

DSCI 521 - Computing Platforms for Data Science

Admin stuff

- If you haven't done so already, please sign in at www.github.ubc.ca so your account is created.
- There will be two extra IT Help Desk drop in sessions (see [MDS Calendar](#)):
 - Wed, September 11, from 12:30 to 13:30, in ICCS 238
 - Wed, September 18, from 12:30 to 13:30, in ICCS 238

High-level goal of this course:

Get you up to speed on how to use the essential software pieces of our Data Science stack.

Essential Data Science stack we will be using

Software	What for?	Why?
Bash Shell	<ul style="list-style-type: none">• Navigate files system• Install Software• Run automated scripts and pipelines	<ul style="list-style-type: none">• Makes automation easy• Has stood the test of time• Open Source
VS Code	Edit text documents	<ul style="list-style-type: none">• General purpose and modern

Software	What for?	Why?
		<ul style="list-style-type: none"> • Easy to install • Active development • Open Source
Git	Manage the Versioning of your projects	<ul style="list-style-type: none"> • Fits well with Github • Local and Remote • (Most?) Popular • Open source
Python	Programming for: <ul style="list-style-type: none"> • Statistics 	<ul style="list-style-type: none"> • Popular

Software	What for?	Why?
	<ul style="list-style-type: none"> • Visualization • Machine Learning 	<ul style="list-style-type: none"> • Active development community • Readable • Open Source
R	Programming for: <ul style="list-style-type: none"> • Statistics • Visualization • Machine Learning 	<ul style="list-style-type: none"> • Popular • Active development community • Open Source
Jupyter	Interactively develop: <ul style="list-style-type: none"> • Python • R • Julia 	<ul style="list-style-type: none"> • Popular • Great for

Software	What for?	Why?
		<p>analytic reports</p> <ul style="list-style-type: none"> • Open Source
RStudio	Interactively develop R projects	<ul style="list-style-type: none"> • Popular • Great for script development • Great for analytic reports • Has notebook feature • Open Source
















note - there are other programs we will install and use later (e.g., Make, PostgreSQL, Docker, etc), but those listed above are the essentials

Why do we focus on Open Source?

- You will be able to use the Software after you leave school without paying exorbitant amounts of money;
- You have access to the source code for the Software you depend on - you can find out/test how good it is (*even closed source, Commercial Software has bugs, they are just harder to see...*);
- Open Source software is customizable and flexible;
- It's becoming the standard in many trailblazing companies;
- You are part of an awesome and active community!

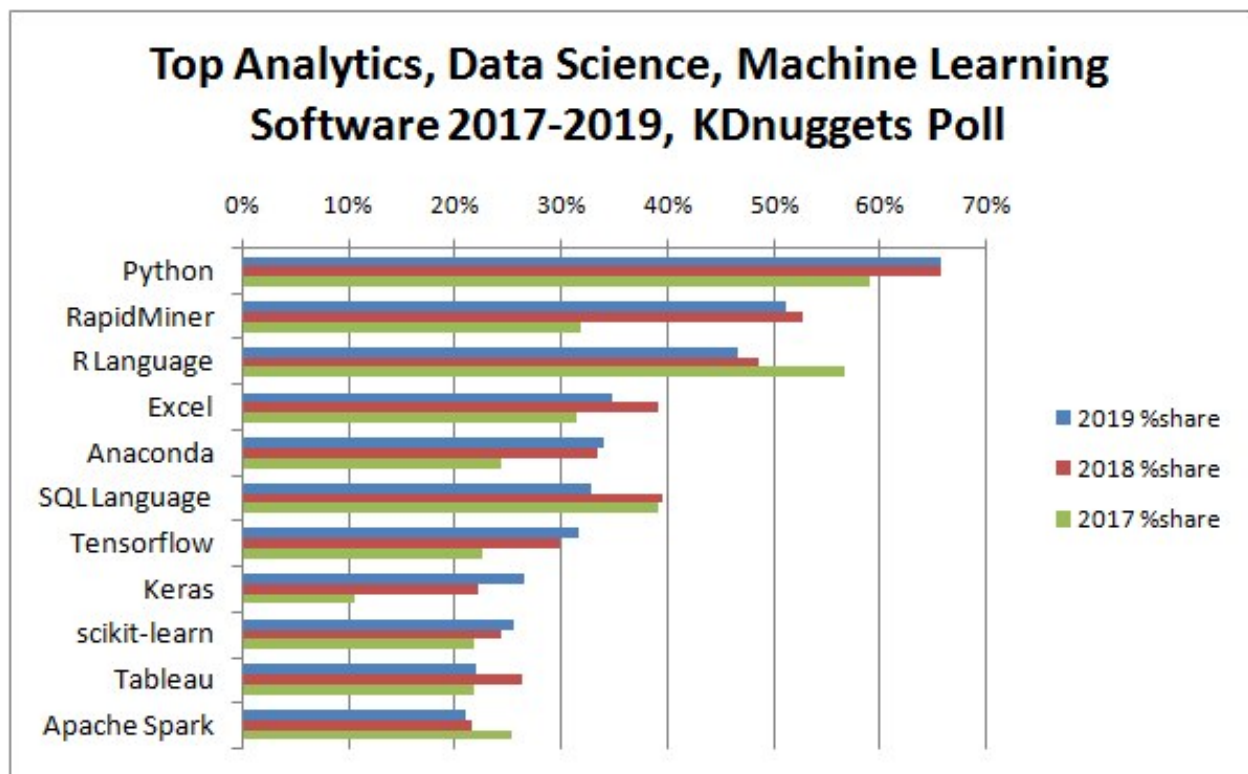
Why do we focus on Python & R?

- Both are in the top 10 most popular programming languages

Rank	Language	Type	Score
1	Python	  	100.0
2	Java	  	96.3
3	C	  	94.4
4	C++	  	87.5
5	R		81.5
6	JavaScript		79.4
7	C#	   	74.5
8	Matlab		70.6
9	Swift	 	69.1
10	Go	 	68.0

source: <https://spectrum.ieee.org/computing/software/the-top-programming-languages-2019>

- Well suited for a large range of applications
- Additionally, they are in the top 3 most frequently used analytics / data science software



source: <https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html>

- Growing and (mostly) welcoming communities.
- But even big tech companies that created their own programming languages use this? DEFINITELY!!!! Not only that, but they also contribute to these languages. For example,
 - Tensorflow from Google;
 - Pytorch from Facebook;
 - Microsoft actually created a distribution of R, named `Microsoft R Open`;

Course syllabus:

- https://github.ubc.ca/MDS-2019-20/DSCI_521_platforms-dsci_students/blob/master/README.md

By the end of this lecture, you should be able to:

1. [Use the Unix command line to navigate the computer's filesystem;](#)
2. [Define and distinguish between absolute file paths and relative file paths;](#)
3. [Acknowledge the importance of version control;](#)
4. [Use git's basic functions: clone, add, commit, pull, and push](#)
5. [Use Git and GitHub to access, edit and submit MDS homework.](#)
6. [Create, edit and run Jupyter notebooks \(a type of reproducible literate code document\).](#)

1. Navigating and manipulating the filesystem using the Shell

- Navigating the filesystem can be done via your visual operating system's user interface (e.g., Finder, Explorer, Nautilus, etc) OR using the Command line/terminal/unix shell (it comes from the Unix/Linux world, but it's available for everybody)

Useful unix shell commands for navigating and manipulating the filesystem:

Comm and	Purp ose	Example use
<code>pwd</code>	Prints curre nt worki ng direct ory	<code>pwd</code>
<code>ls</code>	List conte nts	<code>ls Documents</code>
<code>cd</code>	Chan ge direct ory	<code>cd Desktop</code>
<code>mkdir</code>	Creat e a direct ory	<code>mkdir my_new_directo ry</code>
<code>rm</code>	delete a file	<code>rm file_to_be_del eted</code>

- Also, you can press `TAB` for autocomplete;

Useful unix shell shortcuts and wildcards:

Symbol	Definition
<code>.</code>	Current working directory

Symbol	Definition
..	Parent directory
~	HOME directory
*	Wildcard, matches any character

Example

Let's try each of these commands.

Exercise

Use command line to:

1. figure out where you are when you open your command line
2. navigate to your Documents folder
3. navigate from your Documents folder to your Desktop

note: if you want to learn more about Unix shell, take a look at the [Software Carpentry lessons on Unix Shell](#);

1.1 Reading the Shell command docs

- Depending on the program that runs your Shell, you can get documentation about a command and its optional flags/arguments via `man` or `--help`.
- If you use the `man` option, type `q` to exit the help page.
- For example, to learn about the `pwd` command, type:

```
man pwd
```

And you will see something like this printed to your shell/command line:

```
PWD(1)                                BSD General Commands Manual
      PWD(1)
```

```
NAME
```

```
pwd -- return working directory name
```

```
SYNOPSIS
```

```
pwd [-L | -P]
```

```
DESCRIPTION
```

```
The pwd utility writes the absolute pathname of the current working
directory to the standard output.
```

```
Some shells may provide a builtin pwd command which is similar or
identical to this utility. Consult the builtin(1) manual page.
```

The options are as follows:

-L Display the logical current working directory.

-P Display the physical current working directory (all symbolic links resolved).

If no options are specified, the -L option is assumed.

ENVIRONMENT

Environment variables used by pwd:

PWD Logical current working directory.

EXIT STATUS

The pwd utility exits 0 on success, and >0 if an error occurs.

SEE ALSO

builtin(1), cd(1), csh(1), sh(1), getcwd(3)

STANDARDS

The pwd utility conforms to IEEE Std 1003.1-2001 ('`POSIX.1').

BUGS

In csh(1) the command dirs is always faster because it is built into that shell. However, it can give a different answer in the rare case that the current directory or a containing directory was moved after the shell descended into it.

The -L option does not work unless the PWD environment variable is exported by the shell.

BSD

April 12, 2003

BSD

Example

Let's check out the documentation of ls. What does the option -a do?

2. Relative vs absolute path

You are here:

```
pwd
/Users/timberst/Documents
```

These are the files in your current working directory:

```
ls
DSCI-100  MDS-2018-19  textbooks  research
```

You can navigate to the research directory using one of two options, a relative path:

```
cd research
```

or, using an absolute path:

```
cd /Users/timberst/Documents/research
```

What is the difference? When you use a relative path with a command like `ls` or `cd`, it tries to find that location from where we are, rather than from the root of the file system. However, when you specify the absolute path to a directory by including its entire path from the root directory, which is indicated by a leading slash, The leading `/` tells the computer to follow the path from the root of the file system, so it always refers to exactly one directory, no matter where we are when we run the command.

Discussion question:

When should I use a relative path versus an absolute path?

3. Introduction to local and remote version control software

Why version control?

source: "Piled Higher and Deeper" by Jorge Cham, <http://www.phdcomics.com>

Additional reasons to learn version control (especially using Git & Github)

Beyond managing and keeping versions of documents (reports, code, etc) under control, there are other reasons for using the Git/Github version control system:

- Github (website) can act as a back-up for files housed there
- Github can be used to host websites/blogs
- Github has a fantastic search functionality

And Git is the language we have to speak to communicate with Github. Sometimes it is a quirky language...

Github as a means for submitting your homework/assignments and quizzes

- We use Github to submit homework/assignments and quizzes
- Today as we go through the lesson material we will refresh all the essentials you need to submit your homework assignment

Exercise

Let's access your own GitHub home for MDS. There you can find the links for all the courses' repositories, as well as, the labs' repositories. Your home is located at:

```
https://github.ubc.ca/MDS-2019-20/yourCWL_home
```

where `yourCWL` should be replaced by your `CWL`.

(Hint: you might want to bookmark this page as it gives you easy access to all relevant courses' repositories).

Setting up Git

Using the command line to tell Git about yourself:

```
git config --global user.name 'Tiffany Timbers'
git config --global user.email 'tiffany.timbers@stat.ubc.ca'
git config --global --list
```

The top two `git config` commands return nothing. So we add the third line to check that Git understood what we typed.

Exercise

Use these commands to set up your own Git. (**Hint:** remember that you are not Tiffany! Use your own info.)

note - if you have previously set up Git you can likely skip this step

Cloning repositories from Github

- If you want to be able to run the code on your own machine, and modify it there, we need to "clone" it.
- To clone it:
 1. go into the repository (your copy of it)
 2. click on the green "Clone or download" button (make sure the pop-up says "Clone with HTTPS")
 - note - we will learn to use SSH authentication later...
 3. copy the URL to the clipboard
 4. open a Unix Shell instance (e.g., terminal or Git for Windows) on your laptop
 5. navigate to your `Documents` directory

```
cd Documents
```
 6. type `git clone` and paste the URL

```
git clone https://github.com/github_username/repository_name.git
```
 7. press enter to clone the repository to your local machine

For more details, see Github's docs on [cloning](#).

Adding & committing changes to version controlled files

There are two ways to make changes to your files:

1. Make changes on files you have cloned locally to your computer, and then "push" the changes back to Github.
2. Edit files directly on Github.

Making changes locally

- Once you have made a change to a file in a Git repository that you have locally (*i.e.*, on your laptop), you need to commit the changes.
- Committing the changes is essentially saving the differences between the new version of the document and the previous version of the document in the hidden `.git` directory in the Git repository.
- This is a two stage process with Git:
 1. Add changes to staging area

```
git add file_name(s)
```
 2. Commit changes to hidden `.git` directory

```
git commit -m "Some useful message about the changes you made"
```

For more details, see Software Carpentry's lesson on [tracking local changes with Git](#).

What is the staging area, and why is it there?

How do I know if I made changes?

`git status` is a friendly command that lets you know if you have changes that you have yet to add or commit.

Push local changes to Github

Once you have made local changes, and added and committed them, you need to "push" the changes to Github:

```
git push
```

For more details, see Github's doc on [pushing changes to Github](#).

ALWAYS VERIFY YOUR HOMEWORK GOT PUSHED!

4. Create, edit and run Jupyter notebooks (a type of reproducible literate code document)

- Useful Jupyter's Lab shortcuts (these are shortcuts I use (almost) every day):

•	Shortcut	description
b	adds cell	
x	removes cell	
Shift+Enter	runs cell and moves to the next cell (creates a cell if there isn't a next cell)	
CTRL+Enter	runs cell but keeps the cursor in	

•	Shortcut	description
	the same cell	
z	undo cell operation	
m	changes a cell to markdown	
y	changes a cell to code	

- You can check the shortcuts using the Command palette on the left bar (or press CTRL+ALT+C).

Example

Let's explore Jupyter a little bit together.

Exercise

Start working on [lab 1](#)!

Concept map wrap-up

- Use the Unix command line to navigate their computer's filesystem.
- Define and distinguish between absolute file paths and relative file paths.
- Use local and remote version control software (e.g., Git and GitHub) to access, edit and submit MDS homework.
- Create, edit and run Jupyter notebooks (a type of reproducible literate code document).

Attribution

- [Happy Git and GitHub for the useR by Jenny Bryan and the STAT 545 TAs](#)
- [Software Carpentry](#), specifically the Unix Shell and Git lessons