UBC Dynamic Brain Circuits Cluster Onboarding Portfolio Documentation

Release 2019

Glaynel Alejo

Aug 26, 2022

Contents:

CHAPTER 1

Introduction

Dynamic Brain Circuits in Health and Disease

Welcome to the Dynamic Brain Circuits in Health and Disease research cluster! The Brain Circuits cluster is composed of 30 researchers focused on the study of brain connections and their dynamic changes during development, learning, and disease.

This onboarding portfolio is intended to introduce new cluster members and neuroscience graduate students to tools and resources that they may need as they begin their research at UBC. The portfolio is designed as a quick start guide and is therefore recommended as a reference for unfamiliar things that new members may encounter or new skills they may need to acquire.

Most of what is presented here is central to neuroscience research and the portfolio encourages the use of free resources that are catered directly to the needs of neuroscientists.

Note: The cluster has released a white paper on data amanagement, including data storage and sharing. Check it out here!

1.1 Labs and Investigators

Timothy Murphy (cluster lead)

Brian MacVicar

Ann Marie Craig

Fidel Vila-Rodriguez

Cheryl Wellington

Shernaz Bamji

Lara Boyd

Paul Pavlidis

Martin Mckeown

Jon Stoessl

Peter Cripton

Jason Snyder

Vesna Sossi

Wolfram Tetzlaff

Anthony Phillips

Catharine Winstanley

Yu Tian Wang

Todd Woodward

Matthew Farrer

Jeremy Seamans

Terry Snutch

Ian Mackenzie

Lynn Raymond

Kurt Haas

Mark Cembrowski

Jane Roskams

Fabio Rossi

CHAPTER 2

Cluster Resources

Table of Contents

2.1 Social Media and Accounts

- Email: brain.circuits@ubc.ca
- Website
- Twitter: @BrainUBC

The following links appear throughout the portfolio:

- Dataverse
- Federal Research Data Repository (FRDR)
- GitHub
- OSF

2.2 NeuroImaging and NeuroComputation Center (NINC)

The NeuroImaging and NeuroComputation Center (NINC) provides imaging and computational support to the Djavad Mowafaghian Centre for Brain Health (DMCBH) community. It is located in the DMCBH Koerner Laboratories.

Some of NINC's services include MATLAB and Python tutorials, imaging and light microscopy systems training, as well as access to software and high performance computing infrastructure.

Note: Check out NINC on GitHub for helpful resources.

2.2.1 Databinge

Databinge is a weekly drop-in neuroscience meeting where interested labs can have a data show-and-tell, problem-solving session, conference report, and/or technique presentation using the facilities of the NINC. Databinge aims to promote practical training and collaboration among the cluster labs.

Databinge is at 4:00-5:30 pm on Fridays in Koener F103. Check out the calendar to see the schedule.

Attention: There's free pizza!

CHAPTER 3

Beginner's Guide to Coding

Table of Contents

3.1 MATLAB <–> Python

```
if command == 'matlab'
        command = 'python';
end
```

```
if command == 'python':
        command = 'matlab'
```

Take a look at the code blocks above. One is in MATLAB and the other is in Python. Which one is which?

Python and MATLAB have very similar syntax so knowing one while learning the other can get tricky. It's easy to mix up the commands.

A couple of resources are listed below to help users of both languages find equivalent commands:

- MATLAB commands in numerical Python from Mathesaurus by Vidar Bronken Gundersen
  - HTML
  - PDF
- NumPy for Matlab users from SciPy.org

3.2 Good Practices

Learning how to code can be frustrating but it is ultimately rewarding. As a beginner, it is best to form good habits early. Additionally, knowing common mistakes and some basic tricks can save a lot of time and effort.

3.2.1 Format as you go

Sometimes when you're in the zone, the last thing you're worried about is your coding style. As a result, your code comes out looking like a wild beast which is a nightmare to tame when it spits out errors.

Unformatted code is harder to debug because it's harder to read. It's therefore best if you format as you write.

Tip: In Python, *Pythonic* code is good code - see the section on *pycodestyle* for the specifics of Python style. Advanced coders can take advantage of Black, a Python formatter.

MATLAB does not have an official style guide. However, there is a popular document called MATLAB Style Guideline 2.0 written by Richard Johnson for those interested in polishing up their code.

3.2.2 Comment wisely

Comments are essential because they describe the purpose of your code. It's important to write code that future you and other people will undestand. For example, functions should always be accompanied by a description.

It's good to comment frequently but do so with some discretion - try not to comment the obvious.

Hint: Comments begin with a `#` in Python and a `%` in MATLAB.

3.2.3 Don't just copy from online sources

It's easy to stitch together a script from code snippets you find online but it's not a productive learning experience. A valuable part of learning how to code is problem-solving, so it pays off to figure it out on your own first.

If you're really stuck and need help, at least try to understand what each line of code does before using it.

3.2.4 Write test cases: Catch me if you can!

In programming, it is best to be proactive - know when and how your code will fail. Run test cases through your functions to ensure that they give the expected result in every scenario.

This is especially important if the code you are producing will be used by others so you can catch the errors and prevent your program from crashing on users.

Attention: Don't forget to address the edge cases!

3.2.5 Back it up

Coding takes a lot of work so you don't want to lose your progress when your program crashes. This lesson is painful to learn by mistake so back up your code regularly or better yet, version it!

See the section on *Version Control* in the Data Management Page.

3.3 Debugging

> "90% of coding is debugging. The other 10% is writing bugs" – @bramcohen

There is a good reason why this tweet has thousands of retweets and likes - because it's sometimes true. Here are some tips when expectation != reality when coding.

3.3.1 Consult the documentation

Debugging can be as easy as checking the documentation. When the output of your code doesn't make sense, it might be because:

- a command you're using does not do what you think it does
- you're not providing the correct input argument(s) to a command
- you forgot to specify a value for a default argument of a command

For all these cases, reading the documentation carefully will likely resolve the issue.

Attention: Before writing your own function, check if a command already exists for the result you require!

3.3.2 Learn to read and write error messages

As with everything else in life, coding requires you to learn from your mistakes but before you can learn from them, you must first know exactly what those mistakes are and what caused them.

Errors can be more puzzling than the code that caused them. However, often times, they are actually helpful and descriptive - that is, if you know how to read them. It takes practice but with time and experience, you'll have an inkling about what's wrong with your code even with the vaguest of error messages.

Before you start coding, take some time to read up on the documentation about error handling.

- Errors and Exceptions in Python

- Error Handling in MATLAB

Knowing how to read error messages will also help you once you start writing your own programs. To make sure your programs are user-friendly, make sure you anticipate potential mistakes your users may make so you can write clear error messages to help them (remember *this* good practice?).

3.3.3 Diagnose with print statements

One of the simplest ways to find out where your code is going wrong is to insert print statements between lines.

Example

```
>>> for item in some_range:
...:     # insert step 1 here
...:     print("Step 1 complete!")
...:     # insert step 2 here
...:     print("Step 2 complete!")
```

When you run the for loop, you'll know exactly if and where a bug occurs by which print statement fails to appear.

Tip: Python has a standard logging module, which can track events while a program runs. Logging can be used to check and record the execution of your code, including errors and warnings. This is a more robust and efficient approach to debugging than print statements.

```
>>> import logging
```

3.3.4 Think first, then Google

It's tempting to Google an error straightaway but this prevents you from building an intuition for debugging. Spend some time thinking about the logic of your code first; often times, there is a disparity between what you want your computer to do and what it is actually doing.

Doing this will familiarize you with solutions to errors that may pop up again in the future and it may even help you foresee the same errors before you hit *Run*.

3.3.5 Use a debugger

A debugger should only be your last line of defense. For beginners, it is better to establish your ability to diagnose problems before relying on a debugger.

The Python Debugger (pdb)

pdb is a module from the Python Standard Library that interactively debugs Python programs. This module enables coders to step through their programs line-by-line as it executes.

- How To Use the Python Debugger, a tutorial that highlights the features of pdb written by Lisa Tagliaferri for DigitalOcean
- pdb Tutorial, a tutorial that uses an example of a Python script for a dice game to illustrate the use of pdb.

Note: A **breakpoint** is used to pause a running program at a specific location and *break* into the debugger, allowing you to step through the code following the breakpoint.

In Python, a breakpoint can be hard-coded into a program by first importing the pdb module then inserting the breakpoint method above the line at which you would like the debugging session to begin.

```
import pdb
# some code in between
pdb.set_trace()
# debugger begins at this line
```

Tip: For versions 3.7 and onwards, a built-in `breakpoint()` function can replace `import pdb; pdb.set_trace()`.

Debugging in MATLAB

MATLAB automatically generates warnings for lines that may cause errors. These warnings are indicated by orange highlight within the script and as orange lines in a narrow sidebar on the right-hand side of the Editor.

Debugging in MATLAB can be done in one of two ways:

1. point-and-click through the Editor/Debugger: Debug a MATLAB Program
2. through debugging functions in the Command Window: Debugging and Analysis

Tip: In MATLAB, `dbstop` is the equivalent function to `breakpoint()` in Python.

CHAPTER 4

MATLAB

MATLAB is a computing and visualization environment and a high-level programming language.

Table of Contents

4.1 Installation

UBC has a campus-wide MATLAB license hence it is free for students, faculty, staff, and researchers.

To install MATLAB, create a MathWorks account with your UBC email, which automatically links your account with UBC's license. Once you have verified your email address, log into MathWorks. Your account page should display the license under "My Software". Click the down arrow located in the same row as the license to download MATLAB.

For more information on MATLAB installation and troubleshooting, check out UBC IT's MATLAB page.

4.2 Tutorials

4.2.1 MATLAB Onramp

MATLAB Onramp is a free, interactive online tutorial offered by MATLAB.

This 2-hour tutorial aims to familiarize new users with the functions and syntax of MATLAB through hands-on excercises with feedback and hints.

A MATLAB Onramp Certificate is generated with a percentage indicating progress in the course.

4.2.2 NINC MATLAB Tutorials

The NeuroImaging and NeuroComputation Centre (NINC) offers free MATLAB tutorials in Koerner F103. The tutorial materials are available on the NINC GitHub.

If interested in participating in these tutorials, please contact Jeffrey LeDue through email (jledue@mail.ubc.ca).

Tip: To download a repository from GitHub, click on the green button 'Clone or download' then click 'Download Zip'.

4.3 MATLAB Online

A MathWorks account enables access to MATLAB from a web browser through MATLAB Online. Files uploaded into MATLAB Online are stored in MATLAB Drive so they can be accessed upon future log-ins.

Note: 5GB of storage is available in MATLAB Drive.

This platform is convenient for quick access on workstations without MATLAB. It is also recommended for sharing and teaching purposes, such as workshops, where it would be ideal for all users to have the same and/or latest version of MATLAB.

CHAPTER 5

Python

Python is a high-level, interpreted programming language that is widely used for scientific analysis and computation.

Table of Contents

5.1 Installation

Mac OS X and some Linux distributions have Python installed by default.

Windows users can download the latest version of Python from the website.

Once installation is done, click the Python application file again (it should be in the Downloads folder as python-<version_number>.exe). A window will pop up. Click on *Modify* then select *Next*, which leads to *Advanced Options*.

Tick the box labeled *Add Python to environment variables* then click *Install*. Python will now be avaliable in both Command Prompt and Windows PowerShell.

To verify installation or to check the version of Python on your system, open a Terminal window (Mac OS X) or Command Prompt (Windows) and enter the command below.

```
$ python --version
```

5.2 Tutorials

5.2.1 NINC Python Tutorials

The NeuroImaging and NeuroComputation Centre (NINC) offers free drop-in Python tutorials every Friday from 3-4 pm in Koerner F103.

You can ask for help with your own code or get some guidance on how to get started.

5.2.2 Oregon Health & Science University's Python Neurobootcamp

The Neuroscience Graduate Program at Oregon Health and Science University (OHSU) hosted a 5-day Python Neurobootcamp, the materials from which are available on GitHub.

The course is structured around five Jupyter Notebooks:

- Day 1: Introduction to the Python Language: Data types and handling errors
- Day 2: Introduction to Pandas Data Frames: Intro to imaging data analysis
- Day 3: More on Pandas and Numpy: Electrophysiology data analysis
- Day 4: Diving Deeper into Pandas with Imaging Data
- Day 5: In Class Evaluation Exercise

For more information, check out the syllabus.

Tip: To download a repository from GitHub, click on the green button 'Clone or download' then click 'Download Zip'.

5.2.3 Python Bootcamp (Allen Institute for Brain Science)

This Python Bootcamp is from the 2021 Summer Workshop on the Dynamic Brain (SWDB), a two-week, project-based course co-hosted by the Allen Institute for Brain Science and the Computational Neuroscience Program at the University of Washington.

The materials are available on the SWDB_2021 GitHub under the folder "PythonBootcamp", which includes 8 Jupyter Notebooks:

- 00: Introduction
- 01: Basic Python I - Object and Data Structures
- 02: Basic Python II - Control Flow and Functions
- 03: Intro to Scientific Computing

- 04: Introduction to Numpy
- 05: Custom Modules and Version Control
- 06: Introduction to Matplotlib
- 07: Introduction to Pandas
- 08: Development Tools

5.3 Documentation

5.3.1 Python Standard Library

The Python Standard Library contains descriptions of built-in functions, constants, types, exceptions, and modules.

5.3.2 PEP 8

PEP 8 is the official Python style guide. It outlines Python coding conventions to promote readability of code and consistency within and between projects.

Tip: Adopt good coding habits early. To check that your code is compliant with PEP 8 conventions, run *pycodestyle* on your script.

5.3.3 Python Package Index (PyPI)

The Python Package Index (PyPI) is a repository of Python packages.

Important: `pip` is the Python package installer program. To install a package, enter the command `pip install package_name` into Terminal or Command Prompt.

The packages listed here form the basis of most data analysis and processing.

1. IPython

IPython is an interactive Python shell. It is best for exploratory and demonstrative purposes, like quickly testing functions or playing around with new commands.

Documentation: https://ipython.readthedocs.io/en/stable/

Example of use

In a Terminal or Command Prompt window,

```
C:\Users\User>ipython
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)]
Type 'copyright', 'credits' or 'license' for more information
IPython 7.6.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: x = 5 + 2

In [2]: x
Out[2]: 7
```

Tip: Try using IPython to go through the examples in this page.

2. NumPy

NumPy is a library for scientific computation, which includes support for array and matrix operations, as well as mathematical functions.

Documentation: NumPy Manual

Important: To enable the use of NumPy and other packages during a session, it must be *imported*. For ease of use, it is standard for most modules to be imported using an abbreviation. For NumPy, it is typically "np".

```
>>> import numpy as np
```

Example of use

Calculate the average potential of a membrane at rest. Suppose the data was stored in a variable called potential, which is an array of voltages in units of millivolts.

```
>>> type(potential) # check that potential is indeed an array
numpy.ndarray
>>> mean = np.mean(potential)
>>> mean
-70.232
```

Note: Comments in Python begin with #.

3. Matplotlib

Matplotlib is a 2D plotting library. The pyplot API matplotlib.pyplot is a collection of MATLAB-like functions intended for simple plots.

Documentation: User's Guide

Hint:

```
>>> import matplotlib.pyplot as plt
```

4. SciPy

Scipy is a library that contains submodules for integration, interpolation, signal processing, and statistics, among others.

Documentation: https://docs.scipy.org/doc/scipy/reference/

Hint:

```
>>> from scipy import stats
```

5. pandas

pandas is a library that provides tools for the creation and manipulation of data structures, as well as data analysis. It is best for working with tabular data (csv, xlsx) or time series data.

Documentation: https://pandas.pydata.org/pandas-docs/stable/index.html

Example of use

Load in mice.csv and store it in a variable.

```
>>> import pands as pd
>>> mice = pd.read_csv('mice.csv')
>>> mice
  mouse sex     cage
0  M802   M  C3M0009
1  M002   F  C3P0032
2  M194   F  C3M0009
```

Store the IDs of female mice in a variable called `female_ID`.

```
>>> female_ID = mice.mouse[mice.sex == 'F']
>>> female_ID
1    M002
2    M194
Name: mouse, dtype: object
>>> # OR...
>>> female_ID = mice.groupby(['sex']).get_group('F')['mouse']
>>> female_ID
1    M002
2    M194
Name: mouse, dtype: object
```

The first method directly indexes `mice`. The second method groups the mice by sex first, from which it then gets the female group, and finally extracts the mouse IDs by indexing with the `mouse` column.

6. Seaborn

Seaborn is a statistical data visualization library based on matplotlib. It enables easy creation of appealing figures, like violin plots and heat maps.

Documentation: https://seaborn.pydata.org/api.html

Hint:

```
>>> import seaborn as sns
```

Tip: Check out this tutorial from EliteDataScience.com which uses a Pokémon dataset to explore the features of seaborn: The Ultimate Python Seaborn Tutorial: Gotta Catch 'Em All.

7. pycodestyle

pycodestyle is the Python style guide checker. It was formerly known as pep8.

Documentation: https://pycodestyle.readthedocs.io/en/latest/

Example of use

Note the use of the `more` command to see the contents of example.py.

```
$ more example.py
a="Welcome to the Brain Circuits Cluster!"

print(a)

$ python example.py
Welcome to the Brain Circuits Cluster!

$ pycodestyle example.py
example.py:1:2: E225 missing whitespace around operator
```

A space must be added before and after the equal sign. Modify the script and run it through pycodestyle again.

```
$ more example.py
a = "Welcome to the Brain Circuits Cluster!"

print(a)

$ pycodestyle example.py

$
```

All is well!

5.4 Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) that can launch commonly used Python applications, such as Spyder, Jupyter Lab, and Jupyter Notebook.

Installation instructions are available here:

- Windows
- macOS
- Linux

CHAPTER 6

Jupyter

Project Jupyter is an open-source project intended to support interactive scientific computing across numerous languages.

Documentation: https://jupyter.readthedocs.io/en/latest/

Tip: Scared of commitment? To try Jupyter without installation, click here.

Table of Contents

6.1 Jupyter Notebook

The Jupyter Notebook is an open-source web application which provides an interactive environment that smoothly integrates live code, images, equations, data visualization, interactive widgets, and text.

Note: Jupyter is not exclusive to Python - it supports over 40 languages, including R, Java, and MATLAB. IPython is the default kernel of Jupyter Notebook but other kernels can be installed to enable the use of other languages within Jupyter.

Check out the list of available kernels here.

6.1.1 LaTeX

Jupyter Notebooks can include inline and displayed mathematical equations written in LateX.

LaTeX is used to typeset mathematical notation in scientific documents.

- The Jupyter Notebook documentation has a brief section on LaTeX equations.
- For the basics of LaTeX, An Introduction to Using TeX in the Harvard Mathematics Department is a concise and useful reference written by R. Kuhn, R. Scott, and L. Andreev.
- A list of LaTeX Math Symbols prepared by L. Kocbach.

Example of use

In a Markdown cell,

```
The Nernst Equation is as follows

$$E_x = \frac{RT}{zF}ln{\frac{[X]_o}{[X]_i}}$$

where $R$ is the gas constant, $T$ the temperature, $z$ the valence of the ion, $F$ the Faraday
constant, and $[X]_o$ and $[X]_i$ the concentrations of the ion outside and inside the cell.
```

where `$$` and `$` indicate the start and end of a displayed and inline equation respectively.

This produces

The Nernst Equation is as follows

$$E_x = \frac{RT}{zF}ln\frac{[X]_o}{[X]_i}$$

where R is the gas constant, T the temperature, z the valence of the ion, F the Faraday constant, and $[X]_o$ and $[X]_i$ the concentrations of the ion outside and inside the cell.

6.1.2 ipywidgets

ipywidgets are interactive HTML widgets that can be used to build GUIs within Jupyter Notebooks. Available widgets include buttons, sliders, textboxes, and checkboxes.

Hint:

```
$ pip install ipywidgets
$ jupyter nbextension enable --py widgetsnbextension
```

To use:

```
import ipywidgets as widgets
```

6.1.3 Tools

nbviewer

nbviewer renders Jupyter Notebooks in a browser.

nbconvert

nbconvert converts Jupyter Notebooks (.ipynb files) to other formats, including HTML and PDF.

Hint: Installing Jupyter (`pip install jupyter`) also installs nbconvert. To use nbconvert from the command line, enter the following command in the directory in which the notebook is stored.

```
$ jupyter nbconvert --to format notebook.ipynb
```

Replace `format` wtih the desired format and `notebook.ipynb` with the notebook file.

Tip: Saving as different formats is also possible within Jupyter. To see the available formats:

- In Jupyter Notebook,

  click on *File* then hover over *Download as*.

- In Jupyter Lab,

  click on *File* then hover over *Export Notebook As...*.

6.2 Jupyter Lab

Jupyter Lab is the web-based user interface intended to replace Jupyter Notebook. It has all the classic features of its predecessor plus some cool new ones, most notably it offers a flexible and unified workspace that can include a code console, terminal, text editor, and Notebook.

6.3 Syzygy

Syzygy is a service provided by the Pacific Insittute for the Mathematical Sciences (PIMS), Compute Canada, and Cybera that launches Jupyter notebooks in a browser. It is accessed by logging in with a CWL through https://ubc.syzygy.ca/.

Note: Each user is allocated 1GB of space.

CHAPTER 7

Statistics

Statistics is an inevitable part of research but not necessarily an inevitable or significant part of undergraduate education.

Since neuroscientists come from a diverse range of backgrounds, knowledge and understanding of statistics can vary greatly from person to person so here are some resources on statistics in neuroscience flavor.

Table of Contents

7.1 Applied Statistics for Neuroscience (UC Berkeley)

The materials for a UC Berkeley course called Applied Statistics for Neuroscientists are available on GitHub. The course is an intensive introduction to statistics and provides an opportunity to gain a solid understanding of frequently used statistical methods in neuroscience which are not always explained in depth.

The course constitutes of tutorials and labs in Jupyter notebooks, which are organized into three parts:

Part 00: Setup and Review

- 00 - Setup
- 01 - Probability and Statistics Review
  - Lab A - Probability and Python
  - Lab B - Statistics with Pandas and Seaborn

Part 01: Statistical Testing

- 02 - Inferential Statistics and Error Bars
  - Lab A - Inferential Statistics
  - Lab B - Error Bars
- 03 - Hypothesis Testing
  - Tutorial - Hypothesis Testing
  - Lab - Hypothesis Testing
- 04 - Tests for 2-Sample Data
  - Tutorial - Tests for 2-Sample Data
  - Lab A - Unpaired t-tests
  - Lab B - Paired t-tests and Non-Parametric Tests
- 05 - ANOVA I
  - Tutorial - ANOVA by Hand
  - Lab - One-Way ANOVA
- 06 - ANOVA II
  - Tutorial - Two-Way Anove by Hand
  - Lab A - Two-Way ANOVA
  - Lab B - Multiple Comparisons and ANOVA

Part 02: Statistical Modeling

- 07 - Linear Algebra
  - Tutorial - Linear Algebra for Neuroscientists
  - Lab - Linear Transformations
- 08 - Bootstraping and Correlation
  - Tutorial A - Sampling and Bootstrapping
  - Lab A - Visualizing Bootstrapping and One-Sample Tests
  - Tutorial B - Dependence and Correlation
  - Lab B - Computing and Bootstrapping Correlation
- 09 - Model Specification
  - Lab - Visualizing Models
- 10 - Model Fitting
  - Lab -Fitting Models
- 11 - Model Accuracy and Reliability
  - Lab - Model Accuracy and Reliability
- 12 - Classification
  - Lab - Classification
- 13 - Clustering

- Tutorial - Clustering

Since the code used for examples is written in Python, familiarity with the language would be helpful and is therefore recommended.

Attention: To prevent errors while running the notebooks, ensure the latest versions of matplotlib and seaborn are installed on your computer. Enter the command

```
$ pip install --upgrade --user matplotlib
```

into Terminal or Command Prompt.

Error: Before you start Part 02-10: Model Fitting, open utils.py in the util folder and comment out all instances of `ax.set_aspect('equal')`. This command does not apply to 3D plots and will therefore prevent them from rendering.

An equivalent function called `axis_equal_3d` was included instead, which was written by Stack Overflow user Ben as a response to this post.

7.2 Statistics and Data Analysis Tutorial (Center for Brains, Minds and Machines)

The Brains, Minds and Machines (BMM) Summer Course by The Center for Brains, Minds and Machines (CBMM) at MIT is a three-week course on computation, neuroscience, and cognition in the fields of human and machine intelligence research.

The Statistics and Data Analysis tutorial from the 2018 BMM summer course is available on Youtube. It is a quick overview of the basics of statistics by Ethan Meyers.

00:00-10:13 Introduction

10:14-12:22 Box and violin plots

12:23-13:42 Joy plots

13:43-14:52 Dynamite plots

14:53-16:40 What is a statistic?

16:41-18:26 Correlation coefficient

18:27-19:22 Descriptive statistics

19:23-19:48 Population/process parameters

19:49_-20:11 Statistical Inference

20:12-20:24 Estimation

20:25-22:27 Regression

22:28-23:56 Sampling Distribution

23:57-24:29 Estimation (continued)

24:30-27:17 Confidence intervals

27:18-33:17 Bootstrapping, 95% confidence interval

33:18-35:05 What is a p-value?

35:06-40:34 Hypothesis testing

40:35-41:34 Permutation tests, parametric tests

41:35-43:29 Visual hypothesis tests

43:30-53:50 Permutation test example

53:51-58:28 Type I and Type II errors

58:29-1:03:44 Multiple hypothesis tests

1:03:45-end Data Science

7.3 Statistics for Brain and Cognitive Science (MIT)

Statistics for Brain and Cognitive Science is an MIT course that covers three main topics: probability theory, statistical theory, and the linear model. It includes topics such as estimation, Bayesian methods, bootstrap, hypothesis testing, and confidence intervals.

The course materials from fall 2016, including the syllabus, lecture notes, and assignments, are available for download on MIT OpenCourseWare.

CHAPTER 8

Data Management

With big data comes big responsibility.

Table of Contents

8.1 Data Repositories

8.1.1 Scholars Portal Dataverse

Scholars Portal Dataverse is a research data repository that emphasizes discoverability and long-term preservation of data while ensuring academic credit. For more information, see the section on Scholars Portal Dataverse in the UBC Brain Circuits Data Management white paper.

The Brain Circuits Cluster has its own dataverse, within which certain labs have set up their own dataverses.

Attention: The maximum file size for upload on Scholars Portal Dataverse is 2.5 GB.

If a dataverse for your lab already exists, ask your PI for access.

If your lab is interested in creating a dataverse, contact Jeffrey LeDue through email (jledue@mail.ubc.ca).

8.1.2 Federated Research Data Repository (FRDR)

The Federated Research Data Repository (FRDR) is a collaboration between Portage, Compute Canada (CC), and the Canadian Association of Research Libraries (CARL) which provides a system that combines curated large-scale data preservation with data discovery across Canadian research data repositories. Refer to the FRDR section in the UBC Brain Circuit's Data Management white paper for more information.

Datasets in FRDR are searchable by anyone, however FRDR is currently in Limited Production and therefore only accepts data deposits from a select number of research groups. The cluster is pleased to be one of four Special Storage Groups and the first in UBC to deposit a dataset into FRDR.

PIs can deposit datasets into the cluster's FRDR collection by becoming a depositor. Contact Jeffrey LeDue (jledue@mail.ubc.ca) to request access.

Attention: FRDR automaticallly archives data submissions less than 4 TB.

8.2 Version Control

Simply put, version control is a way of tracking changes to a file. It can seem like a lot of work on top of the big helping of work you likely already have so why add it to your plate?

Version control is the solution to many problems which are unfortunately familiar:

- **File recovery**: computer crash, accidental deletion, corrupted file - in case of emergency, break the version control glass.
- **Versioning**: Make experimental changes without relying on the undo button to save you when things goes awry.
- **Backup**: Lost your copy of the file? Working from another computer? No problem.
- **Attribution**: Credit is given where credit is due. A record of user contributions simplifies collaboration.
- **Changelog**: Have you ever looked at previous work and wondered, *Why did I do that?*? Version control keeps a record of the changes you've made and why you've made them so your past self doesn't confuse your future self.

Note: A **remote** repository is hosted on a server (online) while a **local** repository exists on a computer (offline). When referring to the same repository, the local *repo* is usually a clone of the remote which typically lives on GitHub.

8.2.1 Git

Git is a popular open-source distributed version control system.

GitHub hosts Git repositories. Check out the cluster's GitHub Team repository. Each lab can be added as a Team. If interested, please contact Jeffrey LeDue (jledue@mail.ubc.ca).

Tip: Learning Git can be a tough task. Here are some resources to help you get started:

- Git Handbook, a 10-minute read on the basics of Git and its integration with GitHub
- An introduction to Git: what it is, and how to use it, a tutorial on Git commands by freeCodeCamp

CHAPTER 9

Image Analysis

There are two main image analysis software programs in neuroscience, namely ImageJ and Cell Profiler, both of which are free, open-source, and have a large community of users.

Table of Contents

9.1 ImageJ

ImageJ is a Java image processing and analysis program.

- Wiki
- User Guide

9.1.1 Fiji

Fiji is an ImageJ distribution with built-in plugins. It is the recommended way to use ImageJ.

Note: ImageJ commands can be integrated into Python code. To see how, check out the materials from the ImageJ Macro Workshop hosted by NINC during the 2019 MSC-SMC Annual Meeting at UBC.

9.2 Cell Profiler

Cell Profiler is a cell image analysis software that enables users to construct pipelines for cell identification and measurement of phenotypes including morphology, intensity, texture, and size.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search