OrthoEvolution Documentation

Release 0.1

Robert Gilmore and Shaurita Hutchins

Nov 20, 2017

Contents

1 Overview **3**
- 1.1 Installation . . . 3
  - 1.1.1 PyPi . . . 3
  - 1.1.2 GitHub . . . 3
  - 1.1.3 Development Code . . . 3
- 1.2 Examples . . . 4
- 1.3 Tests . . . 4
- 1.4 Contributors . . . 4
- 1.5 Citations . . . 4
- 1.6 License . . . 4

2 Contents **5**
- 2.1 Tutorial . . . 5
  - 2.1.1 Using the Cookies module . . . 5
  - 2.1.2 Using the Manager module . . . 7
  - 2.1.3 Using the Orthologs Module . . . 8
  - 2.1.4 Using the Pipeline module . . . 8
  - 2.1.5 Using the Tools module . . . 9
- 2.2 Cookies Documentation . . . 9
  - 2.2.1 Examples . . . 10
- 2.3 Manager Documentation . . . 10
  - 2.3.1 Why a manager? . . . 10
  - 2.3.2 Examples . . . 10
  - 2.3.3 Notes . . . 10
- 2.4 Orthologs Documentation . . . 10
  - 2.4.1 Usage & Examples . . . 11
  - 2.4.2 Software Dependencies . . . 11
  - 2.4.3 Using `install.sh` on Debian/Ubuntu: . . . 11
- 2.5 Pipeline Documentation . . . 11
  - 2.5.1 Examples . . . 11
  - 2.5.2 Software Dependencies . . . 11
- 2.6 Tools Documentation . . . 11
  - 2.6.1 Examples . . . 12
  - 2.6.2 Additional Documentation . . . 13

3 Indices and tables **15**

An **easy to use** and comprehensive python package which aids in the **analysis and visualization of comparative evolutionary genetics** related projects. More specifically, this project is focused on the **inference of orthologs** using NCBI's blast, various sequence alignment strategies, and phylogenetics analyses including PAML, PhyML, ete3, and more tools.

Ultimately, the goal of this project is to create a **reusable pipeline** for the inference of orthologs in order to ensure reproducibility of data as well as improve the management and analysis of (what can be) large datasets. The Cookies, Manager, Pipeline, and Tools modules act as a framework for our workflow, while the Orthologs module provides access to specific functions for our various ortholog inference projects.

View our read the docs and feel free to also read this related paper to gain more insight into this project/python package.

CHAPTER 1

Overview

1.1 Installation

View the below methiods for installing this package.

1.1.1 PyPi

```
pip install ortho-evol
```

1.1.2 GitHub

1. Download the zip file and unzip it or `git clone https://github.com/datasnakes/OrthoEvolution.git`
2. `cd OrthoEvolution`
3. `pip install .`

1.1.3 Development Code

WARNING : This code is actively under development and may not be reliable. Please create an issue for questions about development.

1. Download the zip file and unzip it or `git clone -b dev-master https://github.com/datasnakes/OrthoEvolution.git`
2. `cd OrthoEvolution`
3. `pip install .`

1.2 Examples

Check out this tutorial in our Wiki Docs.

```
import OrthoEvol
```

1.3 Tests

To run tests, type `nosetests Tests/` in the OrthoEvolution directory.

1.4 Contributors

This package was created by the Datasnakes.

- Rob Gilmore | Github: @grabear |
- Shaurita Hutchins | Github: @sdhutchins |

If you would like to contribute to this package, install the package in development mode, and check out our contributing guidelines.

1.5 Citations

We're so thankful to have a resource such as Biopython. They inspired this package.

Cock, P.J.A. et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. Bioinformatics 2009 Jun 1; 25(11) 1422-3 http://dx.doi.org/10.1093/bioinformatics/btp163 pmid:19304878

1.6 License

MIT

CHAPTER 2

Contents

2.1 Tutorial

Datasnakes-Scripts has been built with Python 3.5 (and up) as a multi-faceted package and pipeline framework for comparative genetics in order to infer orthologous genes.

Currently, this python package is comprised of 5 major modules:

1. *Cookies Module* - Project structure creation using cookiecutter.
2. *Manager Module* - Configuration management as well project deployment.
3. *Orthologs Module* - Tools for comparative genetics analysis including alignment analysis and phylogenetics.
4. *Pipeline Module* - Various preconfigured pipelines to be used in orthology inference.
5. *Tools Module* - Utilities that aid in ftp downloading, server communication, and reusable everyday functions

When used together, these 4 modules offer a cohesive environment for easily creating, managing, and deploying a bioinformatics pipeline for orthologous genes/species. In the future these tools will also be accessible from the command line and from a web application.

READMEs are provided in each module's directory, but we've compiled a mini tutorial here that can inform users on how to use these modules.

2.1.1 Using the Cookies module

Overview

The Cookies module acts as a repository for custom cookiecutter templates.

Each "CookBook" allows us to quickly create and deploy different projects with various directory structures. They are meant to help organize projects and data in a standardized way. This module is used almost extensively by the Manager module.

In the context of the Manager module the CookBook class is used to deploy an entire repository geared towards developing a web-page using Flask and R-Shiny. Cookies can also be used to create standalone projects that don't require an entire repository.

- Templates used when creating a full repository:
- *Cookies/new_repository*
- *Cookies/new_user*
- *Cookies/new_project*
- *Cookies/new_research*
- *Cookies/new_database* (for NCBI, proprietary, etc. databases)
- *Cookies/new_app* (for R-Shiny applications)
- *Cookies/new_website* (for Flask applications)
- Template for standalone projects
- *Cookies/new_basic_project*

Examples

```
from Datasnakes.Cookies import Oven
from pathlib import Path
import os

# Create the names used.
home = os.getcwd()
repo = "Development"
user = "RAG"
project = "Ortholog"
research = "GPCR"
research_type = "Comparative Genetics"
# Create the paths used
repo_path = Path(home) / Path(repo)
user_path = repo_path / Path('users')
project_path = user_path / Path(user) / Path('projects')
research_path = project_path / Path(project)

# Initialize the Oven object to create a full repository
Full_Kitchen = Oven(repo=repo, user=user, project=project, basic_project=False,␣
↪output_dir=home)
# Create the new project
Full_Kitchen.bake_the_repo()
Full_Kitchen.bake_the_user(cookie_jar=user_path)
Full_Kitchen.bake_the_project(cookie_jar=project_path)
Full_Kitchen.bake_the_research(research=research, research_type=research_type, cookie_
↪jar=research_path)

# Initialize the Oven object to setup a basic project
Basic_Kitchen = Oven(project=project, basic_project=True, output_dir=home)
# Create the new project
Basic_Kitchen.bake_the_project()
```

2.1.2 Using the Manager module

Overview

The Manager module uses the CookBook class in order to deploy a bioinformatics repository with an organized directory structure based on specific users and the projects that they create. Pipeline customization and configuration will also be possible through YAML files.

Future Direction

First, a database_management class for dealing with the various databases (NCBI, BioSQL, etc.) will be developed. Then the Management class will become responsible for functioning alongside Flask in order to create a web interface. The web interface will give each user access to the Tools and Orthologs modules as well as data generated by the pipeline functionality.

Examples

```
# Manager classes can be used explicitly, or...
from Datasnakes.Manager.management import Management
from Datasnakes.Manager.management import RepoManagement
from Datasnakes.Manager.management import UserManagement
from Datasnakes.Manager.management import WebsiteManagement
from Datasnakes.Manager.management import ProjectManagement

# ...they can be use implicitly through the main pipeline class.
from Datasnakes.Manager.data_management import DataMana
```

Explicit Usage

```
from Datasnakes.Manager.management import ProjectManagement
# Use the flags to create a new repository/user/project/research directory system
pm = ProjectManagement(repo="repository1", user='user1', project='project1', research=
↪'research1',
    research_type='comparative_genetics', new_repo=True, new_user=True, new_
↪project=True, new_research=True)
# Access the path variables
print(pm.research_path)
print(pm.research)
print(pm.Pantry.research_cookie)
```

Implicit Usage

```
from Datasnakes.Manager.data_management import DataMana
# Use a prebuilt configuration file in Manager/config/
# *start* a *new* project automatically
# This builds everything and then starts the pipeline
import os
pipeline = DataMana(pipeline='Ortho_CDS_1', project_path=os.getcwd(), start=True,␣
↪new=True)
```

2.1.3 Using the Orthologs Module

Overview

The Orthologs module is the central data processing unit of our package. Any published data will be generated using these submodules.

The sub modules are used for BLASTing NCBI's refseq database to discover orthologous genes, parsing and analyzing BLASTn data, generating GenBank files for the orthologs, generating sequence data for the orthologs, aligning the orthologous sequences for each gene, generating phylogenetic trees for each gene, and doing phylogenetic analysis for each gene.

Examples

```
from Datasnakes.Manager.management import ProjectManagement
from Datasnakes.Orthologs.Blast.blastn_comparative_genetics import OrthoBlastN
from Datasnakes.Orthologs.GenBank.genbank import GenBank
from Datasnakes.Orthologs.Align.msa import MultipleSequenceAlignment as MSA

# In a real situation a dictionary configuration from YAML files will be used
# However a dictionary can be manually set up by the user within the script
# See the config files in Manager/config or use data_management.py as an example
management_cfg = mlast_cfg = genbank_cfg = alignment_cfg = {}

# Initialize the Project Manager
proj_mana = ProjectManagement(**management_cfg)

# Initialize the BLAST tool
# Compose this class with the Project Manager
myblast = OrthoBlastN(proj_mana=proj_mana, **management_cfg, **blast_cfg)
myblast.blast_config(myblast.blast_human, 'Homo_sapiens', auto_start=True)

# Initialize the GenBank parser
# Compose this class with the BLAST tool
# Implicitly uses the Project Manager as well
genbank = GenBank(blast=blast, **management_cfg, **genbank_cfg)
# Use the Blast tool data to get the desired GenBank files
genbank.blast2_gbk_files(myblast.org_list, myblast.gene_dict)

# Initialize the Aligner
# Compose this class with the GenBank parser
# Implicitly uses the Project Manager and the BLAST tool as well
al = MSA(genbank=genbank, **management_cfg, **alignment_cfg)
al.align(alignment_config['kwargs'])  # Underdeveloped
```

2.1.4 Using the Pipeline module

The pipeline module integrates the python package *luigi* with our package to create a pipeline that is accessible via the command-line and can be utilized with a qsub/pbs job scheduling system.

Examples

2.1.5 Using the Tools module

The tools module is a grouping of utilities used by our package. While they could have be stored in each modules util.py file, they were used and developed on a global scale, and hence required their own module.

Overview

Some of the tools/classes in the tools module are:

- `NcbiFTPClient` - provides functions to easily download ncbi databases/files and update them.
- `LogIt` - A wrapper around logzero for easy logging to the screen or a file.
- `Multiprocess` - A simple and effective class that allows the input of a function to map to a user's list in order to take advantage of parallel computing.
- `SGEJob` - A class to aid in submission of a job via `qsub` on a cluster.
- `Qstat` - A class that parses the output of `qstat` to return job information. It also waits on job completion.
- `ZipUtils` -
- `Slackify` -
- `MyGene` -

Can I integrate these tools with each other and with orther modules including my own? **YES!** We'll provide some examples below!

Examples

```
# Import a tools module
from Datasnakes.Tools import Slackify

# Slack takes a config file thats already set up
slack = Slackify(slackconfig='path/to/slackconfig.cfg')

# Message a channel and link to a user.

message_to_channel = 'Hey, <@username>. This is an update for the current script.'
slack.send_msg(channel='channelname', message=message_to_channel)
```

For more information, view the slackify readme.

2.2 Cookies Documentation

For this project/package, we recommend using cookiecutter (along with Flask) to set up your directory if you intend to create a web app/interface for your project.

Cookies makes it very easy to do this.

Learn more about the cookiecutter package.

2.2.1 Examples

The Manager module uses the *CookieRecipes* and *Oven* classes as a primary means of functioning.

Simple Implementation

```
from OrthoEvol.Cookies import Oven

Kitchen = Oven(repo="repo", user="user", project="project", output_dir="project_path")
Pantry = Kitchen.Ingredients
Kitchen.bake_the_*()
```

2.3 Manager Documentation

The classes and functions in this module have been designed to help manage existing and new projects using the Cookies module as well as the different utilities found in the Tools module.

2.3.1 Why a manager?

This module is intended to mesh with a Flask user interface. * Whenever a new website is made the RepoManagement and WebManagement classes are used. * Whenever a new user is created in the Flask webpage, the UserManagement class is used. * Whenever an existing user creates a new project, the ProjectManagement class is used.

However, this module does not have to be used to create a Flask webpage. The full repository can be used for higher level organization, or standalone projects can be made using the ProjectManagements *basic_project* flag.

The `DataManagement` class helps to tie everything together into a pipeline.

2.3.2 Examples

Beware that this is under heavy development. ### Utilizing DataManagement to run a pipeline

```
import os
from OrthoEvol.Manager import DataManagement

DataManagement(pipeline="Ortho_CDS_1", start=True, new=True)
```

Utilizing DatabaseManagement to download databases

2.3.3 Notes

Please view our BioSQL documentation and view some of the static/config related files.

2.4 Orthologs Documentation

This top level module includes submodules such as Align (for aligning multi fasta files), Phylogenetics (for analyzing multiple sequence alignments), **'BioSQL <>'__** (for database creation), Blast (includes tools for using NCBI's blastn command line), and Genbank. (for tools to extract features from genbank files).

2.4.1 Usage & Examples

These classes are optimized to be used together (very little work to do that), but can also be used as standalone classes/methods.

This is a simple example of using all of the `Orthologs` submodules together.

```
from OrthoEvol.Orthologs.Blast import OrthoBlastN
from OrthoEvol.Orthologs.Align import ClustalO
from OrthoEvol.Orthologs.Phlogenetics import ETE3PAML
```

2.4.2 Software Dependencies

Ensure that the following software is installed and in your path: - Clustal omega - NCBI Blast+ 2.6.0 or greater - PAML - PhyML - Phylip - IQTREE - Mafft - Prank - Clustalw - Guidance2 - Pal2Nal

If you are a sudo user, you may use the script we've provided, install.sh.

2.4.3 Using `install.sh` on Debian/Ubuntu:

```
# Change to the directory of the file.
cd
chmod +x install.sh
./sudo-install.sh
```

2.5 Pipeline Documentation

The Pipeline module is designed to provide the user with easily callable and command line usable pipelines that allow orthology inference to be completed in a parallel fashion.

Soon, there will be many preconfigured pipelines that you can run if you are using a cluster (**specifically one that uses pbspro or sun grid engine**).

2.5.1 Examples

Running a Blast Pipeline

2.5.2 Software Dependencies

Ensure that you have at least pbs version `14.1.0`

2.6 Tools Documentation

The Tools module is a collection of often used classes or functions that either enhance our other modules and create reusable functions to be used in various modules.

We've incorporated tools for sge tools for use with pbs, a pandoc script and class for converting docx files to markdown formats, multiprocessing in multiprocess, and a ftp module that aids in downloading files from NCBI's ftp repository.

2.6.1 Examples

Take a look at the examples below to get an idea of how to incorporate these tools in your project and how we use these tools in our project.

Download NCBI databases with our NCBI FTP Client

```
from OrthoEvol.Tools.ftp import NcbiFTPClient

ncbiftp = NcbiFTPClient(email='somebody@gmail.com')
ncbiftp.getblastdb(database_name='refseq_rna')
```

List all subdirectories in a NCBI FTP Path

```
ncbiftp.listdirectories(path='/blast/db/')
Out[54]: ['FASTA', 'cloud']
```

Utilize multiprocessing to speed up your code

```
from OrthoEvol.Tools import Multiprocess


def printwords(word):
    print(word)


words = ['bae', 'luh', 'cuh']

if __name__ == '__main__':
    mp = Multiprocess()
    mp.map2function(printwords, words)
```

Integrate logging in a simple and quick way

```
from OrthoEvol.Tools import LogIt

# Set up your loggers
logit = LogIt()

# Log to one file
logfile = 'test.log'

test1 = logit.default('test1 log', logfile)

# Start logging
test1.info('hi')

# Shutdown logging without deleting the logfile
logit.shutdown()
```

2.6.2 Additional Documentation

Check the specific modules for more detailed readmes and examples of using the tools with this package.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search