

Introduction to the module

Download (Windows only)

Troubleshooting

Introduction to command line

Basic command line

Command prompt

pwd

ls

ls options

mkdir

cd

Special characters

cp and mv

Renaming

Practice

Integrating data science across undergraduate STEM curriculum

2018 CTLT Spring Institute

Dr. Kim Dill-McFarland


version May 06, 2018

Introduction to the module

This module assumes no prior knowledge of the tools used.

Learning objectives After this module, students will be able to:

- Identify uses of command line inside and outside the classroom
- Apply basic syntax and functions in command line
- Use command line to navigate a computer's file structure

Throughout this tutorial, look for the  to denote lines of code or other steps that you should run on your computer if you are following along.

This module and all materials herein were developed as part of the Experiential Data science for Undergraduate Cross-Disciplinary Education (EDUCE (????????)) initiative at the U. of British Columbia.

Download (Windows only)

Prior to starting this module, you will need to download a Linux-based command line terminal. We will use the free, open-source GitBash terminal, which you can download here (<https://git-for-windows.github.io>).

Mac machines come with a Linux-based command line terminal pre-installed.

Troubleshooting

- Try installing Git as an administrator by right-clicking the download and selecting “Run as administrator”
- Try restarting your machine after installation is complete.

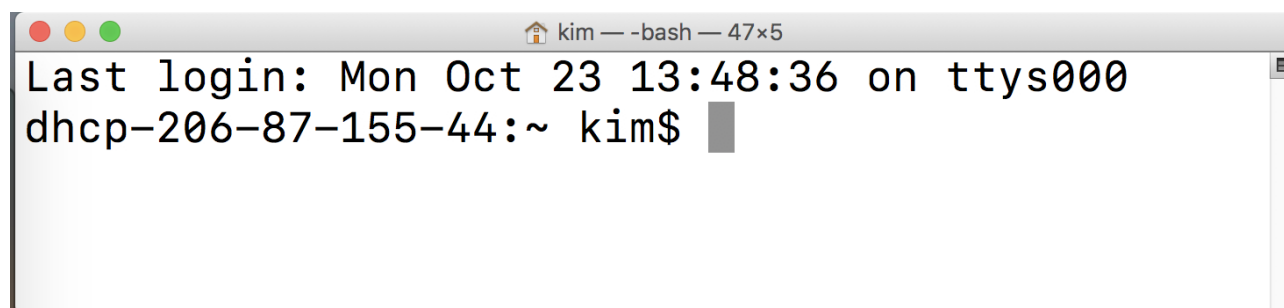
Introduction to command line

Basic command line

Data science most often deals with big data, as in terabytes worth of it. Specific programs and computers are needed to collect, analyze, and store these data. Many (and in some fields, most) of these tools require the use of command line to either run a program or access a remote super computer. So, a great place to begin in data science is to become familiar with command line.

Command-line allows you to interact with your computer through text-based commands input through a simple program called a **shell**.

First open a command line window on your computer. This will be the built in command line (Terminal) for Mac or Linux machines



or GitBash for Windows machines.



If you do not know where this program is on your computer, simply search for the appropriate name in Finder (Mac) or the start menu (Windows).

Throughout this tutorial, you will generally see a Mac Terminal as the example output. If something looks significantly different in GitBash, we will also show this window.

Command prompt

Notice the `$` symbol in the above windows. This is the command prompt which lets you know that the terminal is ready for you to input your next command.

pwd

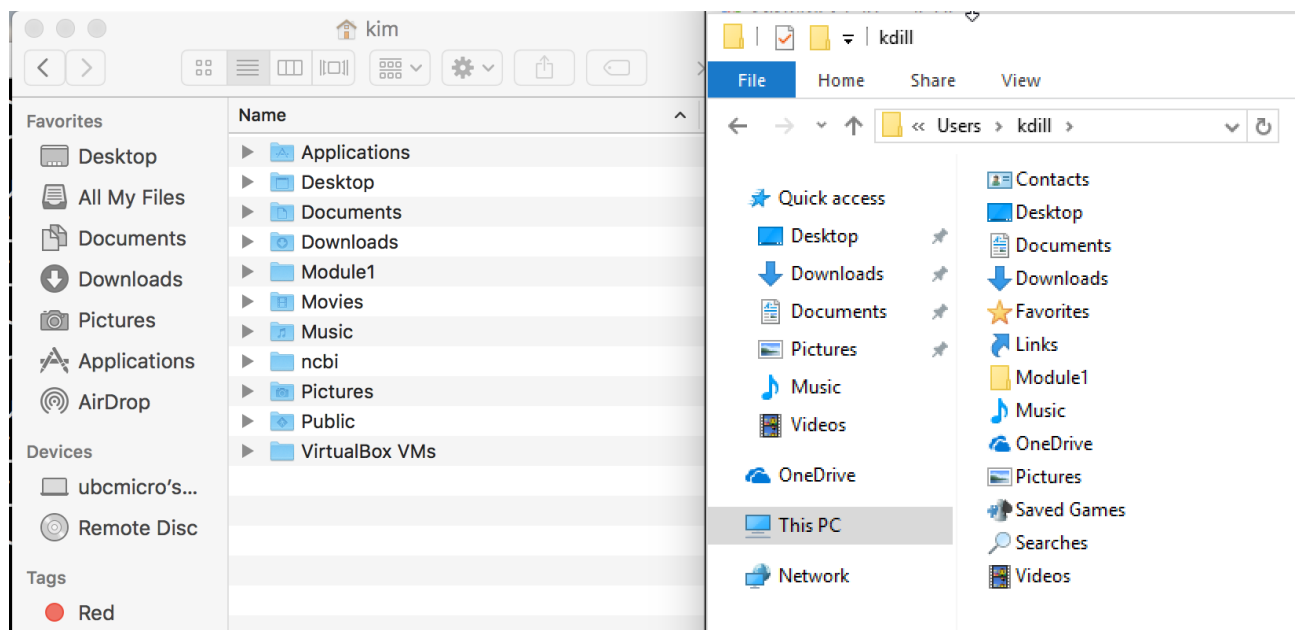
When you open a new terminal, you will automatically be in your `home` directory. To find out where this is, use `pwd` which stands for **print working directory**. This tells you where your terminal is currently pointing.



pwd

```
kim — -bash — 47x5
Last login: Mon Oct 23 13:48:36 on ttys000
[dhcp-206-87-155-44:~ kim$ pwd
/Users/kim
dhcp-206-87-155-44:~ kim$
```

We see that we are in `/Users/kim`, which we can also go to in the more commonly used Finder programs on Mac (left) or Windows (right).



I will start most Terminal screenshots with `pwd` so that we are able to orient ourselves within the file structure before performing a new step.

ls

We can also orient ourselves within the current working directory by **listing** all the files in that directory with `ls`.



```
kim — -bash — 47x28
[dhcp-206-87-155-44:~ kim$ ls
Applications      Movies
Desktop           Music
Documents         Pictures
Downloads         Public
Google Drive     VirtualBox VMs
Library           mothur
Module1
dhcp-206-87-155-44:~ kim$
```

You can see that the files listed in the terminal are the same as in the Finder window above.

ls options

For those working in GitBash, you will automatically see some additional information with `ls`.

```
MINGW64:/c/Users/kdill
kdill@DESKTOP-2ROMA1P MINGW64 ~
$ ls
AppData/
'Application Data'@
Contacts/
Cookies@
Desktop/
Documents/
Downloads/
Favorites/
Links/
'Local Settings'@
Music/
'My Documents'@
NetHood@
NTUSER.DAT
ntuser.dat.LOG1
ntuser.dat.LOG2
```

GitBash colors files by type as well as includes indicators at the end of some types. Folders are blue and end in `/`, symbolic links (like shortcuts) are teal and end in `@`, files are grey, etc.

You can ask for these indicators in the Mac/Linux Terminal by adding modifiers to the `ls` command. You do this with `-l` after the command. For example, if we want to show all of the file type indicators, we use `ls -l` to list showing folders.

```

kim — -bash — 47x28
dhcp-206-87-155-44:~ kim$ ls
Applications      Movies
Desktop           Music
Documents         Pictures
Downloads         Public
Google Drive     VirtualBox VMs
Library           mothur
Module1

dhcp-206-87-155-44:~ kim$ ls -F
Applications/     Movies/
Desktop/         Music/
Documents/       Pictures/
Downloads/       Public/
Google Drive/    VirtualBox VMs/
Library/         mothur/
Module1/

dhcp-206-87-155-44:~ kim$

```

Most commands can be modified with text following a - as in `ls -F`. You can learn more about these modifications by looking up the **manual** of any function with `man` like `man ls`.

```

kim — less + man ls — 71x28

LS(1)                                BSD General Commands Manual
LS(1)

NAME
    ls -- list directory contents

SYNOPSIS
    ls [-ABCFGHLOPRSTUW@abcdefghijklmnopqrstuvwxyz1] [file ...]

DESCRIPTION
    For each operand that names a file of a type other than directory,
    ls
    displays its name as well as any requested, associated information
    . For
    each operand that names a file of type directory, ls displays the
    names
    of files contained within that directory, as well as any requested
    , asso-
    ciated information.

```

This will bring up a text documents with every possible modifier of the `ls` function. You then **exit the manual** back to the terminal with `q` for 'quit'.

mkdir

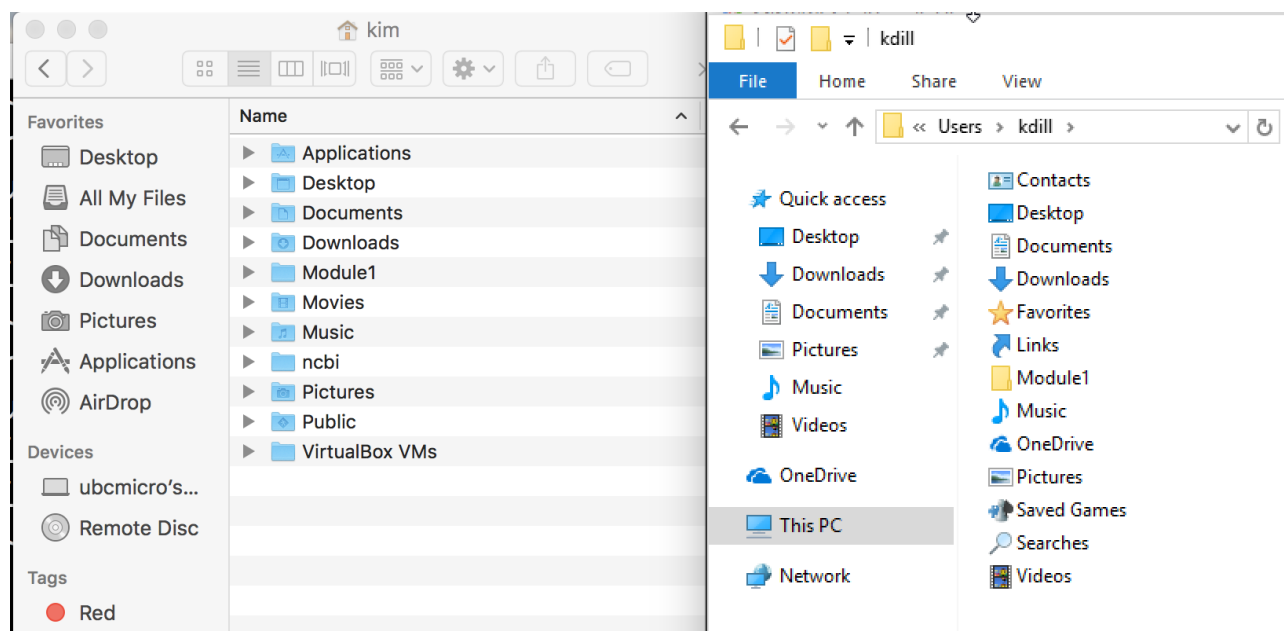
To stay organized, let's make a directory called 'Module1' for all of our work in today's tutorial. This is done with `mkdir` which stands for **make directory**.



```
mkdir Module1
```

```
kim — -bash — 47x28
dhcp-206-87-155-44:~ kim$ mkdir Module1
dhcp-206-87-155-44:~ kim$
```

The terminal doesn't output anything showing us that this directory was created. If we open Finder (Mac, left) or File Explorer (Windows, right), we can see that we do have a directory named Module1.



cd

So now that we've made a new directory, we can move into that directory with `cd` for **change directory** and see that we've actually moved with `pwd` again. Note that the command line is case sensitive so `Module1` \neq `module1`



```
cd Module1/
```

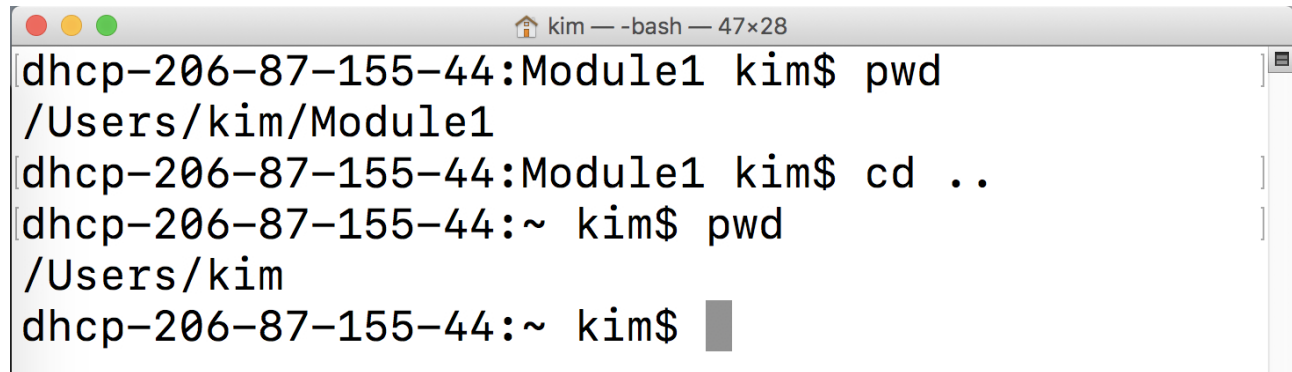


```
pwd
```

```
Module1 — -bash — 47x28
dhcp-206-87-155-44:~ kim$ pwd
/Users/kim
dhcp-206-87-155-44:~ kim$ cd Module1/
dhcp-206-87-155-44:Module1 kim$ pwd
/Users/kim/Module1
dhcp-206-87-155-44:Module1 kim$
```

Special characters

You then have a couple of options for getting out of this directory. You can go **up one directory** using `cd ..`. In this case, we would go back to our home directory since this is one up from our `Module1` directory.

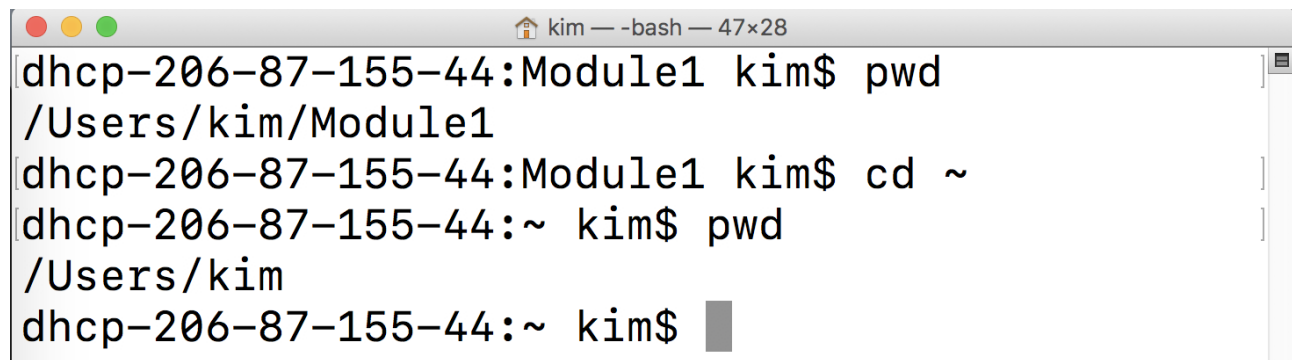


```

kim — -bash — 47x28
[dhcp-206-87-155-44:Module1 kim$ pwd
/Users/kim/Module1
[dhcp-206-87-155-44:Module1 kim$ cd ..
[dhcp-206-87-155-44:~ kim$ pwd
/Users/kim
[dhcp-206-87-155-44:~ kim$

```

Or you can go straight to the home directory from anywhere, no matter how many directories up it is from your current location, using `cd ~`. The `~` is shorthand for your **home directory**.



```

kim — -bash — 47x28
[dhcp-206-87-155-44:Module1 kim$ pwd
/Users/kim/Module1
[dhcp-206-87-155-44:Module1 kim$ cd ~
[dhcp-206-87-155-44:~ kim$ pwd
/Users/kim
[dhcp-206-87-155-44:~ kim$

```

Choose one to run on your computer so that you are back in your `home` directory.




```
cd ..
```


cp and mv

We will now add a file to our new `Module1` directory. Choose any file on your computer; I will use `Notes.docx` on my Desktop. We can either **copy** the file with `cp` or **move** it with `mv`. In either case, the syntax is `cp_or_mv where_the_file_is where_you_want_the_file_to_go`

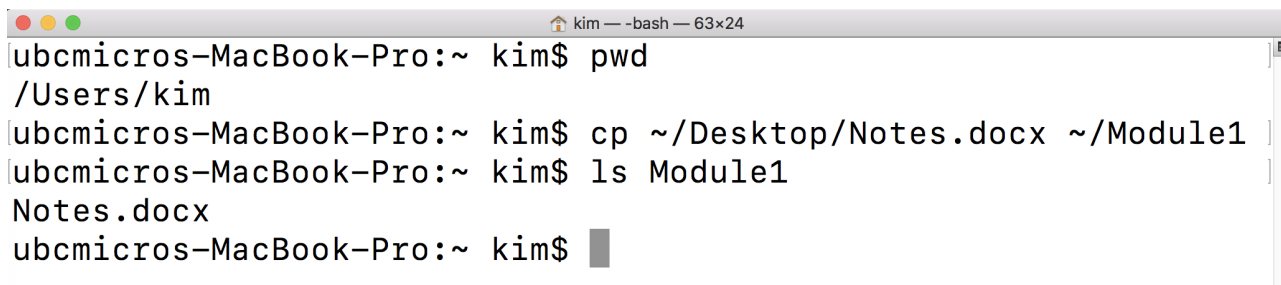
To be safe, I will copy my notes file so that the original remains on my Desktop. I then check that the file has been copied by listing everything in the `Module1` directory with `ls Module1`. You could also check by `cd` ing into `Module1` and then using just `ls`.



```
cp ~/Desktop/Notes.docx ~/Module1
```



```
ls Module1
```


A screenshot of a macOS Terminal window. The title bar shows the window name 'kim' and its dimensions '63x24'. The terminal text shows a user at 'ubcmicros-MacBook-Pro' with a shell prompt 'kim\$'. The user enters 'pwd' and the output is '/Users/kim'. Then, the user enters 'cp ~/Desktop/Notes.docx ~/Module1' and 'ls Module1', with the output showing 'Notes.docx'. The prompt returns to 'kim\$' with a cursor.

```
ubcmicros-MacBook-Pro:~ kim$ pwd
/Users/kim
ubcmicros-MacBook-Pro:~ kim$ cp ~/Desktop/Notes.docx ~/Module1
ubcmicros-MacBook-Pro:~ kim$ ls Module1
Notes.docx
ubcmicros-MacBook-Pro:~ kim$
```

Notice how instead of writing out the entire file path like `Users/kim/Desktop/Notes.docx`, I can use the shortcut `~` for my home directory which is `Users/kim`.

Renaming

You can also rename files as you copy/move them. Simply give the destination path a new name!

```
cp ~/Desktop/Notes.docx ~/Module1/Notes_new.docx
```

Practice

Command line skills come with practice. Start the following exercises in class and then complete outside of class as needed before our next data science tutorial. Please feel free to work together!

1. Move around your computer using the command line. Try to find the following directories.
 - Desktop
 - Downloads
 - Where the Terminal (Mac) or GitBash (Windows) program is installed
 - A USB or other external drive (make sure to plug one into your computer first)
2. Move the `Module1` directory we created today to wherever on your computer you would like it to reside. This could be in “My Documents”, on your Desktop, etc.
3. Retrieve a file from the internet and put it in your `Module1` directory using only the command line. *Hint*: you can `cp` from a web address.
4. Select 1 directory where you would like to improve the organization. Using the command line, reorganize and rename the files within until you are happy with the new structure. Take a screenshot of `ls` within your chosen directory both before and after.