R-Based High Performance Computing for Social Science : Using the HMDC Cluster

Instructor: Soubhik Barari
Faculty Advisor: In Song Kim
MIT Political Science Methods Workshop

Friday 20th October, 2017

Contents

1 Introduction

This workshop will introduce the concept of high performance computing (HPC) using the **R** programming language. We will walk through specific computational exercises and discover how to improve them with high performance computing. In particular, we will be using the **Harvard-MIT Data Center (HMDC) cluster**, a computing cluster maintained and used for computational social science research. We will interface with HMDC using a tool called RCE (Research Computing Environment), a fully encompassing environment providing editors, browsers, and OS access to work with the cluster.

The purpose of this document is to provide directions on HMDC account set-up (to be used following the workshop) as well as general resources for HPC tasks. Some pre-requistes in order to effectively perform high performance R computing are listed below:

1. Prerequisite: Proficiency in R.
2. Prerequisite: A working version of **RStudio**.
3. Prerequisite: Cygwin or PuTTY installed (for Windows users). [1]
4. Preferred: Basic knowledge of the UNIX/DOS command line. [2]

To best follow along with the coding demos, *it is recommended that you set up an HMDC account ahead of the workshop.*

All code, slides, and other materials for this worshkop will be made public at the following GitHub code repository: `https://github.com/soubhikbarari/MITMethodsOct2017_hpc`.

2 Introduction to HMDC

2.1 About the HMDC cluster

The HMDC RCE website can be rather difficult to navigate, and much of the information you need to use the resources is not on the website. They are extremely responsive to email, however, so if you're ever having issues, shoot them an email.

2.2 Account Set-up

In order to apply for an RCE account, you have to send an email to support@help.hmdc.harvard.edu. In the email, include answers to the following questions. Below the questions are example responses (the text of questions is from the automatic email you receive if

[1] See Appendix A for installation and set-up directions.

[2] Resources for basic command line usage are provided in Appendix A; this will also briefly be covered during the workshop.

you email them to ask about setting up an account). Feel free to copy, paste, and modify in your own email request:

1. **Please provide contact information for your account sponsor. This would preferably be an IQSS, CGIS or MIT faculty member. If you are seeking access to the HMDC Research Computing Environment independently, please provide us with contact information for the director of your research (e.g. your thesis advisor). If you are a Harvard affiliate, please provide your school (HKS, HBS, FAS, etc) as well as your specific department or group.**

   Our sponsor is In Song Kim, who is an MIT assistant professor. He is cc'd in this email.

2. **Please provide a very short description of your project or the nature of your work.**

   Our aim is to test whether donating to a candidate and winning makes an individual more likely to donate in the future. The idea that successful past donation encourages future donation is a common hypothesis in psychology, but has not been tested with regard to political participation. We plan on testing this using a regression discontinuity design, which is a statistical method used to make causal inferences about observational data. This is method frequently used in econometrics.

3. **Which statistical programs (R, Stata, SAS, etc.) are you interested in using?**

   We will be using R.

4. **How much total disk space do you anticipate needing?**

   We anticipate using 50 G, if not more than that. The entirety of the donations data we will be working with is 41.6 G. We will be merging several of those datasets together and creating many additional variables, which will take up additional space.

5. **Do you require backups of your data?**

   No.

6. **Will you be working with any confidential data (for example, identifiable human data)?**

   If so, we require documentation from the appropriate Harvard IRB classifying the data as Level 3 or lower.

   (a) Please also provide contact information for the Principal Investigator on your confidential research.

   (b) If someone leaves your project, or your project ends, please let us know as soon as possible so we can update the access rules.

(c) You will be asked to review the list of persons with access to your confidential data annually. If you or another responsible party cannot be reached, your data will be deleted to protect their confidentiality.

Harvard Confidential Information includes any of the following:

(a) A person's name + state, federal or financial identifiers

(b) Business information specifically designated by the School as confidential

(c) Identifiable business information that puts individuals at risk if disclosed

(d) Research data containing private information about identifiable individuals

(e) Student records (such as collections of grades, correspondence)

For guidelines on handling Harvard Confidential Information, please refer to http://projects.iq.harvard.edu/user-services/confidential-information-policy-summary.

No. The data we are working with is a publicly available dataset drawn from public government records of political donations.

7. **Approximately how many months do you expect your project to last?**

   12 months.

8. **If you plan to use cluster resources: How long do you expect your jobs to take?**

   Approximately one week.

9. **How many concurrent jobs do you expect to run?**

   One.

10. **Do you have any other concerns about the handling of your data / jobs?**

    No.

Following such an email request, you should receive a response with credentials for your own RCE account.

3 Working with HMDC

3.1 Accessing the RCE

Now, to install the RCE desktop client on your computer, follow the instructions at `http://projects.iq.harvard.edu/rce/nx4_installation`.

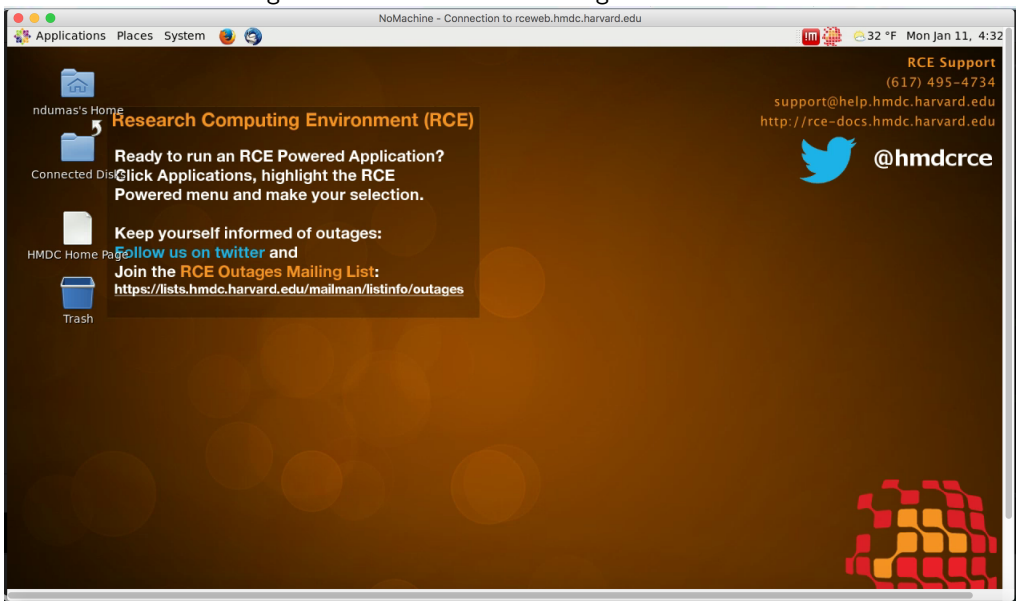
Once the RCE client has been installed, double-click to open. You will need to configure the desktop client.

Additionally, the website at `http://projects.iq.harvard.edu/rce/nx4_webclient` has detailed, step-by-step instructions walking through everything you need to do to configure the client.

3.2 Opening and running an application

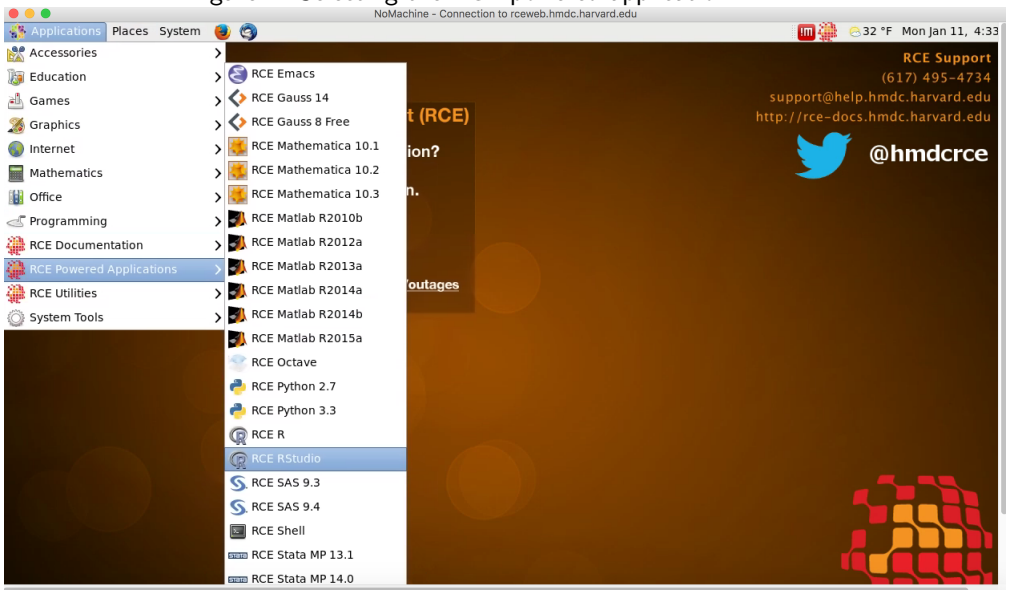
When you open a session, the client will look like the screenshot in Figure 1.

Figure 1: The RCE Client Background.



To open an RCE powered application, like RStudio, choose the "RCE Powered Applications" menu and scroll down to the application you want. Figure 2 shows the menus.

Figure 2: Selecting the RCE powered application.



Once you open the application, you will be asked how much RAM and how many cores you need (Figures 3 and 4).

Figure 3: Select the amount of memory.

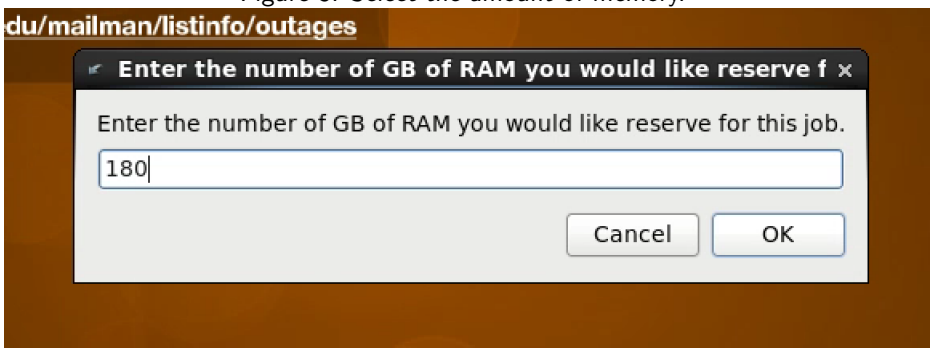
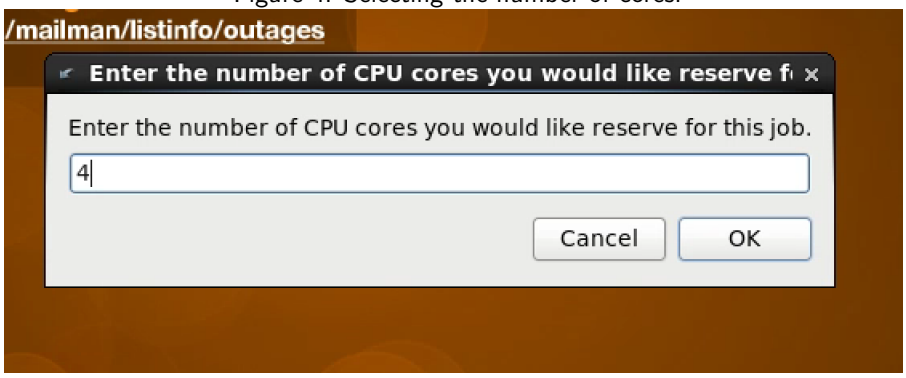
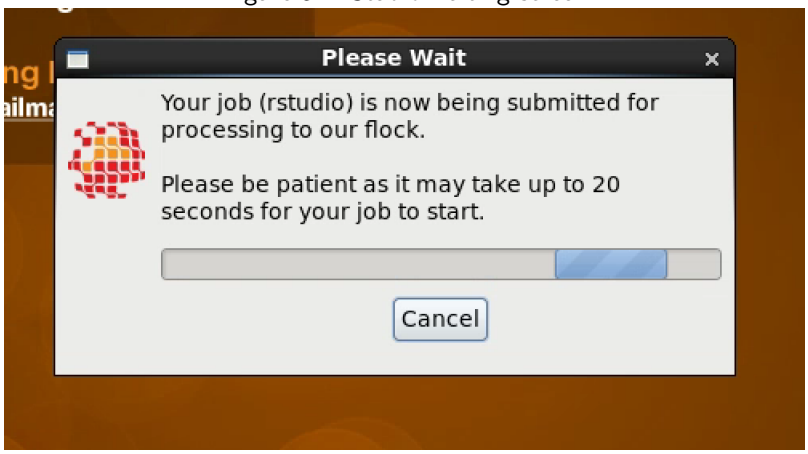


Figure 4: Selecting the number of cores.



Once that's happened, you will see the wait screen shown in Figure 5.

Figure 5: RStudio waiting screen.



Then the application will open, and you use it like any other application.

3.3 Correct keyboard mapping

You may notice that upon typing in RStudio, your output is garbled. This indicates that your virtual machine does not have the correct mapping from your keyboard to the virtual keyboard. This can be easily fixed. *If you do not notice this problem, you may skip this section.*

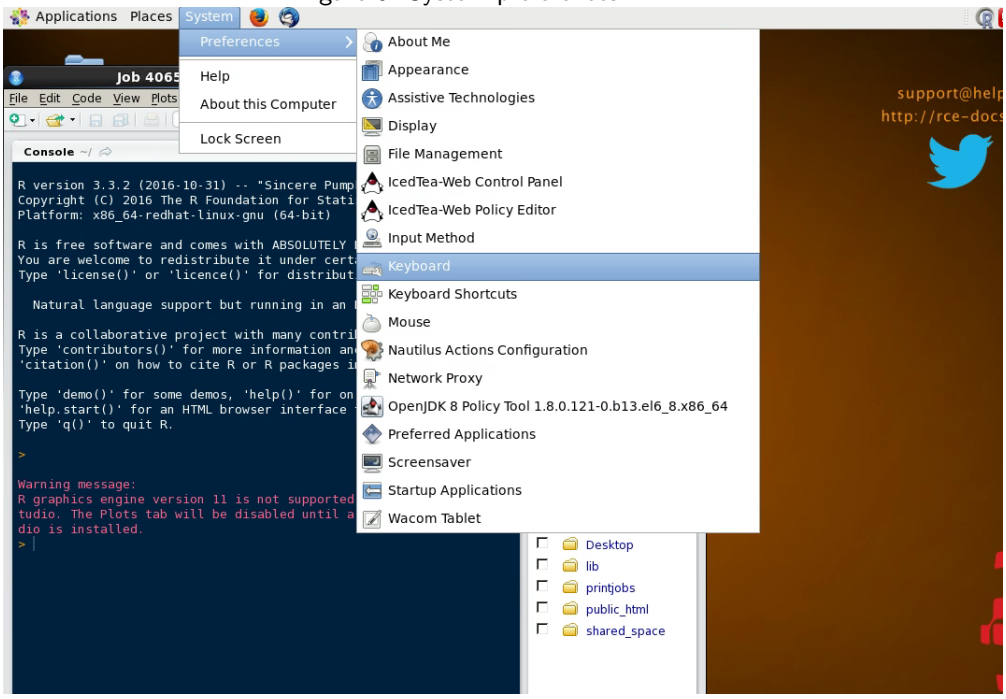
If you are using a standard U.S. 'qwerty' keyboard:

Simply select `Applications > RCE Utilities > Load US keyboard map.`

If you are not using a standard U.S. 'qwerty' keyboard:

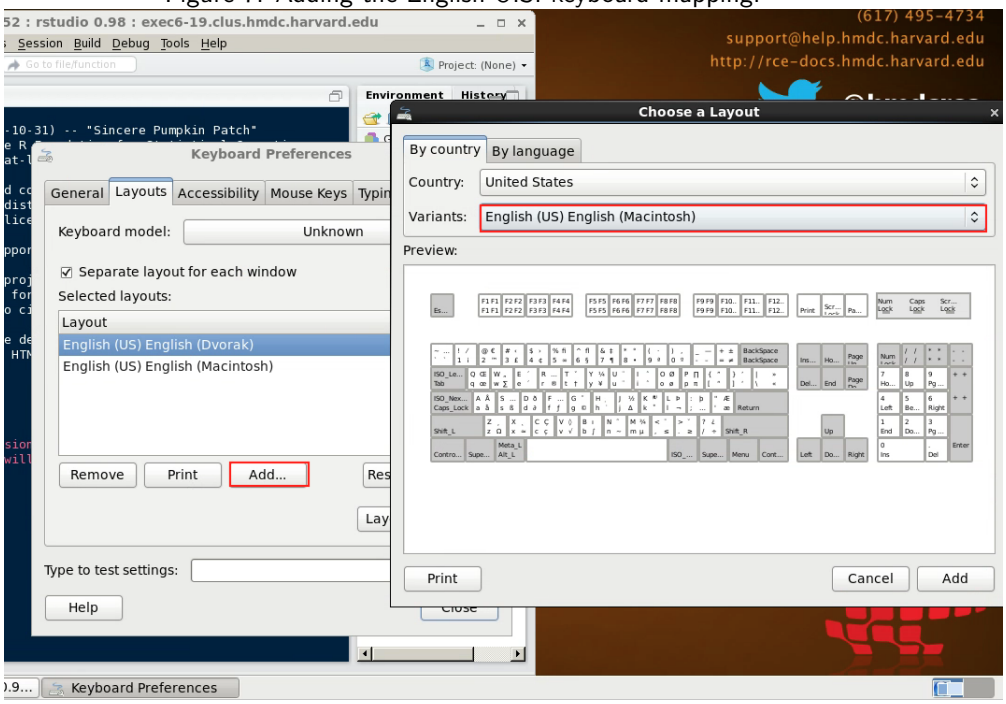
Navigate to `System` > `Preferences` > `Keyboard` as shown in Figure 6.

Figure 6: System preferences.



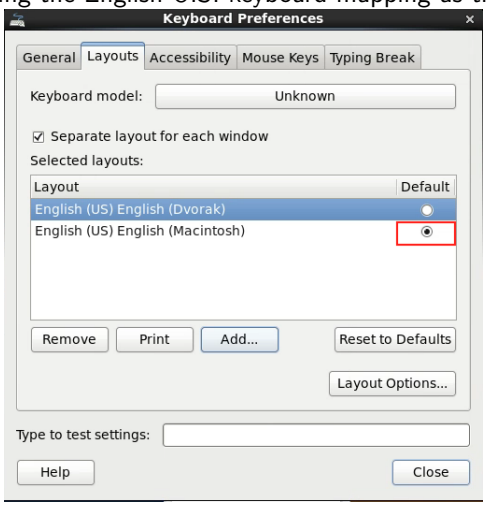
This will open the `Keyboard Preferences` window. Then, in the `Layouts` tab of the window, click on the `Add...` button. You may now select the appropriate keyboard mapping for your machine. This is shown below in Figure 7.

Figure 7: Adding the English U.S. keyboard mapping.



Once you add this layout, you must select it as the default keyboard layout for your account, as in Figure 8. This may require you removing the default layout which may be `English (Dvorak)`:

Figure 8: Selecting the English U.S. keyboard mapping as the default layout.



You should now be able to produce sensible text output with your keyboard input.

If you are still encountering problems, you may wish to click on `Reset to default` in the preferences window and re-add the appropriate layout.

For further troubleshooting, refer to the directions at `http://rce-docs.hmdc.harvard.edu/faq/my-text-garbled-when-i-type-rce-desktop` or send an inquiry to support@help.hmdc.harvard.edu.

3.4 Directory structure

An application like RStudio on the RCE remote desktop client works just like RStudio on a regular desktop or laptop. One thing to make sure of is that your working directory is set to your "project space" as opposed to the home directory. Otherwise, you will end up receiving an email informing you that "Your home directory is over quota."

Changing the working directory can be done either by typing `setwd("~folder_name")` directly into the command line, or by using the "Set Working Directory" option from the dropdown menu, like in any other RStudio application.

For more information on the directory structure of your RCE account, please see `http://rce-docs.hmdc.harvard.edu/book/projects-and-shared-space`.

The following sections give some basic examples that may be used as scripts to run on the RCE-powered Rstudio.

3.5 Running a Serial Job (example)

The following is a dummy function executed in serial and timed for performance:

```
> f <- function () {
+         a <- rnorm(100000000)
+         b <- rnorm(100000000)
+         return(a + b)
> }
> system.time(f())
```

The same function implemented using the `foreach` package (see the package manual:

```
> library(foreach)
>
> f <- function () {
+         x <- foreach(i=1:100000000, .combine='+') %do% rnorm(2)
+         return(x)
> }
> system.time(f())
```

3.6 Running a Parallel Job on a Single Machine (example)

Now, the same function executed with parallelization (assuming that we're using a *single* multicore machine with 2 CPUs):

```
> library(foreach)
> library(doParallel)
>
> registerParallel(cores=2)
> f <- function () {
+         x <- foreach(i=1:100000000, .combine='+') %dopar% rnorm(2)
+         return(x)
> }
> system.time(f())
```

3.7 Running a Parallel Job on a Cluster (example)

Finally, the same function executed with parallelization (assuming that we're using a multicore cluster with 6 cores):

```
> library(foreach)
> library(doParallel)
>
> myCluster <- makeCluster(6)
> registerParallel(myCluster)
```

```
> f <- function () {
+         x <- foreach(i=1:100000000, .combine='+') %dopar% rnorm(2)
+         return(x)
> }
> system.time(f())
```

4 Using the Command Line

The purpose of this section is to provide a basic overview of the command line, which is needed for this workshop in order to **sync existing local project files onto the RCE server using the `ssh` protocol**.

All modern operating systems have command line interfaces that can perform the same functions on the system as the graphical user interface (GUI) that users traditionally use with some powerful additional features. Both Linux and Mac OS can access this command line via the `Terminal` application, which share command line vocabulary and features from a common 'ancestor' called the Unix operating system. Windows users can access the command line through the DOS command prompt.

Some specific examples that motivate using the command line:

- Create single keystroke shortcuts to perform repetitive tasks like opening up a set of files or navigating to a project workspace.
- Write scripts to automate `R` or Python programs based on time of day / week.
- Accessing the file system or programs on a remote machine.

4.1 Basic commands

The following table describes some basic Unix commands for Mac and Linux users:

Command	Example	Description
`ls`	`ls -l`	Lists files in current directory
`cd`	`cd tempdir`	Change current directory to `tempdir`
`mkdir`	`mkdir tempdir`	Make a folder called `tempdir`
`rmdir`	`rmdir tempdir`	Remove folder called `tempdir`
`cp`	`cp file1 file2`	Copy `file1` to `file2`
`rm`	`rm file1`	Remove or delete `file1`
`mv`	`mv old.html new.html`	Move `old.html` to `new.html` (rename)

4.2 Setting up SSH

`ssh` (Secure Shell) is a powerful program that implements the SSH protocol which securely allows remote login, access, and transfer by users over an unsecured network.

This protocol is originally a Unix protocol, and so all Mac and Linux users have `ssh` pre-installed.

If you are a Windows user, the most popular client for SSH is the program Cygwin, which can be downloaded at the developer's website, `https://cygwin.com/install.html`. During the Cygwin installation, you must choose to install OpenSSH from the Net section.

4.3 Using SSH to remotely log in

Open the terminal program that is installed by Cygwin, or Terminal on Linux/Mac. SSH uses the terminal interface to interact with other computers. There is no graphical interface for SSH, so you will need to get comfortable typing in commands.

The most basic command that any SSH client provides is to log in to a remote machine. Given a username (`<username>`) and a IP address (`<remote>`) for your remote machine, you can log in from your own machine using the command:

```
ssh <username>@<remote>
```

You will be asked for your password once the connection is established. You will not see the cursor move or any characters input when you type your password.

Once you correctly provide the credentials to access your remote machine, you can then navigate the remote's filesystem as if it were your own local file system on the terminal.

4.4 Using SSH to remotely transfer files

One of the most useful utilities of SSH is to upload and download files between your remote and local machines.

To upload a file from your local machine to your remote machine, you can use the `scp` (secure copy) command:

```
scp </local/path/to/myfile> <username>@<remote>:</desired/remote/path/for/myfile>
```

Note that you must first ensure that the remote path exists before copying your local file to that location.

Similarly to download a file from your machine to your local machine:

```
scp <username>@<remote>:</remote/path/to/myfile> </desired/local/path/for/myfile>
```

To upload or download folders, simple use the `-r` argument:

```
scp -r </local/path/to/myfolder> <username>@<remote>:</desired/remote/path/for/myfolder>
scp -r <username>@<remote>:</remote/path/to/myfolder> </desired/local/path/for/myfolder>
```

4.5 SSH with your RCE account

In order to SSH interface with your RCE account, simply use your provided RCE user name as your `<username>` credential, `rce.hmdc.harvard.edu` as the `<remote>` addresss, and enter your account's password when prompted.

5 Resources

The following are some helpful resources for different topics covered during this workshop:

- **RCE documentation page:** `http://rce-docs.hmdc.harvard.edu/book/rce-docs`
- **Using the foreach R package:** `https://cran.r-project.org/web/packages/foreach/vignettes/foreach.pdf`
- **Tips for R parallel computing:** `https://www.r-bloggers.com/how-to-go-parallel-in-r-basics-tips/`
- **R packages for advanced high-performance computing:** `https://cran.r-project.org/web/views/HighPerformanceComputing.html`
- **How to set up and use ssh:** `http://www.wikihow.com/Use-SSH`