BioPype: A crash course in bioinformatics and custom pipelines

Ethan Gniot

May 2018

Todo list

Contents

1 Foreword

1.1 Goal

This tutorial aims to improve your general understanding of bioinformatics through several methods:

- Define technical terms commonly used in bioinformatics methods and found in the literature.
- Provide a collection of various useful resources, including...
    1. Resources for finding tools, data, and background information that can help answer your research questions.
    2. Resources that explain details about bioinformatics concepts and techniques in beginner-friendly language.
- Demonstrate how Python can be used to answer your research questions by combining existing bioinformatics tools and automating repetitive or time-consuming tasks.

1.2 How to use this book

Most reference materials are consolidated in the appendices at the end of the book. The main text of this book is written in the right-hand margin. The left-hand margin contains special markers and important notes to the reader. The book can be used as a self-paced tutorial with the help of the markers described below.

This is a margin label. I will write things here to further explain the main text, define jargon, etc.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas eu felis sodales, interdum purus nec, interdum ex. Integer at nunc ultricies, tempus nibh eget, egestas risus. Suspendisse aliquam, lorem at dictum accumsan, dolor elit euismod velit, a gravida risus libero non felis. Donec a tortor tempor, scelerisque sapien posuere, volutpat erat. Morbi in imperdiet velit. Vivamus sagittis, massa sit amet venenatis euismod, elit eros aliquam diam, molestie faucibus lectus nisi euismod erat. Nulla id dictum mi.

! → **The arrow that points to this line is an "Attention" marker. It will indicate key pieces of information that you should pay special attention to.** *Curabitur egestas aliquam nisl, pharetra finibus mauris placerat nec. Nam in diam risus. Nulla mollis purus quis feugiat tristique. Vestibulum et sollicitudin ante, at sagittis ipsum. Nunc hendrerit ante sed massa semper eleifend.*

→ This kind of annotation will reference appendix entries that you can consult for more-detailed information about the main text.

Ut ultrices eros velit, at faucibus ante rutrum eget. Pellentesque a molestie diam. Curabitur mattis dui a risus lacinia fringilla. Phasellus porttitor elit nec neque euismod, id ultrices elit lobortis. Aliquam molestie sem. Curabitur sit amet urna faucibus, vulputate arcu sit amet, fermentum ipsum. Nulla facilisi. Nam ullamcorper eget leo id malesuada.

1.3 Pre-requisites

Things to know

Computer requirements

- At least 4GB RAM bare minimum, 16GB RAM if possible (basically, the more RAM, the better)
- Must be able to run macOS 10.12 Sierra, preferably macOS 10.13 High Sierra
- At least 500GB storage (ideally several TB)

2 Source Code

(Source code can be found at github.com/EthanGniot/LU-microbiome)

3 Sample Information

Still need to add information about the inflammatory bowel disease dataset

While we are building the LU-microbiome pipeline, we will need a dataset that we can use to test our pipeline throughout the process and make sure it is working as intended. In order to do this, we must use a dataset that's already been analyzed so that we know what the results should look like. When we test our pipeline, if our results match the results of the original analysis, then we will know that our tool is working correctly.

There are several test datasets available to us for testing the feature that analyzes the relative abundance of bacteria in the microbiota.

Make URLs footnotes instead of appendix entries

→ Caporaso et al, 2011 [?]

The first is the dataset used in both the QIIME "Illumina Overview Tutorial" (A.1) and the QIIME 2 "Moving Pictures" tutorial (A.2) derived from the *Moving Pictures of the Human Microbiome* study, where two human subjects collected daily samples from four body sites: the tongue, the palm of the left hand, the palm of the right hand, and the gut (via fecal samples obtained by swapping used toilet paper). These data were sequenced using the barcoded amplicon sequencing protocol described in *Global patterns of 16S rRNA diversity at a depth of millions of sequences per sample.* A more recent version of this protocol that can be used with the Illumina HiSeq 2000 and MiSeq can be found here.

→ THIS CITATION NEEDS TO BE FIXED [?]

Fix bibliography entries in this paragraph and in general. They're not correctly referencing even though bib file has entries.

(Here is information about the untested inflammatory bowel disease dataset that we will analyze using the completed pipeline)

4 Setting the scene

("Here" is a hypothetical situation/research question that a student may have. This is the research question that will be answered by the pipeline we are creating.)

5 Microbiome Analysis

(This chapter will give some general background information about the topics listed below. More-detailed information will be provided in later chapters when we are actually creating the pipeline)

5.1 The Gut Microbiome

5.2 Relative Abundance Analysis

5.3 Metagenomics

5.4 Python

6 How to Find Tools

6.1 Finding Data

(Here is where we talk about various databases that users can use to find general information, data files, study results, public datasets, etc.)

6.2 Finding Software

(Here is where we talk about ways/places that people can look for software programs that can help answer their research question.)

7 The Plan

(Break down the sub-tasks required to accomplish the two main tasks: Relative abundance analysis and metagenomic analysis)

1. What is my research question?
2. Is there an existing tool that I can use to directly answer my research question? If not, proceed to Step 3.
3. What is the step-by-step process required to answer my research question?
4. What existing tools are available that can help me accomplish each of these steps?
5. How do I write code that uses these tools to accomplish the steps?

8 Software and Set-up

8.1 Software

Add further software that you end up using (e.g., USEARCH). ALSO, make sure to update the section that mentions installing packages if you do so.

(Table of software name, name in PATH, version number, function of the software for each one we're gonna use)

add 3rd column to table with link to documentation/download source

Software name	*Version number*
Anaconda	5.1
Biopython	1.70
BLAT	35
matplotlib	2.2.2
pandas	0.22.0
sra-tools	2.8.2
trim-galore	0.4.5

Table 8.1: **Software used to create the tutorial pipeline.**

8.2 Set-up and Install Dependencies

Before we write any code, there are several steps that must be completed to prep your machine for the tasks we will be performing in this tutorial. Without these prerequisites, the code you write during this tutorial will not work correctly:

1. Install and open Anaconda
2. Create a new virtual environment
3. Install packages

8.2.1 Install Anaconda

→ RESOURCE FOR LEARNING ABOUT PYTHON PACKAGES

The Anaconda program will play a key role in this tutorial. Anaconda is essentially Python and a lot of scientific computing tools bundled together, along with many popular add-ons to Python called packages. Downloading all of these tools individually can be difficult, as the quirks of one package may conflict with another when they're installed manually; using Anaconda to install packages greatly simplifies the process because Anaconda can smoothly handle all of the minute details that cause manual installations to fail.

Install and Open:

1. Go to the download page for the Anaconda distribution at `https://www.anaconda.com/download`.

2. Select your preferred operating system from the Windows, macOS, or Linux tabs, then select the Download option for the **Python 3.6 version** (Figure 8.1) and follow the installation instructions.

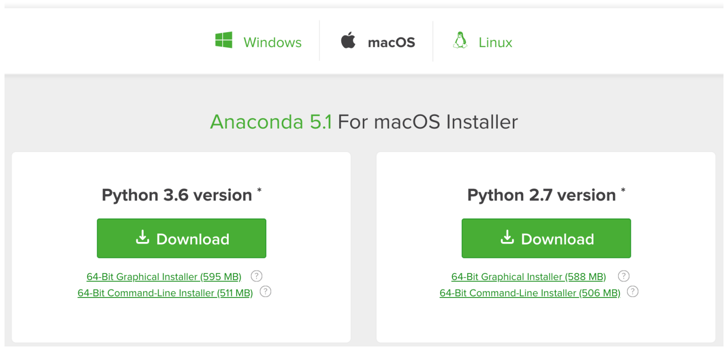
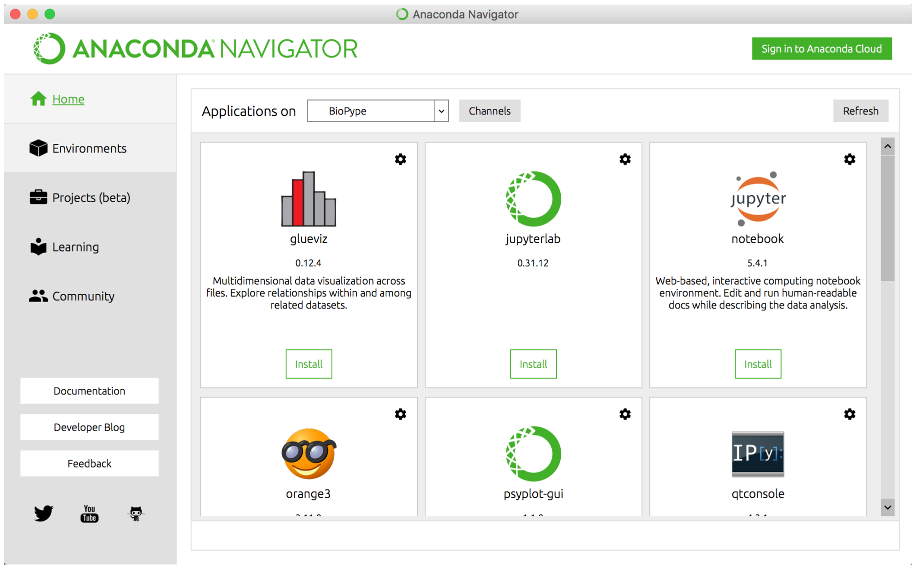

Windows
macOS
Linux
Anaconda 5.1 For macOS Installer
Python 3.6 version *
Download
64-Bit Graphical Installer (595 MB)
64-Bit Command-Line Installer (511 MB)
Python 2.7 version *
Download
64-Bit Graphical Installer (588 MB)
64-Bit Command-Line Installer (506 MB)

Figure 8.1: The Anaconda download options provided on the Anaconda distribution website at `https://www.anaconda.com/download`

.

3. After installation is complete, open the application named "Anaconda-Navigator" (the icon looks like). After a brief start-up period, you should see the following window (Figure 8.2):



Multidimensional data visualization across files. Explore relationships within and among related datasets.

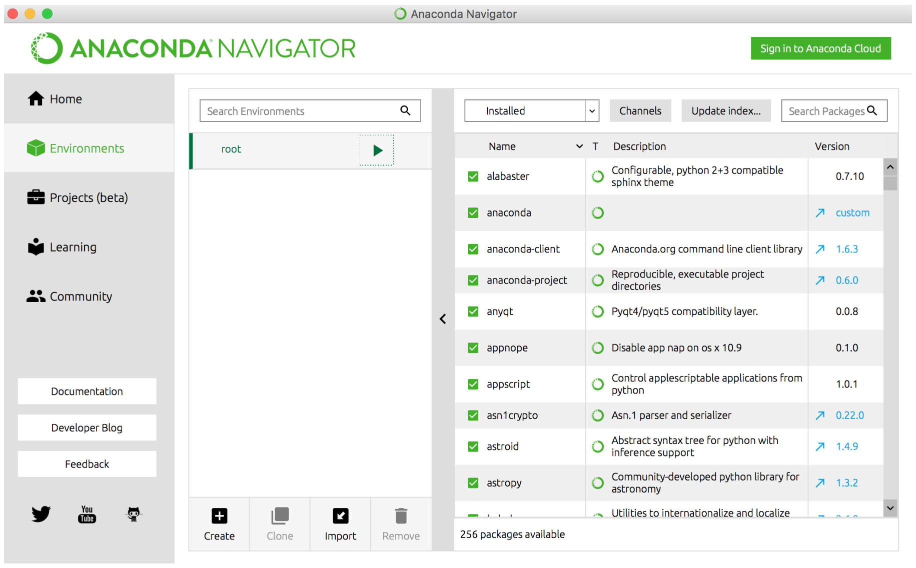
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Figure 8.2: The window displayed to the user upon opening Anaconda-Navigator.

8.2.2 Create a New Virtual Environment

Write a blurb explaining the benefits of using a virtual environment.

Link to resource for further reading on virtual environments



Name	T	Description	Version
✓ alabaster	○	Configurable, python 2+3 compatible sphinx theme	0.7.10
✓ anaconda	○		↗ custom
✓ anaconda-client	○	Anaconda.org command line client library	↗ 1.6.3
✓ anaconda-project	○	Reproducible, executable project directories	↗ 0.6.0
✓ anyqt	○	Pyqt4/pyqt5 compatibility layer.	0.0.8
✓ appnope	○	Disable app nap on os x 10.9	0.1.0
✓ appscript	○	Control applescriptable applications from python	1.0.1
✓ asn1crypto	○	Asn.1 parser and serializer	↗ 0.22.0
✓ astroid	○	Abstract syntax tree for python with inference support	↗ 1.4.9
✓ astropy	○	Community-developed python library for astronomy	↗ 1.3.2
		Utilities to internationalize and localize	

Figure 8.3: The Environments window of the Anaconda-Navigator.

Make sure the computer has an internet connection while completing this section, otherwise Anaconda will not let you create a virtual environment.

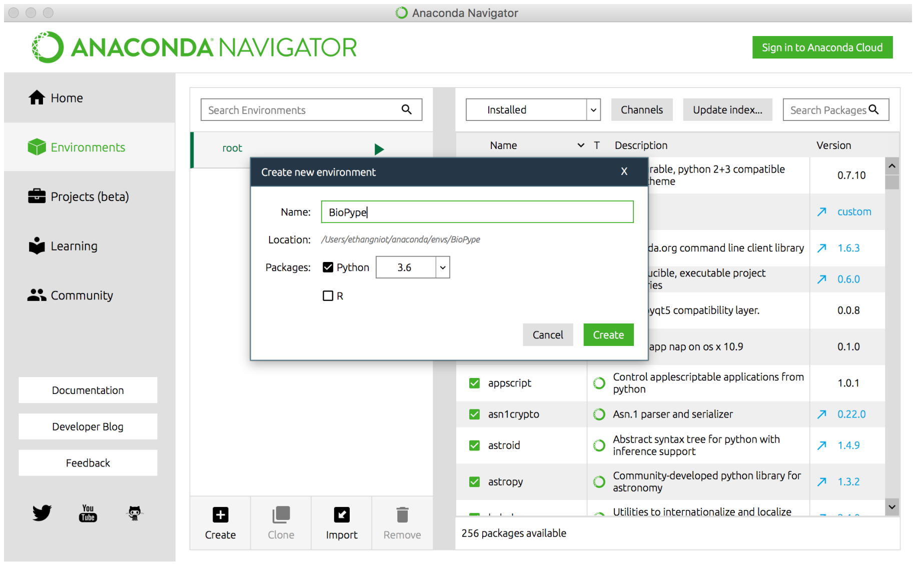
1. On the left side of the Anaconda-Navigator window, click on the tab labeled **Environments**. (Figure 8.3)
2. Click the **Create** button on the bottom of the center panel. A new window titled "Create new environment" will appear. (Figure 8.4)
3. Enter a **Name** for the environment. You may choose any name you want, but for the sake of this tutorial we will name the new environment "BioPype".
4. Select the box labeled **Python** next to the **Packages** heading.
5. Choose the latest version of Python from the adjacent drop-down menu (Python 3.6 is the most current version at the time of this writing, so we choose **3.6**).
6. Click the **Create** button within the "Create new environment window".

8.2.3 Install packages

Write blurb about what packages are.

Link to resource for further reading about packages.

1. Change Anaconda's current environment from the **root** environment by selecting the **BioPype** tab in the middle panel of the Environments window.
2. Click on the drop-down menu in the right-hand panel that says "Installed" and change it to "All".



Name: BioPype
Location: /Users/ethangniot/anaconda/envs/BioPype
Packages: ☑ Python 3.6
☐ R

	Name	T	Description	Version
			rable, python 2+3 compatible heme	0.7.10
				custom
			da.org command line client library	1.6.3
			ucible, executable project ies	0.6.0
			yqt5 compatibility layer.	0.0.8
			app nap on os x 10.9	0.1.0
☑	appscript		Control applescriptable applications from python	1.0.1
☑	asn1crypto		Asn.1 parser and serializer	0.22.0
☑	astroid		Abstract syntax tree for python with inference support	1.4.9
☑	astropy		Community-developed python library for astronomy	1.3.2
			Utilities to internationalize and localize	

256 packages available

Figure 8.4: The "Create new environment" window.

3. In the "Search Packages" box, enter "biopython". The search should return a package named "biopython". Select the checkbox to the left of the name. (Figure 8.5)
   - A pair of green and red boxes (reading "Apply" and "Clear", respectively) will appear in the bottom-right of the window once the package is selected. Do not click these just yet.

Solve the issue with failing to center figures when they're on their own page

4. Use the search bar to find and select the other packages listed in Table 8.1. Once all packages have been selected, click the green "Apply" button in the bottom right corner of the window, then select "Apply" again within the "Install Packages" window that appears. (Figure 8.6) Anaconda will now install the selected packages.

5. Talk about setting up the sra-tools workspace (https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=toolkit_doc&f=std)

6. PYPIPER IS ONLY COMPATIBLE WITH MAC AND LINUX. Start by coding with just subprocess module commands. If the scripts work on Windows computers, then forget about using pypiper. But if the subprocess scripts don't work on windows, then we'll be developing exclusively for Mac anyways, so you could use pypiper without any worries. In that case, talk about installing pypiper here. OTHERWISE, delete this section.

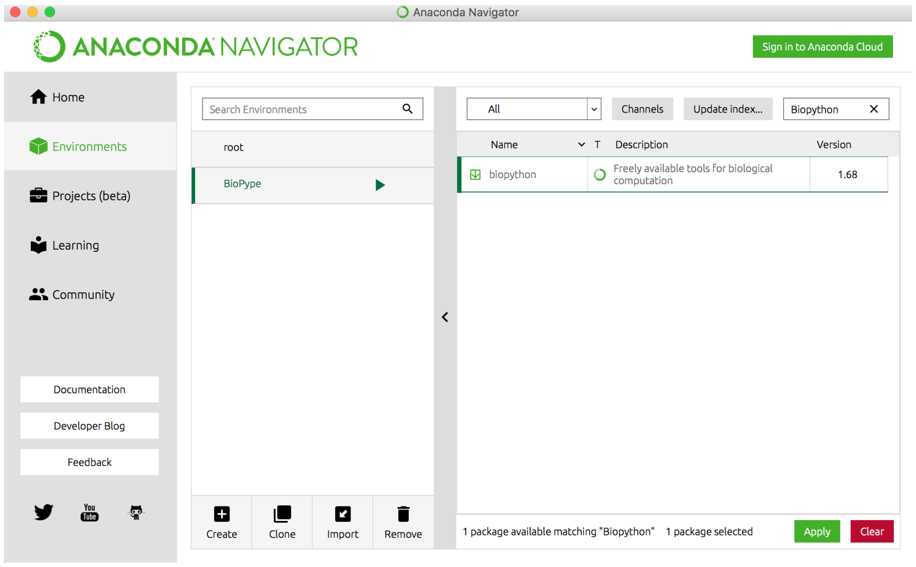
   (a) Open a terminal window in the BioPype environment by clicking the "play" button on the BioPype environment tab and then selecting "Open Terminal".

   (b) Wait for the terminal window to finish opening. You'll know it's finished when you see

   show image of what the prompt looks like when it's done initializing.

   (c) Install **pypiper** by typing the following at the command prompt, followed by pressing return/enter:

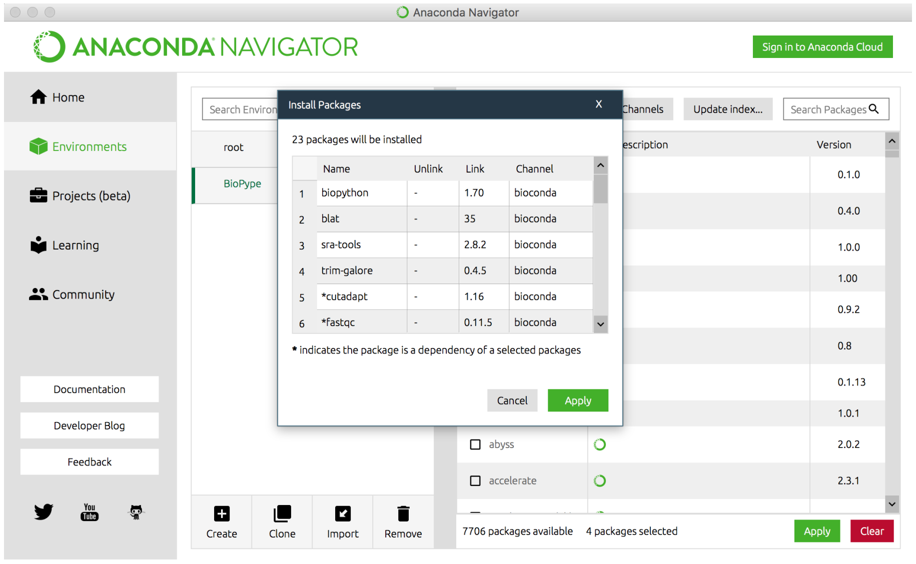
```
pip install --user pypiper
```



Name	T	Description	Version
biopython		Freely available tools for biological computation	1.68

1 package available matching "Biopython" 1 package selected

Figure 8.5: Searching for a package. When a package is selected, the check-box next to the package's name will be green.



Install Packages

23 packages will be installed

	Name	Unlink	Link	Channel
1	biopython	-	1.70	bioconda
2	blat	-	35	bioconda
3	sra-tools	-	2.8.2	bioconda
4	trim-galore	-	0.4.5	bioconda
5	*cutadapt	-	1.16	bioconda
6	*fastqc	-	0.11.5	bioconda

* indicates the package is a dependency of a selected packages

Cancel Apply

7706 packages available 4 packages selected

Figure 8.6: The window displaying the packages and dependencies that will be installed.

(d) Check that the package was installed correctly by executing the following in the command prompt:

```
conda list
```

This will generate a list of all the packages installed in the current environment. If you see the **pypiper** package listed, the installation was successful and you may skip the rest of this section. If not, proceed with the following steps.

(e) Execute the install command from step (c) again. This time, the Terminal should return a message similar to the one displayed in

Reference the figure pypiper-already-installed

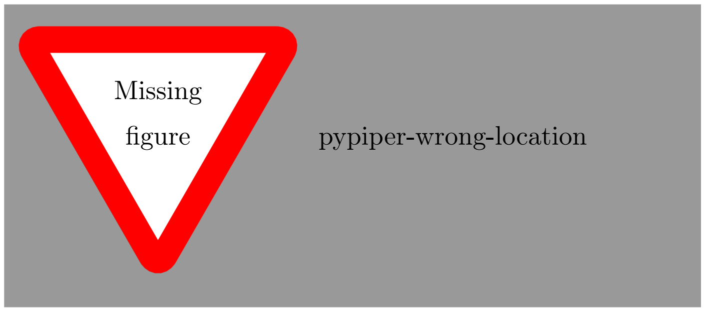
. The line that reads "Requirement already satisfied: pypiper in ...." tells us the location where the package was (incorrectly) installed.

(f) Open a Terminal window, and navigate to the location indicated by the message from the previous step. For my example, I need to start at my home directory and walk through the following folders: .local | lib | python3.6 | site-packages.

- The folders along the path to the pypiper installation may be hidden. On a Mac, these hidden folders are preceded by a "." If the path to the pypiper installation includes hidden locations, reveal them by pressing "Cmd + Shift + ." in the Finder window.
- Once you find the site-packages folder containing two pypiper folders

  reference figure pypiper-wrong-location

  , copy those folders and their contents and paste them into the /anaconda/envs/BioPype/lib/python3.6/site-packages directory. The package should now be installed correctly.



8.2.4 Integrated Development Environment (IDE)

Talk about choosing between PyCharm and other options

8.2.5 PATH

(Talk about putting all the tools in the same path/directory)

8.3 Analysis Pipeline

(Use figures to illustrate the stages of the pipeline)

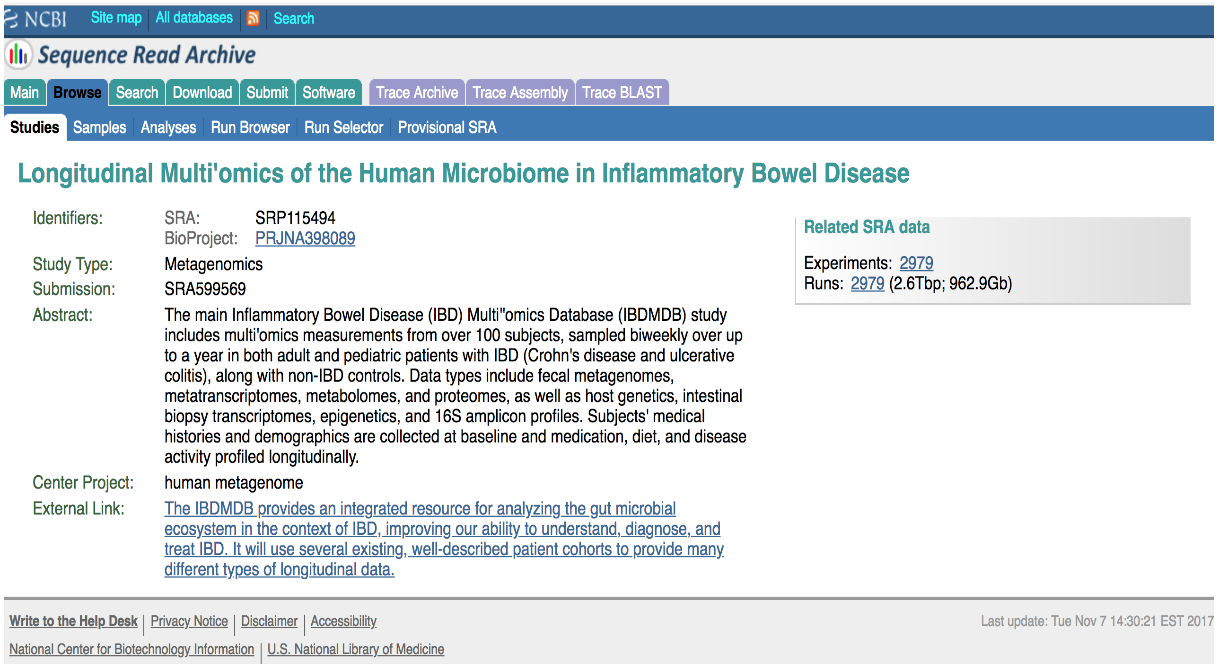
9 The Dataset

Recap

In the previous chapter, we set up our machine so that it has all of the software BioPype needs in order to function.

In this chapter, we will use BioPype to **download** experimental data, and then prepare them for analysis via a process called **"quality control"**. We will also discuss how the BioPype functions used in this workflow were created.

As described in Chapter 4, we want to investigate if there are any differences in the gut microbiomes of younger patients with IBD compared to older patients with IBD. Using the methods described in Chapter 6, we found a study in the SRA database with sequencing data that are useful to us. The webpage for the SRA Study is reproduced in Figure 9.1 and can be accessed at `https://trace.ncbi.nlm.nih.gov/Traces/sra/?study=SRP115494` .



Longitudinal Multi'omics of the Human Microbiome in Inflammatory Bowel Disease

Identifiers:	SRA: SRP115494 BioProject: PRJNA398089
Study Type:	Metagenomics
Submission:	SRA599569
Abstract:	The main Inflammatory Bowel Disease (IBD) Multi''omics Database (IBDMDB) study includes multi'omics measurements from over 100 subjects, sampled biweekly over up to a year in both adult and pediatric patients with IBD (Crohn's disease and ulcerative colitis), along with non-IBD controls. Data types include fecal metagenomes, metatranscriptomes, metabolomes, and proteomes, as well as host genetics, intestinal biopsy transcriptomes, epigenetics, and 16S amplicon profiles. Subjects' medical histories and demographics are collected at baseline and medication, diet, and disease activity profiled longitudinally.
Center Project:	human metagenome
External Link:	The IBDMDB provides an integrated resource for analyzing the gut microbial ecosystem in the context of IBD, improving our ability to understand, diagnose, and treat IBD. It will use several existing, well-described patient cohorts to provide many different types of longitudinal data.

Related SRA data

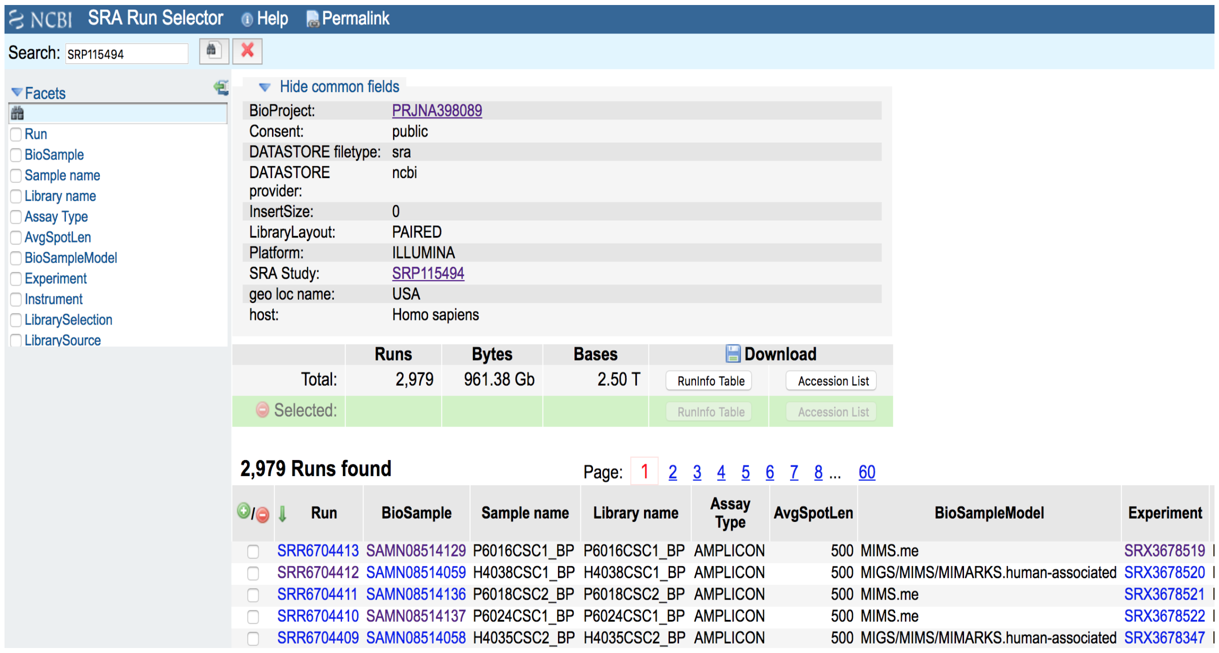
Experiments: 2979
Runs: 2979 (2.6Tbp; 962.9Gb)

Write to the Help Desk | Privacy Notice | Disclaimer | Accessibility
National Center for Biotechnology Information | U.S. National Library of Medicine

Last update: Tue Nov 7 14:30:21 EST 2017

Figure 9.1: The "SRA Study" webpage.

The dataset is reproduced in Figure 9.2 and can be found by clicking on "Runs" under "Related SRA data" on the right side of the SRA Study webpage, or by going to `https://trace.ncbi.nlm.nih.gov/Traces/study/?acc=SRP115494#` .



NCBI SRA Run Selector · Help · Permalink

Search: SRP115494

Facets: Run, BioSample, Sample name, Library name, Assay Type, AvgSpotLen, BioSampleModel, Experiment, Instrument, LibrarySelection, LibrarySource

Hide common fields

BioProject:	PRJNA398089
Consent:	public
DATASTORE filetype:	sra
DATASTORE provider:	ncbi
InsertSize:	0
LibraryLayout:	PAIRED
Platform:	ILLUMINA
SRA Study:	SRP115494
geo loc name:	USA
host:	Homo sapiens

	Runs	Bytes	Bases	Download	
Total:	2,979	961.38 Gb	2.50 T	RunInfo Table	Accession List
Selected:				RunInfo Table	Accession List

2,979 Runs found Page: 1 2 3 4 5 6 7 8 ... 60

Run	BioSample	Sample name	Library name	Assay Type	AvgSpotLen	BioSampleModel	Experiment
SRR6704413	SAMN08514129	P6016CSC1_BP	P6016CSC1_BP	AMPLICON	500	MIMS.me	SRX3678519
SRR6704412	SAMN08514059	H4038CSC1_BP	H4038CSC1_BP	AMPLICON	500	MIGS/MIMS/MIMARKS.human-associated	SRX3678520
SRR6704411	SAMN08514136	P6018CSC2_BP	P6018CSC2_BP	AMPLICON	500	MIMS.me	SRX3678521
SRR6704410	SAMN08514137	P6024CSC1_BP	P6024CSC1_BP	AMPLICON	500	MIMS.me	SRX3678522
SRR6704409	SAMN08514058	H4035CSC2_BP	H4035CSC2_BP	AMPLICON	500	MIGS/MIMS/MIMARKS.human-associated	SRX3678347

Figure 9.2: The sequencing data collected by the *Longitudinal Multi'omics of the Human Microbiome in Inflammatory Bowel Disease* study. (Note: the table's columns extend beyond the right side of the page because there are 36 metadata categories.)

To learn more about how to use these pages to find information about the study and its methods, please refer to Chapter 6.

9.1 Download Dataset

*****Show the user what the result of the commands are. Right now we show the commands, but not the results. e.g., when we run a command to get a list of accession numbers, follow it up by typing the command that just prints the list of numbers in the terminal. Then the user can see what their results should look like if they did everything correctly.

1) Choose sample criteria

Note: The dataset has a metadata category called "BioSample." This is not to be confused with the generic term "samples", which in this case are equivalent to one run (i.e., one row) of the data table.

1. First, we must decide what criteria we will use to choose our experimental samples.
   Remember: our research question investigates the effects of age on the gut microbiomes of inflammatory bowel disease patients. Looking at the study abstract from Figure 9.1, we can see that the researchers investigated two types of IBD: Crohn's disease and ulcerative colitis. So one of the metadata categories from the dataset (i.e., the columns from the dataset in Figure 9.2) that we will use to select our samples

is the "host_disease" category. We will also select our samples based on the "host_age" metadata.

2. Now we want to select our samples based on these categories.

   Download the RunInfo Table file found on the study's sample page. The RunInfo table is the browser's sample table in a .txt file format.

   Missing figure

   highlight where to click to download the runinfo table

3. Open the command line interface for your machine's operating system. On a mac, this is the Terminal application.

4. Navigate to your project's folder and then confirm that you are in the right location by using the the following commands at the command prompt:

```
$ cd <path\_to\_your\_project>
$ pwd
```

5. Activate the Python interpreter by typing the following into the command prompt:

```
$ python3
```

6. Import BioPype to the current Python environment:

```
>>> import BioPype
```

! →

   - If not done already, stop and follow the instructions in Chapter 8 to configure the BioPype working directory and the SRA Toolkit Workspace Location.

7. Import `BioPype.cmds.runtable` to gain access to the `RunTable` class:

   Either shorten this margin-label, or put it in a footnote.

   Using "as" in an import statement lets you reference the full name of the imported package by some shorter name. It is generally considered better practice to **not** import packages "as" some other name, because it makes the code less explicit (which goes against the purpose of Python), but we do it in this manual for the sake of space.

```
>>> import BioPype.cmds.runtable as runtable
```

8. Create a `RunTable` object called "my_table" out of the RunInfo Table .txt file:

   Give a specific name to the RunInfo Table file. People might get confused if they needs to fill in the name of the file themselves.

```
>>> my_table = runtable.RunTable('runinfotable_filename.txt')
```

- With the `my_table` object created, you can remind yourself what the metadata categories are by using the following command:

```
>>> my_table.column_names
['Assay_Type', 'AvgSpotLen', 'BioSample', 'BioSampleModel',
 'Experiment', 'Instrument', 'LibrarySelection', 'LibrarySource',
 'Library_Name', 'LoadDate', 'MBases', 'MBytes', 'Organism',
 'ReleaseDate', 'Run', 'SRA_Sample', 'Sample_Name',
 'collection_date', 'env_biome', 'env_feature', 'env_material',
 'host_age', 'host_disease', 'host_sex', 'host_subject_id',
 'lat_lon', 'BioProject', 'Consent', 'DATASTORE_filetype',
 'DATASTORE_provider', 'InsertSize', 'LibraryLayout', 'Platform',
 'SRA_Study', 'geo_loc_name', 'host']
```

9. We want to first group the samples based on the patients' disease...

```
>>>uc_samples=my_table.filter_data("host_disease", '==', 'ulcerative colitis')
>>>cd_samples=my_table.filter_data("host_disease", '==', "Crohn''s disease")
```

! →

- Note that the "Crohn's disease" string within the parentheses in the second line of code has two single-quotes/apostrophes between the "n" and the "s" of "Crohn's". This is because the value we input must exactly match the value that the original study's researchers input for the `host_disease` category, and (for whatever reason) they entered the value for Crohn's disease as: `"Crohn''s disease"`

  This is an example of why it's important to check the metadata carefully before you begin the analysis. If the argument does not exactly match the value of the metadata category, it won't select the sample.

10. ... then we want to group them based on age:

```
>>>uc_young = uc_samples.filter_data("host_age", '<=', 21)
>>>uc_old = uc_samples.filter_data("host_age", '>=', 60)
>>>cd_young = cd_samples.filter_data("host_age", '<=', 21)
>>>cd_old = cd_samples.filter_data("host_age", '>=', 60)
```

! →

- Note that the third argument passed to the `filter_data()` method is **not** a string like the previous two arguments. It is simply an integer (no quotation marks around it).

11. Now that we have groups of samples selected, we want to get a list of SRA accession numbers for each group. The accession numbers are what we will use to specify the files we want to download from the SRA database.

```
>>>uc_young_nums = uc_young.get_accession_numbers()
>>>uc_old_nums = uc_old.get_accession_numbers()
>>>cd_young_nums = cd_young.get_accession_numbers()
>>>cd_old_nums = cd_old.get_accession_numbers()
```

→ To learn more about .sra file format, see Appendix A.7.

12. We can now use the lists of accession numbers to download .sra files from the SRA database. However, these are large files that can *easily* take up hundreds of gigabytes of memory. They also take time to download (times will vary based on network speed). To account for hardware limitations that would severely bottleneck the analysis at

file-handling steps like these, we will randomly select a subset of only 3 accession numbers from each list we created in the previous step. The analysis will go much faster if we analyze fewer samples, so these subsets are what we will analyze going forward.

→ The "random_sample_ subset()" method is preceded by "RunTable." in this case because it is a "@staticmethod". To learn more, see Appendix A.3.

```
>>>rs_uc_young = RunTable.random_sample_subset(uc_young_nums, n=3)
>>>rs_uc_old = RunTable.random_sample_subset(uc_old_nums, n=3)
>>>rs_cd_young = RunTable.random_sample_subset(cd_young_nums, n=3)
>>>rs_cd_old = RunTable.random_sample_subset(cd_old_nums, n=3)
```

13. Now we need to download the .sra files linked to these accession numbers and get them into .fastq format. First, we'll need to import the necessary functions:

```
>>>from BioPype.cmds.downloadsra import download_sra,
>>>from BioPype.cmds.downloadsra import convert_sra_to_fastq
```

14. Use the `download_sra()` function to download the .sra files:

```
>>>download_sra(rs_uc_young)
>>>download_sra(rs_uc_old)
>>>download_sra(rs_cd_young)
>>>download_sra(rs_cd_old)
```

! →

- The .sra files will not be downloaded to the correct location if the SRA Toolkit Workspace Location has not been properly configured (see Chapter 8 for details).

→ To learn more about .fastq format, see Appendix A.4.

15. Use the `convert_sra_to_fastq()` function to convert the .sra files to .fastq format.

→ To learn more about what threads are, and how many you should use, see Appendix A.5.

```
>>>convert_sra_to_fastq(threads=4)
```

! →

- BioPype will not know where to look for the downloaded .sra files if the SRA Toolkit Workspace Location has not been properly configured (see Chapter 8 for details).
- The more threads your computer can divide this process between, the better (it's more complicated than that, but for now let's go with it). The question is: how many threads is your computer capable of using? For that, you'll have to do some Googling. You can find the resources I used to figure out how many threads my MacBook Pro can run (4) in Appendix A.6. To simplify: my computer has 1 CPU, the CPU has 2 cores, and each core can run 2 threads. 1 x 2 x 2 = 4 threads.

9.1.1 Creating the code

9.2 Perform Quality Control on Dataset

10 Relative Abundance Analysis

(How to compare relative abundance of bacterial taxa between experimental conditions using QWRAP.)

11 Predict ORFs

(how to predict the ORFs of the sequencing reads)

12 Create Non-redundant Gene Sets

(How to create non-redundant gene sets using the predicted ORFs and what kind of information they provide) (Align reads using BLAT)

13 Align Genes

14 Get GenBank Accession Numbers

15 Find COG Functional Classes

16 New Analysis

(Walk user through analysis of new, untested dataset looking for age-related differences in patients with Inflammatory Bowel Disease.)

Appendix A Web Links

A.1 QIIME Illumina Overview Tutorial

`http://nbviewer.jupyter.org/github/biocore/qiime/blob/1.9.1/examples/ipynb/illumina_overview_tutorial.ipynb`

A.2 QIIME 2 Moving Pictures Tutorial

`https://docs.qiime2.org/2018.2/tutorials/moving-pictures/`

A.3 Explanation of @staticmethod decorator vs @classmethod decorator in Python

The Basics: Static methods make code easier to read and let you use a class' methods without needing to have an object of that class first. This is useful when it makes logical sense to place a function within a class (because the function is related to the other tasks that the class handles), but the function doesn't *need* to operate on an object/the data of that class. Normal methods are called by typing: `my_object.method`. Static methods are called by typing: `MyClass().method` or `MyClass.method`.

Basic explanation with slight background on class methods:
`https://julien.danjou.info/guide-python-static-class-abstract-methods/`

More technical explanations:
`https://stackoverflow.com/questions/12179271/meaning-of-classmethod-and-staticmethod-for-beginner`
This Stack Overflow question has some basic, easy to understand explanations mixed in with more in-depth explanations. The top-voted answer isn't the only one that is useful; each of the answers presents their explanation with a different degree of simplicity. Make sure to check several answers if the top ones don't seem helpful.

A.4 FASTQ Format

`https://galaxyproject.org/tutorials/ngs/`

This link contains:

1. An explanation of the FASTQ file format
2. An explanation of PHRED quality scores with an accompanying figure (Fig 4)
3. The following quote: "Fastq format is not strictly defined and its variations will always cause headache for you. See `https://www.ncbi.nlm.nih.gov/books/NBK242622/` for more information."
    - From the NCBI link: "Text formats, such as FASTQ, are supported, but are not the preferred submission medium. Poorly defined specifications and high variability within these formats tend to lead to a higher frequency of failed or problematic submissions."

A.5 How many threads to use?

`https://www.jstorimer.com/blogs/workingwithcode/7970125-how-many-threads-is-too-many`

A.6 How many threads can my computer run?

`https://superuser.com/questions/1101311/how-many-cores-does-my-mac-have`

A.7 SRA/.sra file format

Find resource for explaining the .sra file format that we download from the SRA Database

Appendix B Referenced Studies