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

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. Most reference materials are consolidated in the appendices at the end of the book. 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 Source Code

The source code for this manual, the accompanying tutorial, and the BioPype package can all be found at `github.com/EthanGniot/BioPype`.

1.4 Pre-requisites

Things to know

Talk about the prior biology knowledge people should have

Computer requirements

- At least 4GB RAM minimum.* 16GB RAM if possible (the more RAM, the better)
  - *At 4GB RAM, one step of the BioPype analysis takes $\tilde{2}$.5 hours to finish when analyzing 20 samples. With less RAM, expect the process to take longer. Conversely, more RAM will increase the speed of the analysis.
- BioPype was developed and tested using a MacBook Pro (Retina, 13-inch, Late 2013) running macOS 10.13.4 High Sierra. BioPype should be compatible with similar macOS versions, as well as Linux, though it has not yet been tested on Linux.
- Storage requirements will vary depending on the amount of data you intend to analyze. The analysis performed in this tutorial required at least 500GB and were satisfied by using a 2TB external hard drive.

2 Sample Information

Still need to add information about the inflammatory bowel disease dataset

While we are building the BioPype 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

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) Caporaso et al. (2011a). 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* Caporaso et al. (2011b). A more recent version of this protocol that can be used with the Illumina HiSeq 2000 and MiSeq can be found here.

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

3 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.)

4 Microbiome Analysis

The analysis used in this tutorial will focus on analysis of the human gut microbiome, but BioPype can be used to analyze the microbiomes of other communities as well.

4.1 The Gut Microbiome

→ For more explanation about the **gut microbiota** and **gut microbiome**, see A.3.

The human gut is home to trillions of microbes. The collection of physical micro-organisms that live in the gut is referred to as the **gut microbiota**. The **gut microbiome** refers to all of the genes contained within these micro-organisms, as well as their functions and interactions.

Human microbiota can play host to many different kinds of microbes, including bacteria, archaea, and fungi. For the sake of this tutorial, we are going to focus on the role of bacteria in human microbiomes.

Gut dysbiosis: deviation from "normal" gut microbiota composition.

→ For an approachable overview of the human microbiome, the effects of microbiome variation on human health and disease, and important research questions regarding the microbiome, see Cho and Blaser (2012).

Sometimes the biodiversity of the gut is thrown out of balance. When the composition of the microbiota is significantly altered, and bacterial populations that normally compose a large percentage of the microbiota become underrepresented (or underrepresented populations become overrepresented), the gut is in a state of *dysbiosis*. **Gut dysbiosis** is important to study because it has been implicated in many physical and neurological human health conditions, including diabetes, rheumatoid arthritis, Alzheimer's, inflammatory bowel disease (IBD), colon cancer (Buford, 2017; Kennedy et al., 2014; Castellarin et al., 2012), chronic fatigue syndrome (Lakhan and Kirchgessner, 2010), obesity (Turnbaugh et al., 2006, 2009), bacterial vaginosis (Africa et al., 2014), ulcerative colitis (Mandal et al., 2015a; Matsuoka and Kanai, 2015), anxiety, and depression (Evrensel and Ceylan, 2015; Lach et al., 2018)

4.2 Relative Abundance Analysis

Relative abundance is a measure of how common or rare a taxon is, relative to other taxa in its community, and is usually expressed as a percentage. For more information on **relative abundance** analyses, see Appendix A.4.

→ For information on **amplicon** sequencing vs. shotgun sequencing, see Appendix A.5.

Operational Taxonomic Unit (OTU): A group of sequences that share significant similarity based on some marker gene. Groups can be treated as a single taxon (e.g., genus, species, etc. depending on the clustering threshold) for analysis purposes. For more info, see Appendix A.6.

BioPype lets users examine the **relative abundance** of bacterial taxa in each experimental sample. Relative abundance analyses help visualize the structure of the micro-organism community. Users can use BioPype to analyze bacterial **amplicon** sequencing data to determine which micro-organisms are present and in what proportion. The idea is to take the sequences from a gut sample and assign them to a taxon. To do that, we group (or cluster) sequences based on their similarity to define **Operational Taxonomic Units (OTUs)**. Using the number of sequences assigned to each OTU/taxon, BioPype can determine the relative abundance of each micro-organism in a sample and plot the results on an interactive bar chart.

For example, studies have found that the genera *Bacteroides* and *Lactobacillus* are both prevalent in the healthy human gut microbiome, but *Bacteroides* has a higher relative abundance than *Lactobacillus* Lloyd-Price et al. (2016). In other words, out of all the sequences obtained from a healthy human gut sample, a larger percentage of those sequences will be assigned to the *Bacteroides* OTU than to the *Lactobacillus* OTU during a relative abundance analysis.

Relative abundance is an important measure of biodiversity in the gut. Using BioPype to perform a relative abundance analysis allows examination of gut biodiversity at 7 different taxonomic levels, which helps identify instances of gut dysbiosis in research subjects.

4.3 Metagenomics

! $\rightarrow$ *THIS FUNCTIONALITY HAS NOT BEEN COMPLETED YET, BUT WILL BE ADDED IN FUTURE UPDATES*

BioPype can also be used to predict which genes are present in microbiome samples. From these predicted genes, users can do two things.

First, they can generalize the functions encoded across all bacteria that are present in an experimental condition's microbiome (i.e., at the metagenomic scale) and visualize the relative distribution of these functions with a bar chart. Having this functional snapshot of a metagenome is useful because a functional overview of the experimental condition can be compared to a functional overview of the control condition to see if there are differences in the two groups' functional capabilities. When comparing the microbiomes of disordered vs. non-disordered samples, having a visual depiction of the differences in functional capabilities can make it easier to find the cause of the disorder by narrowing down the potential sources of the condition.

For example, certain G protein-coupled receptors (GPCRs) in the gut are associated with some of the diseases associated with gut dysbiosis, such as inflammatory bowel disease (IBD) (Cohen et al., 2017). Research has also shown that bacteria interact with their environment via the release of small molecules (Li and Tian, 2012). From these premises, one could hypothesize that IBD may be caused by the over-production of certain effector molecules from the gut microbiome. This hypothesis may be tested by using BioPype to compare the functional capabilities of the IBD microbiome with that of a healthy control microbiome. If the results show, for example, that the IBD gut metagenome has a greater proportion of genes associated with lipid production than the control metagenome, it indicates potential support for the user's hypothesis. They can now refine their investigation by narrowing the scope of their search to only analyze genes associated with lipid synthesis.

Second, users can use BioPype to predict the functions of *individual* genes. This is useful because it allows users to identify a specific gene of interest for their research question, which they can then design a validation experiment for.

Continuing with our above example hypothesis about the gut microbiome?s role in IBD, the user can now turn their focus to the functions of specific lipid-synthesis genes that are being overrepresented in IBD patients' microbiomes. Upon examining the functions of individual genes, they find that N-acyl amide synthase genes are enriched in the IBD microbiome. Wet-lab validation experiments will confirm that the lipids encoded by these genes are able to interact with GPCRs that regulate GI tract physiology (Cohen et al., 2017). This suggests that these lipids, and their associated genes in the microbiome, have the potential to cause IBD during times of dysbiosis by activating GPCRs in the gut (Cohen et al., 2017), thus supporting the user's original hypothesis. Additionally, one can use these findings to focus future research questions about the role of the human microbiome in health disorders by searching for correlations between N-acyl amide synthase genes and other conditions. For instance, since the products of these genes interact with GPCRs to potentially regulate IBD, and IBD is associated with anxiety and depression, investigation may reveal that the interaction between these genes and GPCRs potentially regulate anxiety and depression as well.

4.4 Python

To make use of this tutorial and the BioPype package, you will need to be familiar with the basics of the Python coding language. The free DataCamp course *Intro to Python for Data Science* (`https://www.datacamp.com/courses/intro-to-python-for-data-science?tap_a=5644-dce66f&tap_s=116411-750171`) is a quick and easy introduction to the basics of the language. It covers:

- Variables and variable assignment
- Basic data types (strings, integers, floats, lists, booleans,) and converting between types
- Calculations with variables
- Functions and arguments
- The `help()` function.
- Methods
- Packages and importing packages
- The basics of the NumPy package (The material covered in the NumPy chapter of the DataCamp tutorial is not required for using BioPype, but if you are new to Python, it's a great way to practice/apply the concepts they teach you in Chapters 1-3.)

The DataCamp course should take $\tilde{4}$ hours to complete (less, if you forego NumPy chapter), and will teach you everything you'll need to know about Python in order to make use of BioPype.

If you decide that you'd like to start creating your own tools with Python, there's a bit more you'll need to know. Resources for learning how to create your own functions, classes, packages, and more can be found in Appendix B.

5 How to Find Tools

5.1 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.)

5.2 Finding Software

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

6 The Plan

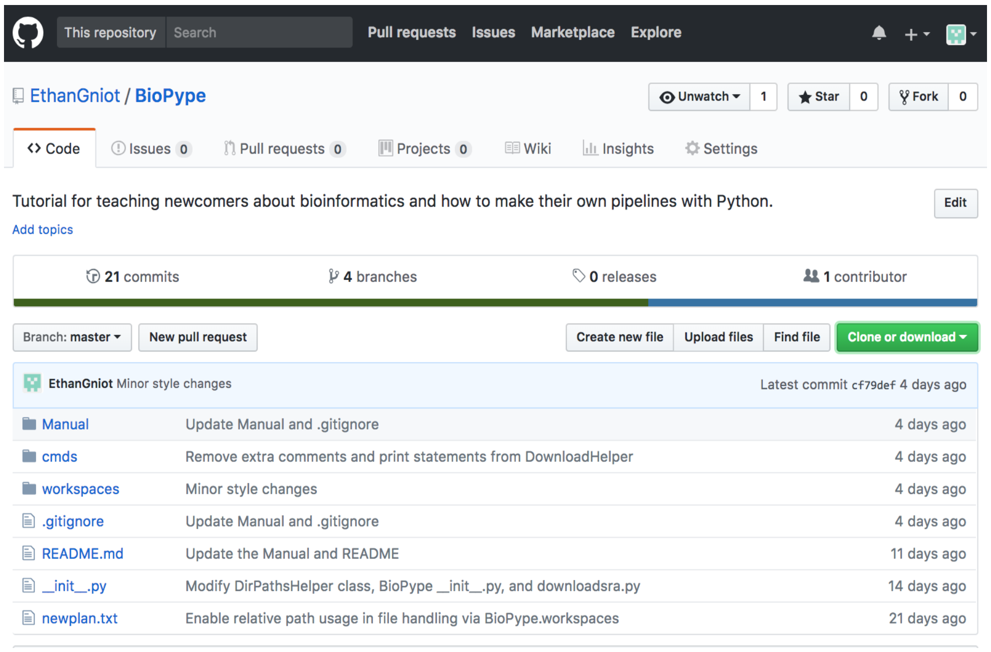
(Break down the sub-tasks required to accomplish the two main tasks: Relative abundance analysis and metagenomic analysis)

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?

7 Software and Set-up

7.1 Downloading BioPype

1. Go to the BioPype Github repository at `https://github.com/EthanGniot/BioPype`. It should bring you to a page that looks like 7.1.
2. Click on the green "Clone or download" button (see Figure 7.1)
   - Note: The Github page that the URL from step 1 brings you to contains the code for the most recent "master" update to the BioPype repository. The "master" updates are usually the most recent *stable* versions of BioPype. However, there may also be other versions of BioPype that are newer, but may contain bugs due to being under active development. To access these versions, click on the "branches" link (in Figure 7.1, it is the link that says "4 branches" next to a branching symbol). If there are any branches (i.e., "versions") of BioPype that are more-current than the "master" branch, you will be able to see them on this page. This page also has links to the branches, where you can download that version of the BioPype code.



Tutorial for teaching newcomers about bioinformatics and how to make their own pipelines with Python.

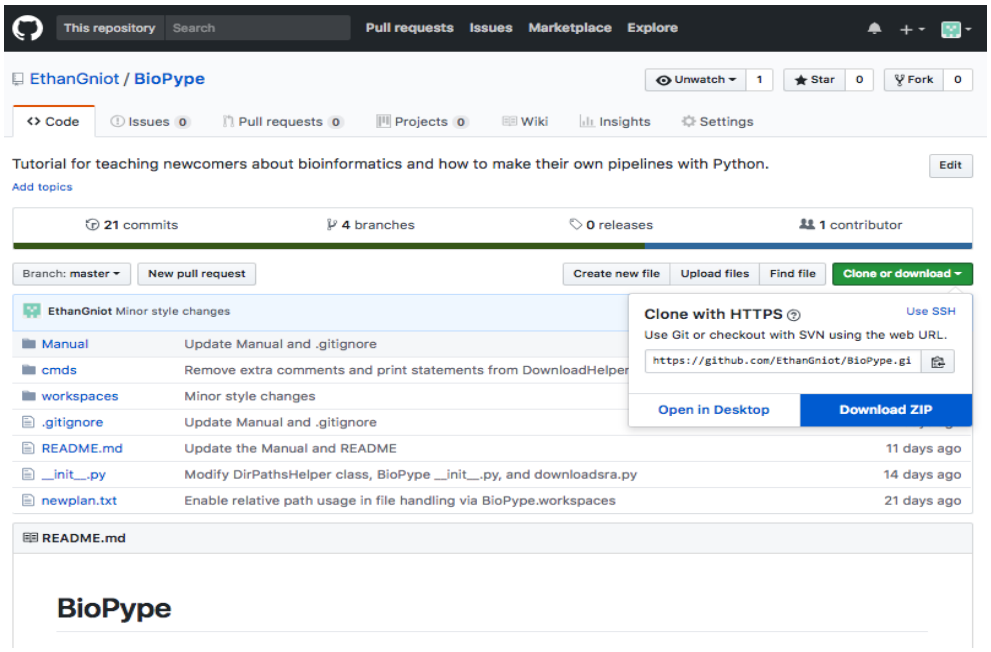
EthanGniot Minor style changes — Latest commit `cf79def` 4 days ago

Manual	Update Manual and .gitignore	4 days ago
cmds	Remove extra comments and print statements from DownloadHelper	4 days ago
workspaces	Minor style changes	4 days ago
.gitignore	Update Manual and .gitignore	4 days ago
README.md	Update the Manual and README	11 days ago
__init__.py	Modify DirPathsHelper class, BioPype __init__.py, and downloadsra.py	14 days ago
newplan.txt	Enable relative path usage in file handling via BioPype.workspaces	21 days ago

Figure 7.1: The BioPype Github webpage.

3. Click on the "Download ZIP" button (the blue button in Figure 7.2).

4. Navigate to your Downloads folder and unzip the BioPype .zip file.



EthanGniot / BioPype

Tutorial for teaching newcomers about bioinformatics and how to make their own pipelines with Python.

21 commits | 4 branches | 0 releases | 1 contributor

EthanGniot Minor style changes

Clone with HTTPS

Use Git or checkout with SVN using the web URL.

https://github.com/EthanGniot/BioPype.gi

Manual	Update Manual and .gitignore	
cmds	Remove extra comments and print statements from DownloadHelper	
workspaces	Minor style changes	
.gitignore	Update Manual and .gitignore	
README.md	Update the Manual and README	11 days ago
__init__.py	Modify DirPathsHelper class, BioPype __init__.py, and downloadsra.py	14 days ago
newplan.txt	Enable relative path usage in file handling via BioPype.workspaces	21 days ago

README.md

BioPype

Figure 7.2: The button to download the BioPype code repository as a .zip file.

7.2 BioPype Dependencies Information

The following tables contain

```
conda create -n test2 --file spec-file.txt
```

Table 7.1: **Python packages that BioPype requires in order to function.**

Package name	*Version number*	*Command-line install*
Anaconda	5.1	
multiqc	1.5	`conda install -c bioconda multiqc`
pandas	0.22.0	`conda install pandas`
parallel-fastq-dump	0.6.3	`conda install -c bioconda parallel-fastq-dump`
qiime2	2018.4	(see above)
sra-tools	2.8.2	`conda install -c bioconda sra-tools`
trim-galore	0.4.5	`conda install -c bioconda trim-galore`

7.3 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, and you won't be able to make use of BioPype:

1. Install and open Anaconda

Table 7.2: **Links to the documentation pages for BioPype's dependencies.**

Software name	*Documentation link*
Anaconda	`https://docs.anaconda.com/anaconda/`
	`https://conda.io/docs/user-guide/getting-started.html`
multiqc	`http://multiqc.info/`
pandas	`https://pandas.pydata.org/pandas-docs/stable/install.html`
parallel-fastq-dump	`https://github.com/rvalieris/parallel-fastq-dump`
qiime2	`https://docs.qiime2.org/2018.4/`
sra-tools	`https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=toolkit_doc`
trim-galore	`https://github.com/FelixKrueger/TrimGalore`

2. Create a new virtual environment
3. Install packages

7.3.1 Install Anaconda or Miniconda

Package manager: (from Wikipedia) a collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs for a computer's operating system in a consistent manner

As you can see in Table **??**, there are several packages that BioPype needs in order to function. Manually downloading and installing packages is an incredible pain, because the quirks of one package frequently conflict with quirks of another. To download the dependencies we need, we will use what's called a **package manager**. Essentially, package managers make it easy to install lots of packages because they can (usually) handle all of the minute details of the process, adjusting the versions and dependencies of each package to resolve any conflicts that would normally wreck the installation.

The package manager that we will be using is called *conda*. There are two ways to obtain conda: the Anaconda distribution, and the Miniconda distribution.

Anaconda is a set of about a hundred packages (many of which are useful scientific computing tools) that also comes bundled with a version of Python. One of these packages is the **conda** package (yes, conda is both a package and a package manager). One of the benefits of Anaconda is that it comes with a graphical interface for downloading and managing new packages called the Anaconda Navigator (see Figures 7.4 and 7.5). Conda can also be used to manage packages from a command-line interface (on a Mac, this is the Terminal application). However, the fact that Anaconda comes with over 100 packages pre-installed can sometimes cause problems when installing new packages. For that reason, you may choose to download Miniconda instead.

Miniconda is a smaller alternative to Anaconda that is *just* conda and its dependencies (as opposed to Anaconda, which is conda and a bunch of other packages). This means that Miniconda doesn't come with a graphical interface like the Anaconda Navigator, so conda must be operated from the command-line. However, it also means that Miniconda requires less storage on your computer (400 MB vs 3 GB) and also avoids some of the difficulties that come from Anaconda's pre-installed packages.

We will cover two methods by which you can download the dependencies for

BioPype using conda, one method uses the command-line with a Miniconda installation (faster, command-line only), and the other uses the Anaconda Navigator with an Anaconda installation (slower, standard graphical interface).

! →

- For the record, the command-line method should theoretically work just as well with an Anaconda installation as it does with a Miniconda installation, but I experienced unknown technical difficulties when trying to download all the dependencies with an Anaconda installation. Due to the unique file-state of individual machines, your mileage may vary.

Installing dependencies with Miniconda

1. Go to `https://conda.io/docs/user-guide/install/index.html#regular-installation`, select your operating system to install Miniconda.

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 7.3) and follow the installation instructions.

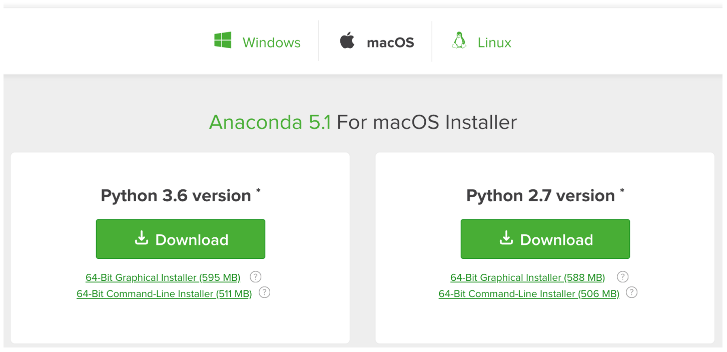

Windows
macOS
Linux
Anaconda 5.1 For macOS Installer
Python 3.6 version *
Download
64-Bit Graphical Installer (595 MB)
64-Bit Command-Line Installer (511 MB)
Python 2.7 version *
Download
64-Bit Graphical Installer (588 MB)
64-Bit Command-Line Installer (506 MB)

Figure 7.3: 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 7.4):

7.3.2 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

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

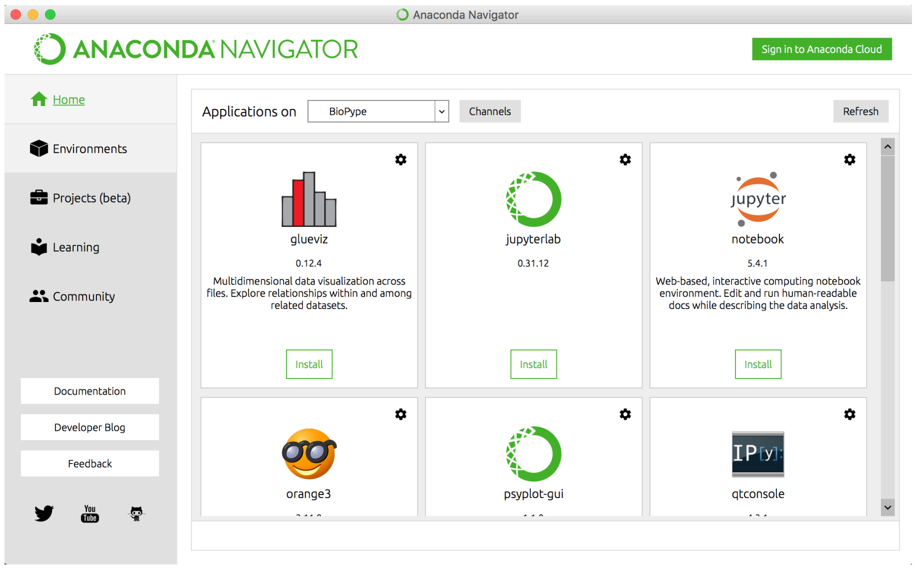

Anaconda Navigator
ANACONDA NAVIGATOR
Sign in to Anaconda Cloud
Home
Environments
Projects (beta)
Learning
Community
Documentation
Developer Blog
Feedback
Applications on
BioPype
Channels
Refresh
glueviz
0.12.4
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install
jupyterlab
0.31.12
Install
notebook
5.4.1
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Install
orange3
psyplot-gui
qtconsole

Figure 7.4: The window displayed to the user upon opening Anaconda-Navigator.

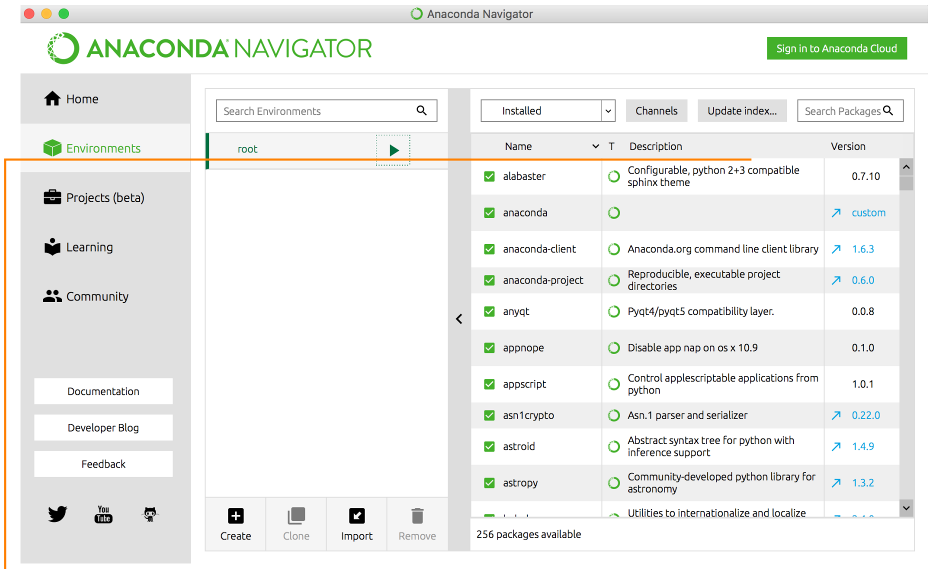
1. On the left side of the Anaconda-Navigator window, click on the tab labeled **Environments**. (Figure 7.5)
2. Click the **Create** button on the bottom of the center panel. A new window titled "Create new environment" will appear. (Figure 7.6)
3. Enter a **Name** for the environment. You may choose 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 a version of Python from the adjacent drop-down menu (Python 3.6 is the most current version at the time of this writing, but the packages we use require Python 3.5 so we chose **3.5**. If you are following the tutorial analysis in this manual, choose version 3.5).
6. Click the **Create** button within the "Create new environment window".

7.3.3 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	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	

256 packages available

Figure 7.5: The Environments window of the Anaconda-Navigator.

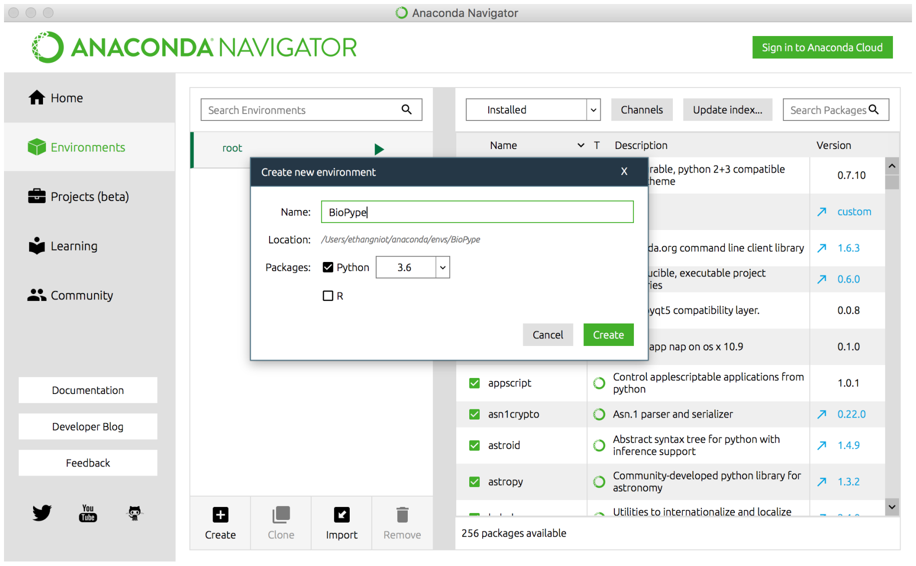
Solve the issue with failing to center figures when they're on their own page

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 7.7)
   - 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.
4. Use the search bar to find and select the other packages listed in Table 7.1. 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 7.8) 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
```



Create new environment

Name: BioPype

Location: /Users/ethangniot/anaconda/envs/BioPype

Packages: ✓ Python 3.6

☐ R

Cancel Create

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	

Figure 7.6: The "Create new environment" window.

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

```
conda list
```

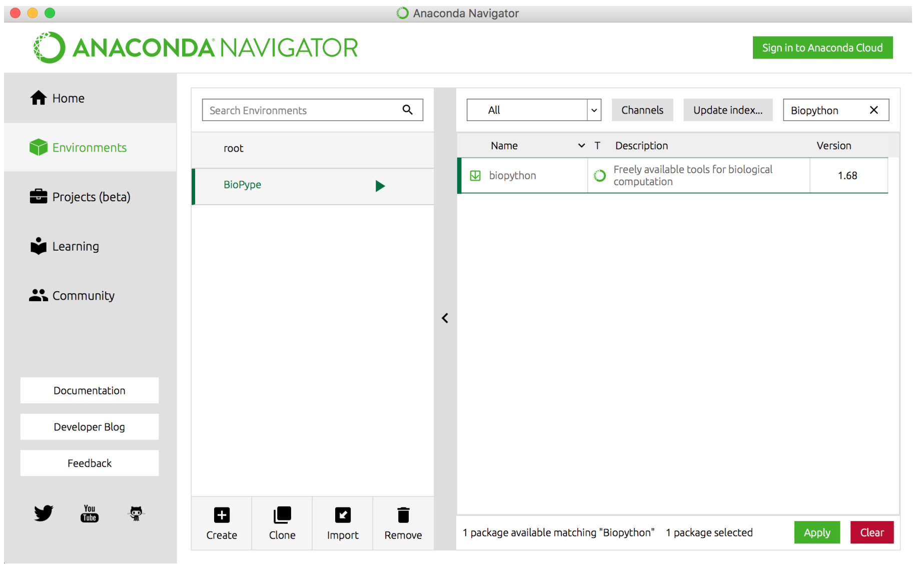
This will generate a list of all the packages installed in the current environment. If you see the **pypiper** package listed, the installation was successful and you may skip the rest of this section. If not, proceed with the following steps.

(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.

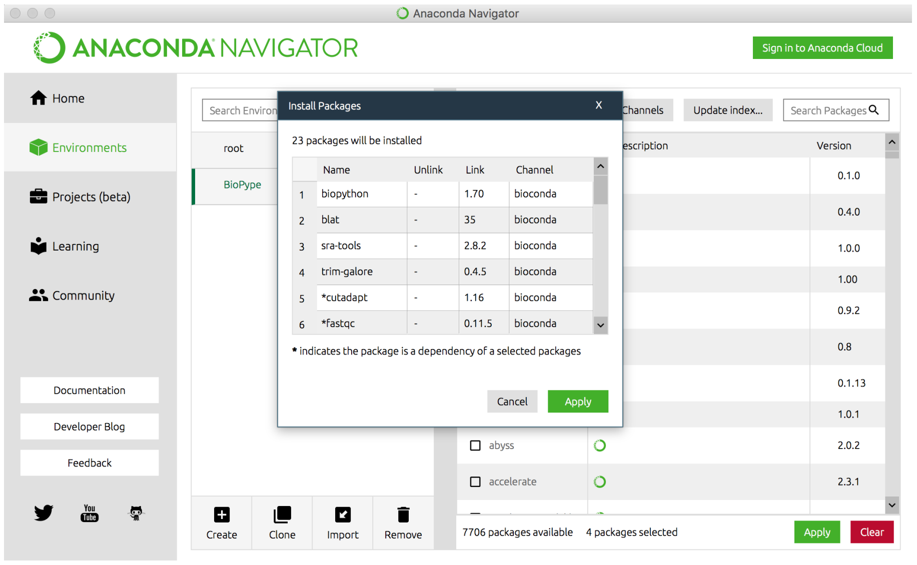
(f) Open a Terminal window, and navigate to the location indicated by the message from the previous step. For my example, I need



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

1 package available matching "Biopython"   1 package selected

Figure 7.7: 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

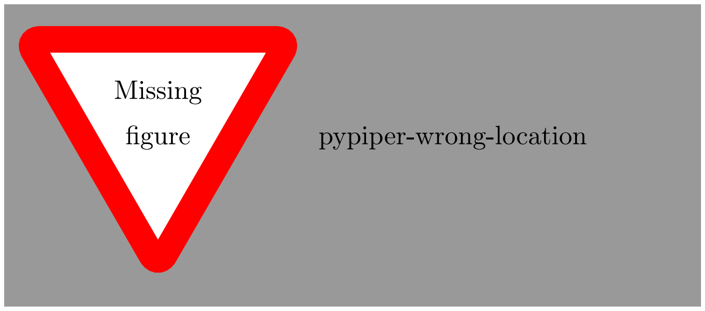
Figure 7.8: The window displaying the packages and dependencies that will be installed.

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.



7.3.4 Integrated Development Environment (IDE)

Talk about choosing between PyCharm and other options

7.3.5 PATH

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

7.4 Analysis Pipeline

(Use figures to illustrate the stages of the pipeline)

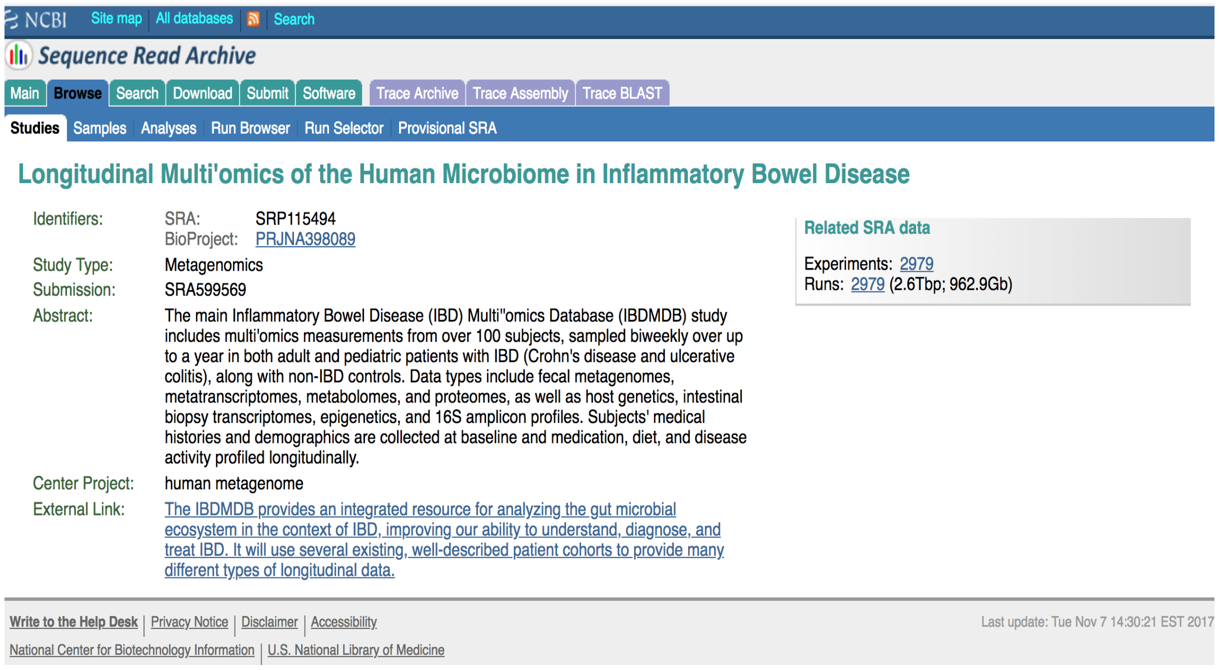
8 The Dataset

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 3, 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 5, 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 8.1 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

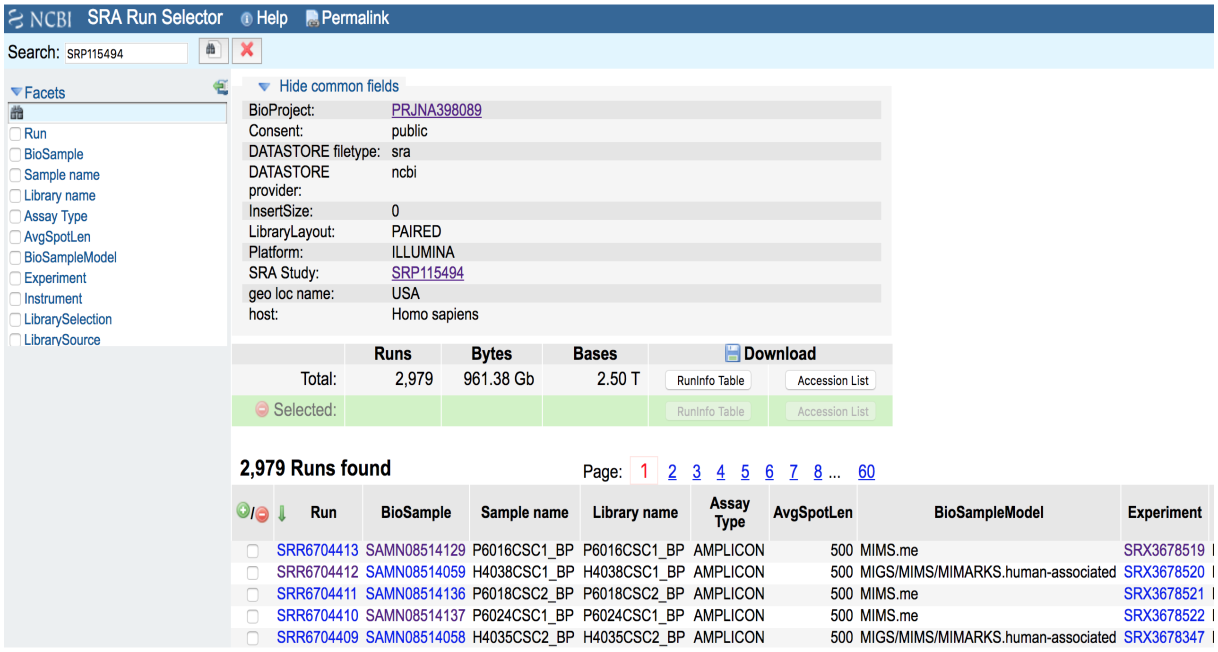
Experiments: 2979
Runs: 2979 (2.6Tbp; 962.9Gb)

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 8.1: The "SRA Study" webpage.

The dataset is reproduced in Figure 8.2 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#` .



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 8.2: 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 5.

8.1 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 8.1, 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 8.2) 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 and put it in the project folder you set up in Chapter 7.1. 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 7 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.11.

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.7.

```
# Shorten the name for the runtable module to save space in the manual:
>>>import BioPype.cmds.runtable as rt

>>>rs_uc_young = rt.RunTable.random_sample_subset(uc_young_nums, n=3)
>>>rs_uc_old = rt.RunTable.random_sample_subset(uc_old_nums, n=3)
>>>rs_cd_young = rt.RunTable.random_sample_subset(cd_young_nums, n=3)
>>>rs_cd_old = rt.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 into named folders:

```
>>>download_sra(rs_uc_young, 'uc_young')
>>>download_sra(rs_uc_old, 'uc_old')
>>>download_sra(rs_cd_young, 'cd_young')
>>>download_sra(rs_cd_old, '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 7 for details).

→ To learn more about .fastq format, see Appendix A.8.

→ To learn more about what threads are, and how many you should use, see Appendix A.9.

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

```
>>>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 7 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.10. 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.

8.1.1 Creating the code

8.2 Perform Quality Control on Dataset

9 Relative Abundance Analysis

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

Appendix A Web Resources and Explanations

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 Microbiota vs Microbiome

"What is the gut microbiota? What is the human microbiome?" from medicalnewstoday.com. https://www.medicalnewstoday.com/articles/307998.php

A.4 Relative- vs Differential-Abundance Analyses

(For more in-depth discussion of differential abundance analyses, see (Mandal et al., 2015b).)

Differential abundance analysis: seeing which samples are most similar/dissimilar based on the raw sequence counts for each sample (E.g., 5 sequences associated with the *Lactobacillus* OTU were found in Sample A, and 10 sequences were found in Sample B). Differential abundance analysis answers the following questions:

- Is Sample A more similar to Sample B, or to Sample C in terms of its microbiome composition? (has an associated p-value)
- Does sequence variant/OTU/taxon/bacterium "X" appear more/less often in Subject A than in Subject B? (raw count)

Relative abundance analysis: seeing which samples are most similar/dissimilar based on the proportion of sequence counts for each sample (E.g., out of all the sequences obtained from Sample A, %50 were associated with the *Lactobacillus* OTU. Out of all the sequences obtained from Sample B, %100 were associated with the *Lactobacillus* OTU. Relative abundance analysis answers the following question:

- Does Sample A have higher/lower percentage of Bacteria X in its microbiome than Sample B?

A.5 Amplicon vs WGS data

http://galaxyproject.github.io/training-material/topics/metagenomics/tutorials/general-tutorial/tutorial.html#amplicon-data

A.6 Operational Taxonomic Units (OTUs)

https://www.drive5.com/usearch/manual/otu_definition.html

A.7 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.8 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.9 How many threads to use?

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

A.10 How many threads can my computer run?

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

A.11 SRA/.sra file format

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

Appendix B Python Resources

B.1 Creating Python Functions

`https://www.tutorialspoint.com/python3/python_functions.htm`
`https://www.datacamp.com/courses/python-data-science-toolbox-part-1` A resources for learning how to create your own Python functions. The DataCamp course is a paid course, but the first chapter, which covers how to write your own functions, is available for free.

B.2 Video Tutorial: Installing and Using Anaconda

`https://www.youtube.com/watch?v=YJC6ldI3hWk`

B.3 Python Dictionaries

`https://docs.python.org/3/tutorial/datastructures.html#dictionaries`

Add more resources for learning python: how to make own packages, classes

References

Africa, C. W. J., Nel, J., and Stemmet, M. (2014). Anaerobes and bacterial vaginosis in pregnancy: virulence factors contributing to vaginal colonisation. *International journal of environmental research and public health*, 11(7):6979–7000.

Buford, T. W. (2017). (Dis)Trust your gut: the gut microbiome in age-related inflammation, health, and disease. *Microbiome*, 5(1):80.

Caporaso, J. G., Lauber, C. L., Costello, E. K., Berg-Lyons, D., Gonzalez, A., Stombaugh, J., Knights, D., Gajer, P., Ravel, J., Fierer, N., Gordon, J. I., and Knight, R. (2011a). Moving pictures of the human microbiome. *Genome biology*, 12(5):R50.

Caporaso, J. G., Lauber, C. L., Walters, W. A., Berg-Lyons, D., Lozupone, C. A., Turnbaugh, P. J., Fierer, N., and Knight, R. (2011b). Global patterns of 16S rRNA diversity at a depth of millions of sequences per sample. *Proceedings of the National Academy of Sciences of the United States of America*, 108 Suppl(Suppl 1):4516–22.

Castellarin, M., Warren, R. L., Freeman, J. D., Dreolini, L., Krzywinski, M., Strauss, J., Barnes, R., Watson, P., Allen-Vercoe, E., Moore, R. A., and Holt, R. A. (2012). Fusobacterium nucleatum infection is prevalent in human colorectal carcinoma. *Genome research*, 22(2):299–306.

Cho, I. and Blaser, M. J. (2012). The human microbiome: at the interface of health and disease. *Nature reviews. Genetics*, 13(4):260–70.

Cohen, L. J., Esterhazy, D., Kim, S.-H., Lemetre, C., Aguilar, R. R., Gordon, E. A., Pickard, A. J., Cross, J. R., Emiliano, A. B., Han, S. M., Chu, J., Vila-Farres, X., Kaplitt, J., Rogoz, A., Calle, P. Y., Hunter, C., Bitok, J. K., and Brady, S. F. (2017). Commensal bacteria make GPCR ligands that mimic human signalling molecules. *Nature*, 549(7670):48–53.

Evrensel, A. and Ceylan, M. E. (2015). The Gut-Brain Axis: The Missing Link in Depression. *Clinical psychopharmacology and neuroscience : the official scientific journal of the Korean College of Neuropsychopharmacology*, 13(3):239–44.

Kennedy, P. J., Cryan, J. F., Dinan, T. G., and Clarke, G. (2014). Irritable bowel syndrome: a microbiome-gut-brain axis disorder? *World journal of gastroenterology*, 20(39):14105–25.

Lach, G., Schellekens, H., Dinan, T. G., and Cryan, J. F. (2018). Anxiety, Depression, and the Microbiome: A Role for Gut Peptides. *Neurotherapeutics*, 15(1):36–59.

Lakhan, S. E. and Kirchgessner, A. (2010). Gut inflammation in chronic fatigue syndrome. *Nutrition & metabolism*, 7:79.

Li, Y.-H. and Tian, X. (2012). Quorum sensing and bacterial social interactions in biofilms. *Sensors (Basel, Switzerland)*, 12(3):2519–38.

Lloyd-Price, J., Abu-Ali, G., and Huttenhower, C. (2016). The healthy human microbiome. *Genome medicine*, 8(1):51.

Mandal, R. S., Saha, S., and Das, S. (2015a). Metagenomic Surveys of Gut Microbiota. *Genomics, Proteomics & Bioinformatics*, 13(3):148–158.

Mandal, S., Van Treuren, W., White, R. A., Eggesbø, M., Knight, R., and Peddada, S. D. (2015b). Analysis of composition of microbiomes: a novel method for studying microbial composition. *Microbial ecology in health and disease*, 26:27663.

Matsuoka, K. and Kanai, T. (2015). The gut microbiota and inflammatory bowel disease. *Seminars in immunopathology*, 37(1):47–55.

Turnbaugh, P. J., Hamady, M., Yatsunenko, T., Cantarel, B. L., Duncan, A., Ley, R. E., Sogin, M. L., Jones, W. J., Roe, B. A., Affourtit, J. P., Egholm, M., Henrissat, B., Heath, A. C., Knight, R., and Gordon, J. I. (2009). A core gut microbiome in obese and lean twins. *Nature*, 457(7228):480–4.

Turnbaugh, P. J., Ley, R. E., Mahowald, M. A., Magrini, V., Mardis, E. R., and Gordon, J. I. (2006). An obesity-associated gut microbiome with increased capacity for energy harvest. *Nature*, 444(7122):1027–131.