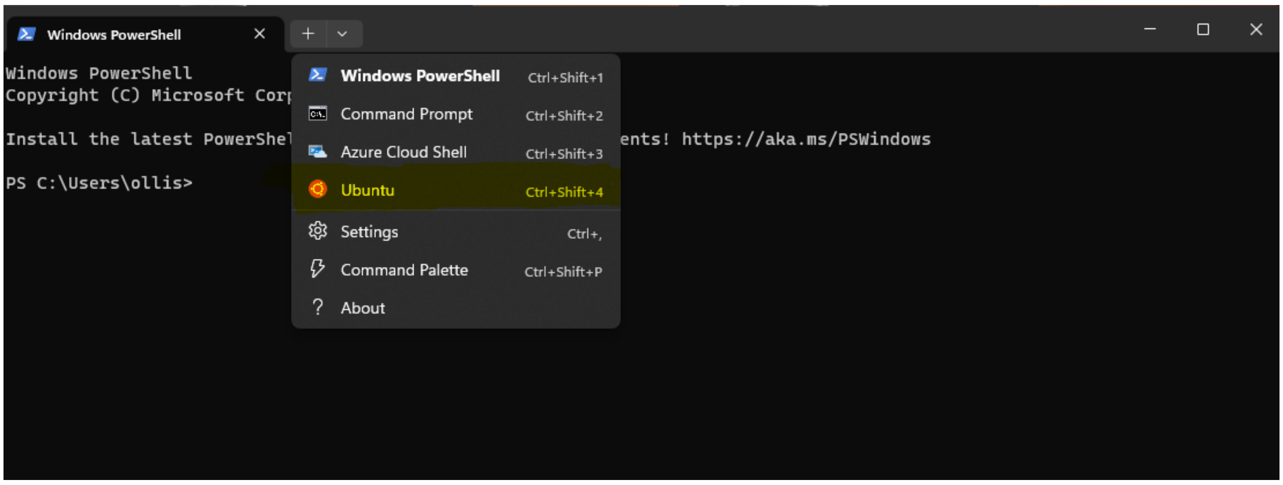
This will take a couple of minutes. After the installation is done, restart the computer.

b. After the restart, PowerShell should open automatically (otherwise open manually). Write the username you wish to use, and press enter. Next, type in the password you wish to use, then press enter and confirm the password by repeating. The password you type will be invisible on the command line. You will need the password for confirming the 'sudo' commands (installation and upgrading of packages).

2. Creating a working folder shared between the WSL and Windows. This is for convenience***. To do this, just create a folder on the Desktop and name it 'wsl'.

***) We note that having a shared folder between the Windows and WSL is not the best practice. In some cases, accessing the files from multiple systems at once could lead to corruption of the WSL. However, this should not be an issue with our implementation.

3. Accessing Ubuntu (at the time of writing, the default installed version is 22.04 LTS)
   a. Open a new tab in PowerShell by selecting Ubuntu.



```
Windows PowerShell
Copyright (C) Microsoft Corp

Install the latest PowerShe                                  ents! https://aka.ms/PSWindows

PS C:\Users\ollis>
```

Windows PowerShell
Windows PowerShell Ctrl+Shift+1
Command Prompt Ctrl+Shift+2
Azure Cloud Shell Ctrl+Shift+3
Ubuntu Ctrl+Shift+4
Settings Ctrl+,
Command Palette Ctrl+Shift+P
About

Optionally, you can update the Ubuntu on the first opening by executing the following two commands,

```
sudo apt-get update
```
and

```
sudo apt-get upgrade -y
```

   b. Execute the following command (should be necessary only on WSL)

```
sudo apt-get install bc
```
In a nutshell, 'bc' is a command line calculator used for arithmetic operations. For some reason, it did not come with WSL installation. If it's missing, there will be a compiler error message "bc: command not found".

4. Set the folder 'wsl' as your working directory by executing

```
cd /mnt/c/Users/<your_name>/Desktop/wsl/
```

Above, replace `<your_name>` with your Windows username (the one that shows under C:\Users in Windows (copy to Windows search box)). Make sure you are you are using this directory in the following steps, or the paths shown in this tutorial are wrong and the RJMCMC **will not work**.

Extra information about the installation of WSL can be found at
https://wiki.usask.ca/display/MESH/Installing+Python+and+the+Windows+Subsystem+for+Linux

Getting the Dependencies

5. Installation of dependencies
   Please note below that WSL calls for ’python3’ instead of ’python’ that is more commonly used in documentations through the internet.

   a. scipy (numpy), matplotlib, seaborn (default version of each)

```
python3 -m pip install scipy matplotlib seaborn
```

The versions we used are Python 3.10.6, scipy 1.11.1, matplotlib 3.7.2, seaborn 0.12.2.

You can check the versions by executing

```
python3 –version

python3 -c "import scipy; print(scipy.__version__)"

python3 -c “import matplotlib; print(matplotlib. __version__)”

python3 -c “import seaborn; print(seaborn. __version__)”
```

Note: CU-MSDSp uses these for plotting.

   b. Open MPI (versio 4.1.2)

```
sudo apt install openmpi-bin libopenmpi-dev
```

Note: CU-MSDSp uses this package to handle the parallelization.

   c. GNU scientific library (version 2.35)

```
sudo apt-get install libgsl-dev
```

Optional troubleshooting : You can check that the installation was successful by following the example provided at https://walkingrandomly.com/?p=6507 . Just copy the code to Windows Notepad and save it as ’dawson.cpp’ . It can be compiled by executing

```
g++ -std=c++11 dawson.cpp -o ./dawson -lgsl -lgslcblas -lm
```

followed by

```
./dawson > results.txt
```

You can then open the ’results.txt’ file from your ’wsl’ folder. If numbers are present, everything should be fine.

   d. CmdStan (version 2.22)

To install CmdStan, execute

```
git clone https://github.com/stan-dev/cmdstan.git --recursive
```

This can take a second. Next, execute either of the following commands:

If you are not in a hurry, do

```
cd cmdstan
```

Otherwise, it’s recommended to take advantage of multiple physical cores by copying

```
make -j4 build
```

The number after ‘-j‘ denotes the number of cores that will be used in the compilation process.

Next, check that the installation was successful by executing

1.

```
make examples/bernoulli/bernoulli
./examples/bernoulli/bernoulli sample data
file=examples/bernoulli/bernoulli.data.json
```

2.

```
ls -l output.csv
```

3.

```
bin/stansummary output.csv
```

If everything went correctly, the output: mean(theta) ± sd(theta), should include 0.2.

The full installation guide and troubleshooting tips for cmdstan can be found at https://mc-stan.org/docs/2_24/cmdstan-guide/cmdstan-installation.html#git-clone.section .

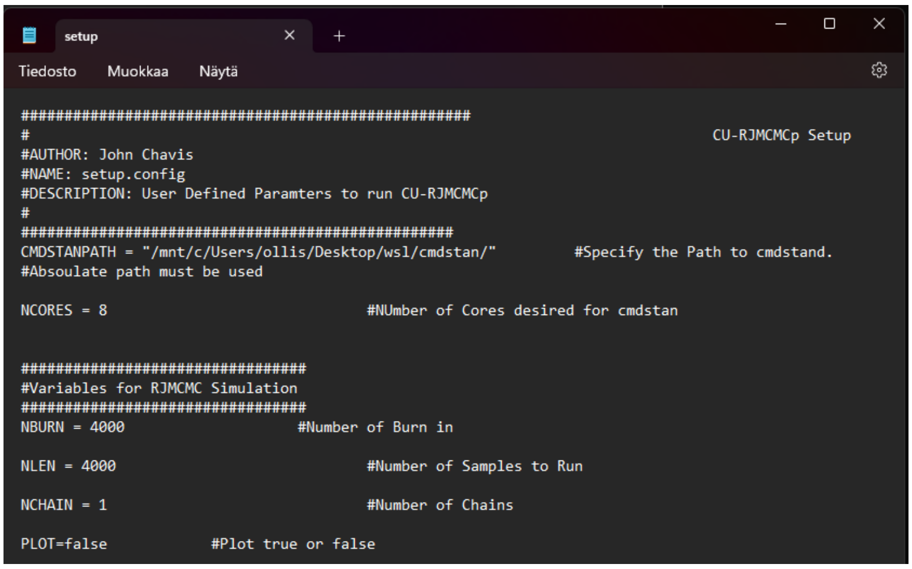
Note: CmdStan is used for MCMC to estimate the models.

6. CU-MSDSp

Below, we give two ways to getting the Reversible jump implementation CU-MSDSp. The first one A) is provided by us and should not require any modification to the **main scripts** by the user to run on WSL (based on the analyses we run). The second option B) is to download the files directly from the source at https://github.com/jtchavisIII/CU-MSDSp and modifying them according to the instructions.

Note: You can use the B) as a reference as what to look at/modify if you run to issues on Ubuntu or on Mac.

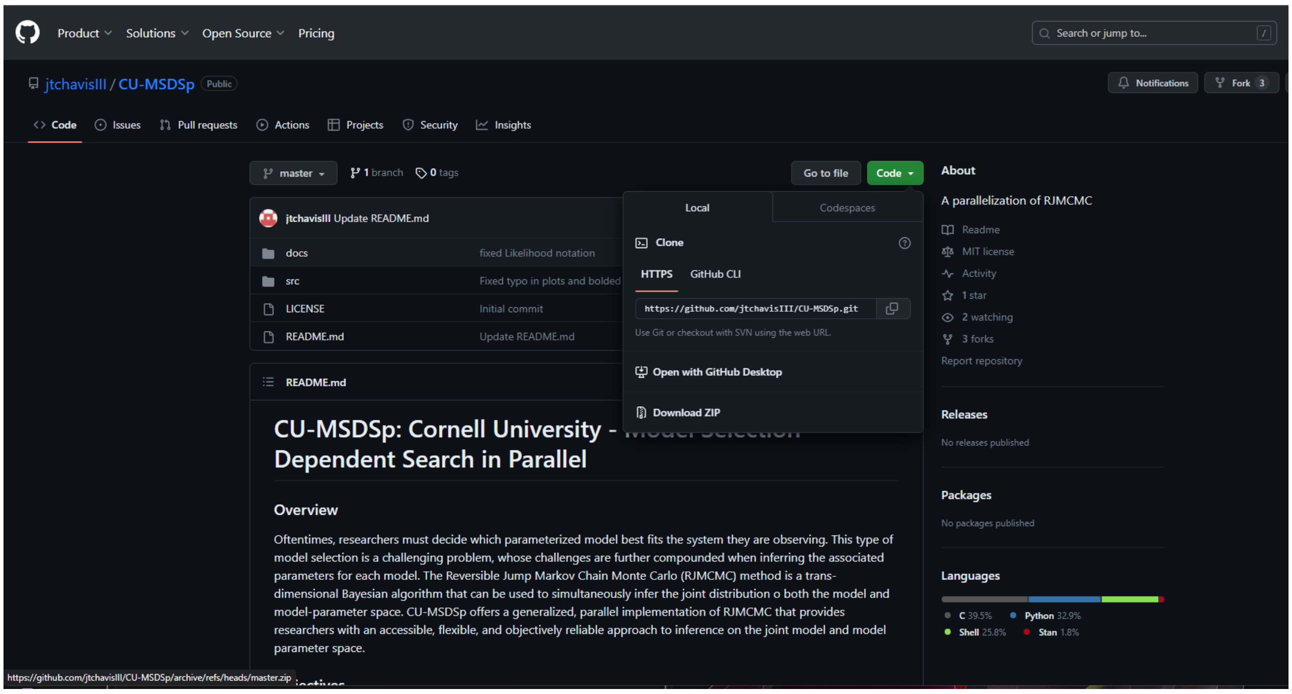
A. From https://github.com/ollisa/BELMM/tree/main/example
   1. Download the example folder
   2. Copy the ‘MSDSP’ folder into your ‘wsl’ folder
   3. In the ‘MSDSP’ folder, open ‘src’ and open ‘setup.config’ using NotePad:



```
####################################################
#                                                                               CU-RJMCMCp Setup
#AUTHOR: John Chavis
#NAME: setup.config
#DESCRIPTION: User Defined Paramters to run CU-RJMCMCp
#
##################################################
CMDSTANPATH = "/mnt/c/Users/ollis/Desktop/wsl/cmdstan/"         #Specify the Path to cmdstand.
#Absoulate path must be used

NCORES = 8                                #NUmber of Cores desired for cmdstan


#################################
#Variables for RJMCMC Simulation
#################################
NBURN = 4000                      #Number of Burn in

NLEN = 4000                               #Number of Samples to Run

NCHAIN = 1                                #Number of Chains

PLOT=false            #Plot true or false
```

a. In the row CMDSTANPATH:
CMDSTANPATH = "/mnt/c/Users/<your_name>/Desktop/wsl/cmdstan/"
replace <your_name> with your Windows username (the one that shows under C:\Users in Windows (copy to Windows search box)).
b. NCORE: set the number of cores you wish to use. At least 1 core/model is recommended.
c. NBURN, NLEN: number of burn-in and number of samples
d. NCHAIN: number of chains.
Note: Only 1 is recommended for mixture models due to possible within-chain label switching.
e. save the changes and close.

B. From the source and the modifications:



jtchavisIII Update README.md

docs	fixed Likelihood notation
src	Fixed typo in plots and bolded
LICENSE	Initial commit
README.md	Update README.md

https://github.com/jtchavisIII/CU-MSDSp.git

Use Git or checkout with SVN using the web URL.

A parallelization of RJMCMC

No releases published

No packages published

C 39.5% Python 32.9% Shell 25.8% Stan 1.8%

README.md

CU-MSDSp: Cornell University - Model Selection Dependent Search in Parallel

Overview

Oftentimes, researchers must decide which parameterized model best fits the system they are observing. This type of model selection is a challenging problem, whose challenges are further compounded when inferring the associated parameters for each model. The Reversible Jump Markov Chain Monte Carlo (RJMCMC) method is a trans-dimensional Bayesian algorithm that can be used to simultaneously infer the joint distribution o both the model and model-parameter space. CU-MSDSp offers a generalized, parallel implementation of RJMCMC that provides researchers with an accessible, flexible, and objectively reliable approach to inference on the joint model and model parameter space.

https://github.com/jtchavisIII/CU-MSDSp/archive/refs/heads/master.zip

a. Download CU-MSDSp at https://github.com/jtchavisIII/CU-MSDSp by choosing "Download zip" under the green 'Code' button.
b. Open the zip file, copy the 'src' folder. Create a folder "MSDSp" into your 'wsl' folder, into which you paste the 'src'.
Note: This folder contains the CU-MSDSp files.
c. Next, do the following modifications:
   1. Open 'runParallel.sh' using the Notepad and in the rows 270 and 577 change "python → python3"
   Note: Your python command may vary depending on your system.
   2. In the row 331, add "#include <string.h>"
   3. Replace the row 523 with "mpicc -o runRJMCMC runRJMCMC.c -lm -ldl -lgsl -lgslcblas && mpirun -np "$NPROC" --oversubscribe ./runRJMCMC"
   Note: This fixes a compilation error.
   4. Save and close the file.
   5. Open 'bodypRJMCMC.c' using Notepad.
   6. On the line 328, replace "modelAcceptProb = 0.1"

Note: This is advisable if you plan to work with models of >500 parameters

7. save and close the file

d. Open 'setup.config' using Notepad. Replace
   1. CMDSTANPATH = "/mnt/c/Users/`<your_name>`/Desktop/wsl/cmdstan/"
      Your Windows user name.
   2. Set the following
      a. NCORE: set the number of cores you wish to use. At least 1 core/model is recommended.
      b. NBURN, NLEN: number of burn-in and number of samples
      c. NCHAIN: number of chains.
         Note: Only 1 is recommended for mixture models due to possible within-chain label switching.
      d. save the changes and close.

Estimating the models and running the RJMCMC

Note 1: The data is stored in the 'data' folder and needs to be **unboxed** JSON format. Please check the R notebook for how to write these given your data. As the models are written currently, the JSON file must contain the following*:

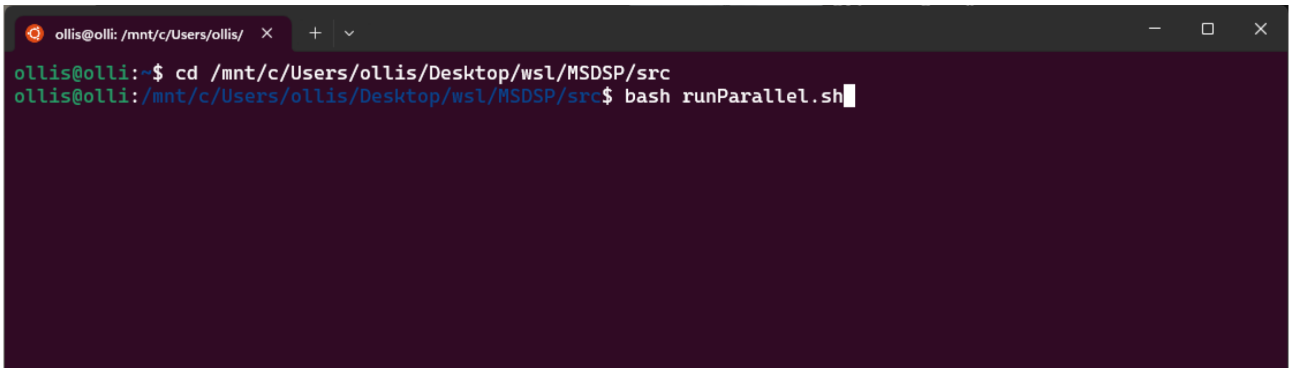
K = 'number of observations', T = 'number of time points', z = 'data matrix' (K x T), sigma_y = 'smoothing prior' (either a vector or single value), a = 'lower bound for the initial values', b = 'upper bound for the initial values'

*) The simulated data examples may differ from this format since they were done first.

Note 2: Check the many models in the 'models' folder for examples on how to structure your Stan models. These are stored in the 'models' folder in 'src'. For the RJMCMC to work, these **must** contain the parameter 'logModel' in the generated quantities block.

1. To estimate the models and run the CU-MSDSp, execute

```
cd /mnt/c/Users/<your_name>/Desktop/wsl/MSDSP/src

bash runParallel.sh
```



```
ollis@olli:~$ cd /mnt/c/Users/ollis/Desktop/wsl/MSDSP/src
ollis@olli:/mnt/c/Users/ollis/Desktop/wsl/MSDSP/src$ bash runParallel.sh
```

Note: In the following runs, you can press 'up arrow' key in the Ubuntu command line to skip writing the commands again.

2. After the estimation is completed, the results are found under ’src’ in folders ‘goldStandardChains’, ’modelSelection’, and ’pics’ (if plotting is selected). The runtime in (seconds x 1000) can be found in ‘timings.txt’.

An example

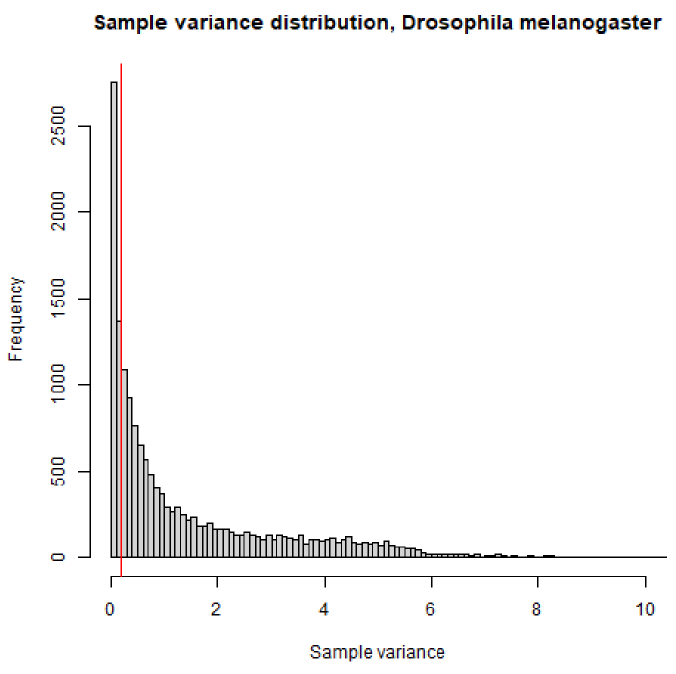
Simulation study, data set 2.

Post-processing and plotting the models

We did the post-processing using RStudio. Copy the ‘src’ folder into your working directory and name it as you like. Then follow the steps shown in one of the Rmd files or open the notebook found in the ‘example’ folder.

Choosing the smoothing prior

Like can be seen from the Figure 4 of “BELMM: Bayesian model selection and random walk smoothing in time-series clustering”, the exact value of the smoothing prior is not that important. We recommend choosing a constant value smaller than the median of the sample variances to be used for all the mixture components. Below, we show the sample variance distribution for the Drosophila melanogaster time series data set. The median value is marked with the red line.


Sample variance distribution, Drosophila melanogaster
Frequency
Sample variance