1.5 - Digression: Some thoughts on The Linux Command Line for R Users

BIOS:7330 - Grant Brown

1 Command Lines and Batch Scripts

2 Git

Section Contents

1 Command Lines and Batch Scripts

Command Lines and Batch Scripts

- When we begin using HPC/cloud resources, we'll need to be comfortable with the Linux command line, so we'll briefly discuss it here. Please read the materials on ICON regarding the command line. (LMYTTFY)
- While both Windows and OSX provide command lines as well, for R development we can pretty much focus on the commands for Linux:
  - The Mac OS command line is extremely similar, for our purposes, running a version of Unix
  - When working with R and R packages on Windows, users need to download a piece of software called Rtools (installation instructions here). This software comes with a Windows compatible copy of the most important GNU command line tools.
  - There's even an official Linux subsystem for Windows these days.

Command Lines and Batch Scripts

- This set of notes is just intended to give some context for the broad array of things one can do by interacting with a command line environment, and to highlight some of the main R programming related tasks.
- We'll explore command line environments in more detail in interactive exercises
- Students are highly encouraged to read one of the many excellent online references and tutorials describing Linux command line environments.

Command Lines and Batch Scripts

- A command line interface is a general category of software with which we interact using text commands (like the R interpreter, for example)
- Within this broad category, we have **shells** - command line interfaces dedicated to interacting with file systems, operating system features, and software which is installed on a particular computer
- The most common Linux shell is known as 'bash', but there are others, which have all sorts of fancy features

Command Lines and Batch Scripts

- You can think of a shell like a ‘folder window’ or ‘file browser’ on Windows or OSX, except that the interface is textual rather than graphical
- Before graphical environments, the command line was the primary way users interacted with a computer. Even today, many computing environments are command line only (cloud servers, HPC environments etc.)

Command Lines and Batch Scripts

- Basic stuff you can do from a command line:
  - find out your current working directory (`pwd`)
  - list the contents of the current working directory (`ls`)
  - change directories (`cd`), or make them (`mkdir`)
  - remove files and folders (`rm`)
  - run software like R or git
- Commands can be given arguments, and generally have configurable options (set using flags, like `-a`, or `--all`)
- Commands can be chained together using the pipe operator: `|`, much like %>% in dplyr.
- Results can be redirected to files using the redirect operators like >

Command Lines and Batch Scripts

- Just like in R, command line programs can print text to the screen
- More precisely, command line programs can send text output to two important **connections**, which shell software ends up showing on the screen:
  - stdout, for general output
  - stderr, for error messages
- These can be redirected to files, which provide a log of program activity and results
- Running: `R CMD BATCH <file name>.R` will create a file called `<file name>.Rout`, containing the output from both of these connections.

R → BASH → R

There are lots of options for interacting with/from/to command line environments and the R computing environment.

- **Do command line stuff from R:** You can run commands in R using the 'system' function
- **R stuff from the command line:**
  - Open an interactive R session via 'R'
  - Submit a batch script with 'R CMD BATCH'
  - Execute an R-script with 'Rscript'
- **Exercise:** See the 'optparse' example on ICON.

Section Contents

2 Git

Git

- Better resources than I can replicate here are available online to learn Git (linked from ICON)
- Instead of presenting notes, let's go through some examples.