

---

# **YMP Documentation**

***Release 0.2.1***

**Elmar Pruesse**

**Aug 17, 2020**



# CONTENTS

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>YMP - a Flexible Omics Pipeline</b>                | <b>1</b>  |
| 1.1      | Features: . . . . .                                   | 1         |
| 1.2      | Background . . . . .                                  | 2         |
| <b>2</b> | <b>Installing and Updating YMP</b>                    | <b>3</b>  |
| 2.1      | Working with the Github Development Version . . . . . | 3         |
| <b>3</b> | <b>Configuration</b>                                  | <b>5</b>  |
| 3.1      | Getting Started . . . . .                             | 5         |
| 3.2      | Referencing Read Files . . . . .                      | 6         |
| 3.3      | Project Configuration . . . . .                       | 7         |
| <b>4</b> | <b>Command Line</b>                                   | <b>9</b>  |
| 4.1      | ymp . . . . .   | 9         |
| <b>5</b> | <b>Stages</b>   | <b>25</b> |
| <b>6</b> | <b>API</b>  | <b>31</b> |
| 6.1      | ymp package . . . . .                                 | 31        |
| <b>7</b> | <b>Indices and tables</b>                             | <b>71</b> |
|          | <b>Python Module Index</b>                            | <b>73</b> |
|          | <b>Index</b>  | <b>75</b> |



## YMP - A FLEXIBLE OMICS PIPELINE

Welcome to the YMP documentation!

YMP is a tool that makes it easy to process large amounts of NGS read data. It comes “batteries included” with everything needed to preprocess your reads (QC, trimming, contaminant removal), assemble metagenomes, annotate assemblies, or assemble and quantify RNA-Seq transcripts, offering a choice of tools for each of those processing stages. When your needs exceed what the stock YMP processing stages provide, you can easily add your own, using YMP to drive novel tools, tools specific to your area of research, or tools you wrote yourself.

### 1.1 Features:

**batteries included** YMP comes with a large number of *Stages* implementing common read processing steps. These stages cover the most common topics, including quality control, filtering and sorting of reads, assembly of metagenomes and transcripts, read mapping, community profiling, visualisation and pathway analysis.

For a complete list, check the [documentation](#) or the [source](#).

**get started quickly** Simply point YMP at a folder containing read files, at a mapping file, a list of URLs or even an SRA RunTable and YMP will configure itself. Use tab expansion to complete your desired series of stages to be applied to your data. YMP will then proceed to do your bidding, downloading raw read files and reference databases as needed, installing requisite software environments and scheduling the execution of tools either locally or on your cluster.

**explore alternative workflows** Not sure which assembler works best for your data, or what the effect of more stringent quality trimming would be? YMP is made for this! By keeping the output of each stage in a folder named to match the stack of applied stages, YMP can manage many variant workflows in parallel, while minimizing the amount of duplicate computation and storage.

**go beyond the beaten path** Built on top of [Bioconda](#) and [Snakemake](#), YMP is easily extended with your own Snakefiles, allowing you to integrate any type of processing you desire into YMP, including your own, custom made tools. Within the YMP framework, you can also make use of the extensions to the Snakemake language provided by YMP (default values, inheritance, recursive wildcard expansion, etc.), making writing rules less error prone and repetitive.

## 1.2 Background

Bioinformatical data processing workflows can easily get very complex, even convoluted. On the way from the raw read data to publishable results, a sizeable collection of tools needs to be applied, intermediate outputs verified, reference databases selected, and summary data produced. A host of data files must be managed, processed individually or aggregated by host or spatial transect along the way. And, of course, to arrive at a workflow that is just right for a particular study, many alternative workflow variants need to be evaluated. Which tools perform best? Which parameters are right? Does re-ordering steps make a difference? Should the data be assembled individually, grouped, or should a grand co-assembly be computed? Which reference database is most appropriate?

Answering these questions is a time consuming process, justifying the plethora of published ready made pipelines each providing a polished workflow for a typical study type or use case. The price for the convenience of such a polished pipeline is the lack of flexibility - they are not meant to be adapted or extended to match the needs of a particular study. Workflow management systems on the other hand offer great flexibility by focussing on the orchestration of user defined workflows, but typically require significant initial effort as they come without predefined workflows.

YMP strives to walk the middle ground between these. It brings everything needed to classic metagenome and RNA-Seq workflows, yet built on the workflow management system [Snakemake](#), it can be easily expanded by simply adding Snakemake rules files. Designed around the needs of processing primarily multi-omic NGS read data, it brings a framework for handling read file meta data, provisioning reference databases, and organizing rules into semantic stages.

## INSTALLING AND UPDATING YMP

### 2.1 Working with the Github Development Version

#### 2.1.1 Installing from GitHub

1. Clone the repository:

```
git clone --recurse-submodules https://github.com/epruesse/ymp.git
```

Or, if you have github ssh keys set up:

```
git clone --recurse-submodules git@github.com:epruesse/ymp.git
```

2. Create and activate conda environment:

```
conda env create -n ymp --file environment.yaml  
source activate ymp
```

3. Install YMP into conda environment:

```
pip install -e .
```

4. Verify that YMP works:

```
source activate ymp  
ymp --help
```

#### 2.1.2 Updating Development Version

Usually, all you need to do is a pull:

```
git pull  
git submodule update --recursive --remote
```

If environments were updated, you may want to regenerate the local installations and clean out environments no longer used to save disk space:

```
source activate ymp  
ymp env update  
ymp env clean  
# alternatively, you can just delete existing envs and let YMP
```

(continues on next page)

(continued from previous page)

```
# reinstall as needed:  
# rm -rf ~/.ymp/conda*  
conda clean -a
```

If you see errors before jobs are executed, the core requirements may have changed. To update the YMP conda environment, enter the folder where you installed YMP and run the following:

```
source activate ymp  
conda env update --file environment.yaml
```

If something changed in `setup.py`, a re-install may be necessary:

```
source activate ymp  
pip install -U -e .
```



## CONFIGURATION

YMP reads its configuration from a YAML formatted file `ymp.yml`. To run YMP, you need to first tell it which datasets you want to process and where it can find them.

### Contents

- *Getting Started*
- *Referencing Read Files*
- *Project Configuration*
  - *Specifying Columns*
    - \* *Example*
  - *Multiple Mapping Files per Project*
  - *Complete Example*

## 3.1 Getting Started

A simple configuration looks like this:

```
projects:
  myproject:
    data: mapping.csv
```

This tells YMP to look for a file `mapping.csv` located in the same folder as your `ymp.yml` listing the datasets for the project `myproject`. By default, YMP will use the left most unique column as names for your datasets and try to guess which columns point to your input data.

The matching `mapping.csv` might look like this:

```
sample, fq1, fq2
foot, sample1_1.fq.gz, sample1_2.fq.gz
hand, sample2_1.fq, gz, sample2_2.fq.gz
```

So we have two samples, `foot` and `hand`, and the read files for those in the same directory as the configuration file. Using relative or absolute paths you can point to any place in your filesystem. You can also use SRA references like `SRR123456` or URLs pointing to remote files.

The mapping file itself may be in comma separated or tab separated format or may be an Excel file. For Excel files, you may specify the sheet to be used separated from the file name by a `%` sign. For example:

```
project:
  myproject:
    data: myproject.xlsx%sheet3
```

The matching Excel file could then have a `sheet3` with this content:

| sample | fq1   | fq2   | srr       |
|--------|---|---|-----------|
| foot   | /data/foot1.fq.gz   | /data/foot2.fq.gz   |           |
| hand   |   |   | SRR123456 |
| head   | <a href="http://datahost/head1.fq.gz">http://datahost/head1.fq.gz</a> | <a href="http://datahost/head2.fq.gz">http://datahost/head2.fq.gz</a> | SRR234234 |

For `foot`, the two gzipped FastQ files are used. The data for `hand` is retrieved from SRA and the data for `head` downloaded from `datahost`. The SRR number for `head` is ignored as the URL pair is found first.

## 3.2 Referencing Read Files

YMP will search your map file data for references to the read data files. It understands three types of references to your reads:

**Local FastQ files:** `data/some_1.fq.gz`, `data/some_2.fq.gz` The file names should end in `.fastq` or `.fq`, optionally followed by `.gz` if your data is compressed. You need to provide forward and reverse reads in separate columns; the left most column is assumed to refer to the forward reads.

If the filename is relative (does not start with a `/`), it is assumed to be relative to the location of `ymp.yml`.

**Remote FastQ files:** `http://myhost/some_1.fq.gz`, `http://myhost/some_2.fq.gz` If the filename starts with `http://` or `https://`, YMP will download the files automatically.

Forward and reverse reads need to be either both local or both remote.

**SRA Run IDs:** `SRR123456` Instead of giving names for FastQ files, you may provide SRA Run accessions, e.g. `SRR123456` (or `ERRnnn` or `DRRnnn` for runs originally submitted to EMBL or DDBJ, respectively). YMP will use `fastq-dump` to download and extract the SRA files.

Which type to use is determined for each row in your map file data individually. From left to right, the first recognized data source is used in the order they are listed above.

Configuration processing an SRA RunTable:

```
projects:
  smith17:
    data:
      - SraRunTable.txt
    id_col: Sample_Name_s
```

## 3.3 Project Configuration

Each project must have a `data` key defining which mapping file(s) to load. This may be a simple string referring to the file (URLs are OK as well) or a more *complex configuration*.

### 3.3.1 Specifying Columns

By default, YMP will choose the columns to use as data set name and to locate the read data automatically. You can override this behavior by specifying the columns explicitly:

1. Data set names: `id_col: Sample`

The left most unique column may not always be the most informative to use as names for the datasets. In the above example, we specify the column to use explicitly with the line `id_col: Sample_Name_s` as the columns in SRA run tables are sorted alpha-numerically and the left most unique one may well contain random numeric data.

Default: left most unique column

2. Data set read columns: `reads_cols: [fq1, fq2]`

If your map files contain multiple references to source files, e.g. local and remote, and the order of preference used by YMP does not meet your needs you can restrict the search for suitable data references to a set of columns using the key `read_cols`.

Default: all columns

#### Example

```
projects:
  smith17:
    data:
      - SraRunTable.txt
    id_col: Sample_Name_s
    read_cols: Run_s
```

### 3.3.2 Multiple Mapping Files per Project

To combine data sets from multiple mapping files, simply list the files under the `data` key:

```
projects:
  myproject:
    data:
      - sequencing_run_1.txt
      - sequencing_run_2.txt
```

The files should at least share one column containing unique values to use as names for the datasets.

If you need to merge meta-data spread over multiple files, you can use the `join` key:

```
project:
  myproject:
    data:
      - join:
```

(continues on next page)

(continued from previous page)

```
- SraRunTable.txt
- metadata.xlsx%reference_project
- metadata.xlsx%our_samples
```

This will merge rows from `SraRunTable.txt` with rows in the `reference_project` sheet in `metadata.xls` if all columns of the same name contain the same data (natural join) and add samples from the `our_samples` sheet to the bottom of the list.

### 3.3.3 Complete Example

```
projects:
  myproject:
    data:
      - join:
        - SraRunTable.txt
        - metadata.xlsx%reference_project
      - metadata.xlsx%our_samples
      - mapping.csv
    id_col: Sample
    read_cols:
      - fq1
      - fq2
      - Run_s
```

## COMMAND LINE

### 4.1 ymp

Welcome to YMP!

Please find the full manual at <https://ymp.readthedocs.io>

```
ymp [OPTIONS] COMMAND [ARGS]...
```

#### Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file <log\_file>**  
Specify a log file
- version**  
Show the version and exit.
- install-completion**  
Install command completion for the current shell. Make sure to have psutil installed.
- profile <profile>**  
Profile execution time using Yappi

#### 4.1.1 env

Manipulate conda software environments

These commands allow accessing the conda software environments managed by YMP. Use e.g.

```
>>> $(ymp env activate multiqc)
```

to enter the software environment for multiqc.

```
ymp env [OPTIONS] COMMAND [ARGS]...
```

## Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file** <log\_file>  
Specify a log file

## activate

source activate environment

Usage: \$(ymp activate env [ENVNAME])

```
ymp env activate [OPTIONS] ENVNAME
```

## Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file** <log\_file>  
Specify a log file

## Arguments

**ENVNAME**  
Required argument

## clean

Remove unused conda environments

```
ymp env clean [OPTIONS] [ENVNAMES]...
```

## Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file <log\_file>**  
Specify a log file
- a, --all**  
Delete all environments

## Arguments

- ENVNAMES**  
Optional argument(s)

## export

Export conda environments

Resolved package specifications for the selected conda environments can be exported either in YAML format suitable for use with `conda env create -f FILE` or in TXT format containing a list of URLs suitable for use with `conda create --file FILE`. Please note that the TXT format is platform specific.

If other formats are desired, use `ymp env list` to view the environments' installation path ("prefix" in conda lingo) and export the specification with the `conda` command line utility directly.

Note:

Environments must be installed before they can be exported. This is due to limitations of the conda utilities. Use the "--create" flag to automatically install missing environments.

```
ymp env export [OPTIONS] [ENVNAMES]...
```

## Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file <log\_file>**  
Specify a log file

- d, --dest** <FILE>  
Destination file or directory. If a directory, file names will be derived from environment names and selected export format. Default: print to standard output.
  - f, --overwrite**  
Overwrite existing files
  - c, --create-missing**  
Create environments not yet installed
  - s, --skip-missing**  
Skip environments not yet installed
  - t, --filetype** <filetype>  
Select export format. Default: yml unless FILE ends in '.txt'
- Options** ymltxt

## Arguments

**ENVNAMES**  
Optional argument(s)

## install

Install conda software environments

```
ymp env install [OPTIONS] [ENVNAMES]...
```

## Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file** <log\_file>  
Specify a log file
- p, --conda-prefix** <conda\_prefix>  
Override location for conda environments
- e, --conda-env-spec** <conda\_env\_spec>  
Override conda env specs settings
- n, --dry-run**  
Only show what would be done
- f, --force**  
Install environment even if it already exists



## Arguments

### ENVNAMES

Optional argument(s)

## list

List conda environments

```
ymp env list [OPTIONS] [ENVNAMES]...
```

## Options

- P, --pdb**  
Drop into debugger on uncaught exception
  - q, --quiet**  
Decrease log verbosity
  - v, --verbose**  
Increase log verbosity
  - log-file <log\_file>**  
Specify a log file
  - static, --no-static**  
List environments statically defined via env.yml files
  - dynamic, --no-dynamic**  
List environments defined inline from rule files
  - a, --all**  
List all environments, including outdated ones.
  - s, --sort <sort\_col>**  
Sort by column
- Options** name|hash|path|installed
- r, --reverse**  
Reverse sort order

## Arguments

### ENVNAMES

Optional argument(s)

## prepare

Create envs needed to build target

```
ymp env prepare [OPTIONS] TARGET_FILES
```

## Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file** <log\_file>  
Specify a log file
- n, --dryrun**  
Only show what would be done
- p, --printshellcmds**  
Print shell commands to be executed on shell
- k, --keepgoing**  
Don't stop after failed job
- lock, --no-lock**  
Use/don't use locking to prevent clobbering of files by parallel instances of YMP running
- rerun-incomplete, --ri**  
Re-run jobs left incomplete in last run
- F, --forceall**  
Force rebuilding of all stages leading to target
- f, --force**  
Force rebuilding of target
- notemp**  
Do not remove temporary files
- t, --touch**  
Only touch files, faking update
- shadow-prefix** <shadow\_prefix>  
Directory to place data for shadowed rules
- r, --reason**  
Print reason for executing rule
- N, --nohup**  
Don't die once the terminal goes away.

## Arguments

### TARGET\_FILES

Optional argument(s)

## remove

Remove conda environments

```
ymp env remove [OPTIONS] [ENVNAMES]...
```

## Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file <log\_file>**  
Specify a log file

## Arguments

### ENVNAMES

Optional argument(s)

## run

Execute COMMAND with activated environment ENV

Usage: ymp env run <ENV> [-] <COMMAND...>

(Use the “-” if your command line contains option type parameters beginning with - or -)

```
ymp env run [OPTIONS] ENVNAME [COMMAND]...
```

## Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file <log\_file>**  
Specify a log file

## Arguments

### ENVNAME

Required argument

### COMMAND

Optional argument(s)

## update

Update conda environments

```
ymp env update [OPTIONS] [ENVNAMES]...
```

## Options

### -P, --pdb

Drop into debugger on uncaught exception

### -q, --quiet

Decrease log verbosity

### -v, --verbose

Increase log verbosity

### --log-file <log\_file>

Specify a log file

### --reinstall <reinstall>

Remove and reinstall environments rather than trying to update

## Arguments

### ENVNAMES

Optional argument(s)

## 4.1.2 init

Initialize YMP workspace

```
ymp init [OPTIONS] COMMAND [ARGS]...
```

## Options

### -P, --pdb

Drop into debugger on uncaught exception

### -q, --quiet

Decrease log verbosity

### -v, --verbose

Increase log verbosity

**--log-file** <log\_file>  
Specify a log file

## cluster

Set up cluster

```
ymp init cluster [OPTIONS]
```

### Options

**-P, --pdb**  
Drop into debugger on uncaught exception

**-q, --quiet**  
Decrease log verbosity

**-v, --verbose**  
Increase log verbosity

**--log-file** <log\_file>  
Specify a log file

**-y, --yes**  
Confirm every prompt

## demo

Copies YMP tutorial data into the current working directory

```
ymp init demo [OPTIONS]
```

### Options

**-P, --pdb**  
Drop into debugger on uncaught exception

**-q, --quiet**  
Decrease log verbosity

**-v, --verbose**  
Increase log verbosity

**--log-file** <log\_file>  
Specify a log file

## project

```
ymp init project [OPTIONS] [NAME]
```

### Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file** <log\_file>  
Specify a log file
- y, --yes**  
Confirm every prompt

### Arguments

**NAME**  
Optional argument

## 4.1.3 make

Build target(s) locally

```
ymp make [OPTIONS] TARGET_FILES
```

### Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file** <log\_file>  
Specify a log file
- n, --dryrun**  
Only show what would be done
- p, --printshellcmds**  
Print shell commands to be executed on shell
- k, --keepgoing**  
Don't stop after failed job

**--lock, --no-lock**  
Use/don't use locking to prevent clobbering of files by parallel instances of YMP running

**--rerun-incomplete, --ri**  
Re-run jobs left incomplete in last run

**-F, --forceall**  
Force rebuilding of all stages leading to target

**-f, --force**  
Force rebuilding of target

**--notemp**  
Do not remove temporary files

**-t, --touch**  
Only touch files, faking update

**--shadow-prefix** <shadow\_prefix>  
Directory to place data for shadowed rules

**-r, --reason**  
Print reason for executing rule

**-N, --nohup**  
Don't die once the terminal goes away.

**-j, --cores** <CORES>  
The number of parallel threads used for scheduling jobs

**--dag**  
Print the Snakemake execution DAG and exit

**--rulegraph**  
Print the Snakemake rule graph and exit

**--debug-dag**  
Show candidates and selections made while the rule execution graph is being built

**--debug**  
Set the Snakemake debug flag

## Arguments

### **TARGET\_FILES**

Optional argument(s)

## 4.1.4 show

Show configuration properties

`ymp show [OPTIONS] PROPERTY`

## Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file** <log\_file>  
Specify a log file
- h, --help**
- s, --source**  
Show source

## Arguments

- PROPERTY**  
Optional argument

### 4.1.5 stage

Manipulate YMP stages

```
ymp stage [OPTIONS] COMMAND [ARGS]...
```

## Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file** <log\_file>  
Specify a log file

## list

List available stages

```
ymp stage list [OPTIONS] STAGE
```



## Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file <log\_file>**  
Specify a log file
- l, --long**  
Show full stage descriptions
- s, --short**  
Show only stage names
- c, --code**  
Show definition file name and line number
- t, --types**  
Show input/output types

## Arguments

- STAGE**  
Optional argument(s)

### 4.1.6 submit

Build target(s) on cluster

The parameters for cluster execution are drawn from layered profiles. YMP includes base profiles for the “torque” and “slurm” cluster engines.

```
ymp submit [OPTIONS] TARGET_FILES
```

## Options

- P, --pdb**  
Drop into debugger on uncaught exception
- q, --quiet**  
Decrease log verbosity
- v, --verbose**  
Increase log verbosity
- log-file <log\_file>**  
Specify a log file
- n, --dryrun**  
Only show what would be done

**-p, --printshellcmds**  
Print shell commands to be executed on shell

**-k, --keepgoing**  
Don't stop after failed job

**--lock, --no-lock**  
Use/don't use locking to prevent clobbering of files by parallel instances of YMP running

**--rerun-incomplete, --ri**  
Re-run jobs left incomplete in last run

**-F, --forceall**  
Force rebuilding of all stages leading to target

**-f, --force**  
Force rebuilding of target

**--notemp**  
Do not remove temporary files

**-t, --touch**  
Only touch files, faking update

**--shadow-prefix <shadow\_prefix>**  
Directory to place data for shadowed rules

**-r, --reason**  
Print reason for executing rule

**-N, --nohup**  
Don't die once the terminal goes away.

**-P, --profile <NAME>**  
Select cluster config profile to use. Overrides cluster.profile setting from config.

**-c, --snake-config <FILE>**  
Provide snakemake cluster config file

**-d, --drmaa**  
Use DRMAA to submit jobs to cluster. Note: Make sure you have a working DRMAA library. Set DRMAA\_LIBRARY\_PATH if necessary.

**-s, --sync**  
Use synchronous cluster submission, keeping the submit command running until the job has completed. Adds qsub\_sync\_arg to cluster command

**-i, --immediate**  
Use immediate submission, submitting all jobs to the cluster at once.

**--command <CMD>**  
Use CMD to submit job script to the cluster

**--wrapper <CMD>**  
Use CMD as script submitted to the cluster. See Snakemake documentation for more information.

**--max-jobs-per-second <N>**  
Limit the number of jobs submitted per second

**-l, --latency-wait <T>**  
Time in seconds to wait after job completed until files are expected to have appeared in local file system view. On NFS, this time is governed by the acdirmax mount option, which defaults to 60 seconds.

- J, --cluster-cores** <N>  
Limit the maximum number of cores used by jobs submitted at a time
- j, --cores** <N>  
Number of local threads to use
- args** <ARGS>  
Additional arguments passed to cluster submission command. Note: Make sure the first character of the argument is not '-', prefix with ' ' as necessary.
- scriptname** <NAME>  
Set the name template used for submitted jobs

## Arguments

### **TARGET\_FILES**

Optional argument(s)



## STAGES

Listing of stages implemented in YMP

Error in “sm:stage” directive: 1 argument(s) required, 0 supplied.

```
.. sm:stage::  
  :source: ymp/rules/00_import.rules:64  
  
  Imports raw read files into YMP.  
  
>>> ymp make toy  
>>> ymp make mpic
```

### **stage annotate\_blast**

Annotate sequences with BLAST

Searches a reference database for hits with `blastn`. Use `E` flag to specify exponent to required E-value. Use `N` or `Mega` to specify default. Use `Best` to add `-subject_besthit` flag.

### **stage annotate\_diamond**

FIXME

### **stage annotate\_prodigal**

Call genes using prodigal

```
>>> ymp make toy.ref_genome.annotate_prodigal
```

### **stage annotate\_tblastn**

Runs `tblastn`

### **stage assemble\_megahit**

Assemble metagenome using MegaHit.

```
>>> ymp make toy.assemble_megahit.map_bbmap  
>>> ymp make toy.group_ALL.assemble_megahit.map_bbmap  
>>> ymp make toy.group_Subject.assemble_megahit.map_bbmap
```

### **stage assemble\_metaspades**

Assemble reads using metaspades

```
>>> ymp make toy.assemble_metaspades  
>>> ymp make toy.group_ALL.assemble_metaspades  
>>> ymp make toy.group_Subject.assemble_metaspades
```

### **stage assemble\_trinity**

### **stage bin\_metabat2**

Bin metagenome assembly into MAGs

### **stage check**

Verify file availability

This stage provides rules for checking the file availability at a given point in the stage stack.

Mainly useful for testing and debugging.

### **stage cluster\_cdhit**

Clusters protein sequences using CD-HIT

```
>>> ymp make toy.ref_query.cluster_cdhit
```

### **stage correct\_bbmap**

Correct read errors by overlapping inside tails

Applies BMap's "bbmerge.sh ecco" mode. This will overlap the inside of read pairs and choose the base with the higher quality where the alignment contains mismatches and increase the quality score as indicated by the double observation where the alignment contains matches.

```
>>> ymp make toy.correct_bbmap
>>> ymp make mpic.correct_bbmap
```

### **stage count\_diamond**

### **stage count\_stringtie**

### **stage coverage\_samtools**

Computes coverage from a sorted bam file using `samtools coverage`

### **stage dedup\_bbmap**

Remove duplicate reads

Applies BMap's "dedupe.sh"

```
>>> ymp make toy.dedup_bbmap
>>> ymp make mpic.dedup_bbmap
```

### **stage dust\_bbmap**

Perform entropy filtering on reads using BMap's `bbduk.sh`

The parameter `Enn` gives the entropy cutoff. Higher values filter more sequences.

```
>>> ymp make toy.dust_bbmap
>>> ymp make toy.dust_bbmapE60
```

### **stage extract\_reads**

Extract reads from BAM file using `samtools fastq`.

Parameters `fn`, `Fn` and `Gn` are passed through. Some options include:

- `f2`: fully mapped (only proper pairs)
- `F2`: not fully mapped (unmapped at least one read)
- `f12`: not mapped (neither read mapped)

### **stage extract\_seqs**

Extract sequences from `.fasta.gz` file using `samtools faidx`

Currently requires a `.blast7` file as input.

Use parameter `Nomatch` to instead keep unmatched sequences.

**stage filter\_bmtagger**

Filter(-out) contaminant reads using BMTagger

```
>>> ymp make toy.ref_phiX.index_bmtagger.remove_bmtagger
>>> ymp make toy.ref_phiX.index_bmtagger.remove_bmtagger.assemble_megahit
>>> ymp make toy.ref_phiX.index_bmtagger.filter_bmtagger
>>> ymp make mpic.ref_phiX.index_bmtagger.remove_bmtagger
```

**stage format\_bbmap**

Process sequences with BMap's format.sh

Parameter `Ln` filters sequences at a minimum length.

```
>>> ymp make toy.assemble_metaspades.format_bbmapL200
```

**stage humann2**

Compute functional profiles using HUMAnN2

**stage index\_bbmap**

```
>>> ymp make toy.ref_genome.index_bbmap
```

**stage index\_blast****stage index\_bmtagger****stage index\_bowtie2**

```
>>> ymp make toy.ref_genome.index_bowtie2
```

**stage index\_diamond****stage map\_bbmap**

Map reads using BMap

```
>>> ymp make toy.assemble_megahit.map_bbmap
>>> ymp make toy.ref_genome.map_bbmap
>>> ymp make mpic.ref_ssu.map_bbmap
```

**stage map\_bowtie2**

Map reads using Bowtie2

```
>>> ymp make toy.ref_genome.index_bowtie2.map_bowtie2
>>> ymp make toy.assemble_megahit.index_bowtie2.map_bowtie2
>>> ymp make toy.group_Subject.assemble_megahit.index_bowtie2.map_bowtie2
>>> ymp make mpic.ref_ssu.index_bowtie2.map_bowtie2
```

**stage map\_diamond****stage map\_hisat2**

Map reads using Hisat2

**stage map\_star**

Map RNA-Seq reads with STAR

**stage metaphlan2**

Assess metagenome community composition using Metaphlan 2

**stage primermatch\_bimap**

Filters reads by matching reference primer

```
>>> ymp make mpic.ref_primers.primermatch_bimap
```

**stage profile\_centrifuge**

Classify reads using centrifuge

**stage qc\_fastqc**

Quality screen reads using FastQC

```
>>> ymp make toy.qc_fastqc
```

**stage qc\_multiqc**

Aggregate QC reports using MultiQC

**stage qc\_quast**

Estimate assembly quality using Quast

**stage quant\_rsem**

Quantify transcripts using RSEM

**stage references**

This is a “virtual” stage. It does not process read data, but comprises rules used for reference provisioning.

**stage remove\_bimap**

Filter reads by reference

This stage aligns the reads with a given reference using BMap in fast mode. Matching reads are collected in the stage *filter\_bimap* and remaining reads are collected in the stage *remove\_bimap*.

```
>>> ymp make toy.ref_phiX.index_bimap.remove_bimap
>>> ymp make toy.ref_phiX.index_bimap.filter_bimap
>>> ymp make mpic.ref_phiX.index_bimap.remove_bimap
```

**stage sort\_bam****stage split\_library**

Demultiplexes amplicon sequencing files

This rule is treated specially. If a configured project specifies a `barcode_col`, reads from the file (or files) are used in combination with

**stage trim\_bimap**

Trim adapters and low quality bases from reads

Applies BMap’s “bbduk.sh”.

**Parameters:** A: append to enable adapter trimming Q20: append to select phred score cutoff (default 20) L20: append to select minimum read length (default 20)

```
>>> ymp make toy.trim_bimap
>>> ymp make toy.trim_bimapA
>>> ymp make toy.trim_bimapAQ10L10
>>> ymp make mpic.trim_bimap
```

**stage trim\_sickle**

Perform read trimming using Sickle



```
>>> ymp make toy.trim_sickle
>>> ymp make toy.trim_sickleQ10L10
>>> ymp make mpic.trim_sickleL20
```

**stage trim\_trimmomatic**

Adapter trim reads using trimmomatic

```
>>> ymp make toy.trim_trimmomaticT32
>>> ymp make mpic.trim_trimmomatic
```

**rule download\_file\_ftp**

Downloads remote file using *wget*

**rule download\_file\_http**

Downloads remote file using internal downloader

**rule prefetch**

Downloads SRA files into NCBI SRA folder (ncbi/public/sra).

**rule fastq\_dump**

Extracts FQ from SRA files

**rule combine\_with\_ref****rule align\_mafft****rule blast7\_merge**

Merges blast results from all samples into single file

**rule blast7\_extract**

Generates meta-data csv and sequence fasta pair from blast7 file for one gene.

**rule blast7\_extract\_merge**

Merges extracted csv/fasta pairs over all samples.

**rule blast7\_all****rule blast7\_reports****rule blast7\_eval\_hist****rule blast7\_eval\_plot****rule cdhit\_fna\_single**

Clustering predicted genes (nuc) using cdhit-est

**rule 87****rule 88****rule 89****rule 90****rule 91****rule 92****rule faa\_fastp****rule fasta\_to\_fastp\_gz****rule gunzip**

Generic temporary gunzip

Use `ruleorder: gunzip > myrule` to prefer gunzipping over re-running a rule. E.g.

```
>>> ruleorder: gunzip > myrule
>>> rule myrule:
>>>   output: temp("some.txt"), "some.txt.gzip"
```

**rule mkdir**

Auto-create directories listed in ymp config.

Use these as input: `>>> input: tmpdir = ancient(icfg.dir.tmp)`

**rule fq2fa**

Unzip and convert fastq to fasta

**rule make\_otu\_table****rule otu\_to\_qiime\_txt****rule otu\_to\_biom****rule blast7\_coverage\_per\_otu****rule pick\_open\_otus**

Pick open reference OTUs

**rule pick\_closed\_otus**

Pick closed reference OTUs

**rule rarefy\_table****rule convert\_to\_closed\_ref**

Convert open reference otu table to closed reference

**rule env\_wait****rule ticktock****rule noop****rule normalize\_16S**

Normalize 16S by copy number using picrust, must be run with closed reference OTU table

**rule predict\_metagenome**

Predict metagenome using picrust

**rule categorize\_by\_function**

Categorize PICRUSt KOs into pathways

**rule raxml\_tree****rule rsem\_index**

Build Genome Index for RSEM

**rule scnic\_within\_minsamp****rule scnic\_within\_sparcc\_filter****rule star\_index**

Build Genome Index for Star

## 6.1 ymp package

`ymp.get_config()`

Access the current YMP configuration object.

This object might change once during normal execution: it is deleted before passing control to Snakemake. During unit test execution the object is deleted between all tests.

**Return type** *ConfigMgr*

`ymp.print_rule = 0`

Set to 1 to show the YMP expansion process as it is applied to the next Snakemake rule definition.

```
>>> ymp.print_rule = 1
>>> rule broken:
>>> ...
```

```
>>> ymp make broken -vvv
```

`ymp.snakemake_versions = ['5.20.1']`

List of versions this version of YMP has been verified to work with

### 6.1.1 Subpackages

#### **ymp.cli package**

`ymp.cli.install_completion(ctx, attr, value)`

Installs click\_completion tab expansion into users shell

`ymp.cli.install_profiler(ctx, attr, value)`

## Submodules

### ymp.cli.env module

`ymp.cli.env.get_env(envname)`

Get single environment matching glob pattern

**Parameters** `envname` – environment glob pattern

`ymp.cli.env.get_envs(patterns=None)`

Get environments matching glob pattern

**Parameters** `envnames` – list of strings to match

### ymp.cli.init module

Implements subcommands for `ymp init`

`ymp.cli.init.have_command(cmd)`

### ymp.cli.make module

Implements subcommands for `ymp make` and `ymp submit`

**class** `ymp.cli.make.TargetParam`

Bases: `click.types.ParamType`

Handles tab expansion for build targets

**classmethod** `complete(ctx, incomplete)`

Try to complete incomplete command

This is executed on tab or tab-tab from the shell

**Parameters**

- `ctx` – click context object
- `incomplete` – last word in command line up until cursor

**Returns** list of words incomplete can be completed to

**exception** `ymp.cli.make.YmpConfigNotFound`

Bases: `ymp.exceptions.YmpException`

Exception raised by YMP if no config was found in current path

`ymp.cli.make.debug(msg, *args, **kwargs)`

`ymp.cli.make.snake_params(func)`

Default parameters for subcommands launching Snakemake

`ymp.cli.make.start_snakemake(kwargs)`

Execute Snakemake with given parameters and targets

Fixes paths of `kwargs['targets']` to be relative to YMP root.

**ymp.cli.shared\_options module**

**class** `ymp.cli.shared_options.Group` (*name=None, commands=None, \*\*attrs*)

Bases: `click.core.Group`

**command** (*\*args, \*\*kwargs*)

A shortcut decorator for declaring and attaching a command to the group. This takes the same arguments as `command()` but immediately registers the created command with this instance by calling into `add_command()`.

**class** `ymp.cli.shared_options.Log`

Bases: `object`

Set up Logging

**classmethod** `logfile_option` (*ctx, param, val*)

**mod\_level** (*n*)

**classmethod** `quiet_option` (*ctx, param, val*)

**static** `set_logfile` (*filename*)

**classmethod** `verbose_option` (*ctx, param, val*)

**class** `ymp.cli.shared_options.LogFormatter`

Bases: `coloredlogs.ColoredFormatter`

Initialize a `ColoredFormatter` object.

#### Parameters

- **fmt** – A log format string (defaults to `DEFAULT_LOG_FORMAT`).
- **datefmt** – A date/time format string (defaults to `None`, but see the documentation of `BasicFormatter.formatTime()`).
- **style** – One of the characters `%`, `{` or `$` (defaults to `DEFAULT_FORMAT_STYLE`)
- **level\_styles** – A dictionary with custom level styles (defaults to `DEFAULT_LEVEL_STYLES`).
- **field\_styles** – A dictionary with custom field styles (defaults to `DEFAULT_FIELD_STYLES`).

**Raises** Refer to `check_style()`.

This initializer uses `colorize_format()` to inject ANSI escape sequences in the log format string before it is passed to the initializer of the base class.

**format** (*record*)

Apply level-specific styling to log records.

**Parameters** **record** – A `LogRecord` object.

**Returns** The result of `logging.Formatter.format()`.

This method injects ANSI escape sequences that are specific to the level of each log record (because such logic cannot be expressed in the syntax of a log format string). It works by making a copy of the log record, changing the `msg` field inside the copy and passing the copy into the `format()` method of the base class.

```
snakemake_level_styles = {'critical': {'color': 'red'}, 'debug': {'color': 'blue'}}
```

**class** `ymp.cli.shared_options.TqdmHandler` (*stream=None*)

Bases: `logging.StreamHandler`

Tqdm aware logging StreamHandler

Passes all log writes through tqdm to allow progress bars and log messages to coexist without clobbering terminal

Initialize the handler.

If stream is not specified, sys.stderr is used.

**emit** (*record*)

Emit a record.

If a formatter is specified, it is used to format the record. The record is then written to the stream with a trailing newline. If exception information is present, it is formatted using `traceback.print_exception` and appended to the stream. If the stream has an `'encoding'` attribute, it is used to determine how to do the output to the stream.

`ymp.cli.shared_options.command(*args, **kwargs)`

`ymp.cli.shared_options.enable_debug(ctx, param, val)`

`ymp.cli.shared_options.group(*args, **kwargs)`

`ymp.cli.shared_options.log_options(f)`

`ymp.cli.shared_options.nohup(ctx, param, val)`

Make YMP continue after the shell dies.

- redirects stdout and stderr into pipes and sub process that won't die if it can't write to either anymore
- closes stdin

## ymp.cli.show module

Implements subcommands for `ymp show`

**class** `ymp.cli.show.ConfigPropertyParams`

Bases: `click.types.ParamType`

Handles tab expansion for `ymp show` arguments

**complete** (*\_ctx, incomplete*)

Try to complete incomplete command

This is executed on tab or tab-tab from the shell

### Parameters

- **ctx** – click context object
- **incomplete** – last word in command line up until cursor

**Returns** list of words incomplete can be completed to

**convert** (*value, param, ctx*)

Convert value of param given context

### Parameters

- **value** – string passed on command line
- **param** – click parameter object
- **ctx** – click context object

**property properties**

Find properties offered by ConfigMgr

```
ymp.cli.show.show_help(ctx, _param=None, value=True)
```

Display click command help

**ymp.cli.stage module**

```
ymp.cli.stage.wrap(header, data)
```

**ymp.stage package**

YMP processes data in stages, each of which is contained in its own directory.

```
with Stage("trim_bimap") as S:
    S.doc("Trim reads with BMap")
    rule bimap_trim:
        output: "{:this:}/{sample}{:pairnames:}.fq.gz"
        input:  "{:prev:}/{sample}{:pairnames:}.fq.gz"
        ...
```

**Submodules****ymp.stage.base module**

```
class ymp.stage.base.BaseStage(name)
```

Bases: `object`

Base class for stage types

```
STAMP_FILENAME = 'all_targets.stamp'
```

The name of the stamp file that is touched to indicate completion of the stage.

```
can_provide(inputs)
```

Determines which of `inputs` this stage can provide.

Returns a dictionary with the keys a subset of `inputs` and the values identifying redirections. An empty string indicates that no redirection is to take place. Otherwise, the string is the suffix to be appended to the prior `StageStack`.

**Return type** `Dict[str, str]`

```
doc(doc)
```

Add documentation to Stage

**Parameters** `doc` (`str`) – Docstring passed to Sphinx

**Return type** `None`

```
docstring: str
```

The docstring describing this stage. Visible via `ymp stage list` and in the generated sphinx documentation.

```
get_all_targets(stack)
```

Targets to build to complete this stage given `stack`.

Typically, this is the `StageStack`'s path appended with the stamp name.

**Return type** `List[str]`

**get\_inputs()**

Returns the set of inputs required by this stage

This function must return a copy, to ensure internal data is not modified.

**Return type** `Set[str]`

**get\_path(stack)**

On disk location for this stage given `stack`.

Called by `StageStack` to determine the real path for virtual stages (which must override this function).

**Return type** `str`

**match(name)**

Check if the `name` can refer to this stage

As component of a `StageStack`, a stage may be identified by alternative names and may also be parametrized by suffix modifiers. Stage types supporting this behavior must override this function.

**Return type** `bool`

**name**

The name of the stage is a string uniquely identifying it among all stages.

**property outputs**

Returns the set of outputs this stage is able to generate.

May return either a `set` or a `dict` with the dictionary values representing redirections in the case of virtual stages such as `Pipeline` or `Reference`.

**Return type** `Union[Set[str], Dict[str, str]]`

**class** `ymp.stage.base.ConfigStage(name, cfg)`

Bases: `ymp.stage.base.BaseStage`

Base for stages created via configuration

These Stages derive from the `yml.yml` and not from a rules file.

**cfg**

The configuration object defining this Stage.

**property defined\_in**

List of files defining this stage

Used to invalidate caches.

**filename**

Semi-colon separated list of file names defining this Stage.

**lineno**

Line number within the first file at which this Stage is defined.



## ymp.stage.expander module

**class** `ymp.stage.expander.StageExpander`

Bases: `ymp.snakemake.ColonExpander`

- Registers rules with stages when they are created

**class** `Formatter(expander)`

Bases: `ymp.snakemake.FormatExpander.Formatter`, `ymp.string.PartialFormatter`

**get\_value**(*key*, *args*, *kwargs*)

**get\_value\_**(*key*, *args*, *kwargs*)

**expand\_ruleinfo**(*rule*, *item*, *expand\_args*, *rec*)

**expand\_str**(*rule*, *item*, *expand\_args*, *rec*, *cb*)

**expands\_field**(*field*)

Checks if this expander should expand a Rule field type

**Parameters** *field* – the field to check

**Returns** True if *field* should be expanded.

## ymp.stage.groupby module

**class** `ymp.stage.groupby.GroupBy(name)`

Bases: `ymp.stage.base.BaseStage`

Dummy stage for grouping

## ymp.stage.pipeline module

Pipelines Module

Contains classes for pre-configured pipelines comprising multiple stages.

**class** `ymp.stage.pipeline.Pipeline(name, cfg)`

Bases: `ymp.stage.base.ConfigStage`

A virtual stage aggregating a sequence of stages, i.e. a pipeline or sub-workflow.

Pipelines are configured via `ymp.yml`.

### Example

**pipelines:**

**my\_pipeline:**

- stage\_1
- stage\_2
- stage\_3

**can\_provide** (*inputs*)

Determines which of *inputs* this stage can provide.

The result dictionary values will point to the “real” output.

**Return type** `Dict[str, str]`

**get\_all\_targets** (*stack*)

Targets to build to complete this stage given *stack*.

Typically, this is the StageStack’s path appended with the stamp name.

**get\_path** (*stack*)

On disk location for this stage given *stack*.

Called by *StageStack* to determine the real path for virtual stages (which must override this function).

**property outputs**

The outputs of a pipeline are the sum of the outputs of each component stage. Outputs of stages further down the pipeline override those generated earlier.

TODO: Allow hiding the output of intermediary stages.

**Return type** `Dict[str, str]`

**property pipeline**

## ymp.stage.project module

**class** `ymp.stage.project.PandasProjectData` (*cfg*)

Bases: `object`

**column** (*col*)

**columns** ()

**dump** ()

**duplicate\_rows** (*column*)

**get** (*idcol*, *row*, *col*)

**groupby\_dedup** (*cols*)

Return non-redundant identifying subset of cols

**identifying\_columns** ()

**rows** (*cols*)

**string\_columns** ()

**class** `ymp.stage.project.PandasTableBuilder`

Bases: `object`

Builds the data table describing each sample in a project

This class implements loading and combining tabular data files as specified by the YAML configuration.

**Format:**

- string items are files
- lists of files are concatenated top to bottom
- dicts must have one “command” value:

- 'join' contains a two-item list the two items are joined 'naturally' on shared headers
  - 'table' contains a list of one-item dicts dicts have form `key:value[, value...]` a in-place table is created from the keys list-of-dict is necessary as dicts are unordered
  - 'paste' contains a list of tables pasted left to right tables pasted must be of equal length or length 1
- if a value is a valid path relative to the csv/tsv/xls file's location, it is expanded to a path relative to CWD

### Example

```
- top.csv
- join:
  - excel.xlsx%left.csv
  - right.tsv
- table:
  - sample: s1,s2,s3
  - fq1: s1.1.fq, s2.1.fq, s3.1.fq
  - fq2: s1.2.fq, s2.2.fq, s3.2.fq
```

`load_data(cfg)`

**class** `ymp.stage.project.Project(name, cfg)`

Bases: `ymp.stage.base.ConfigStage`

Contains configuration for a source dataset to be processed

**KEY\_BCCOL** = 'barcode\_col'

**KEY\_DATA** = 'data'

**KEY\_IDCOL** = 'id\_col'

**KEY\_READCOLS** = 'read\_cols'

**RE\_FILE** = `re.compile('^(?!http://).*?(?:fq|fastq)(?:|\\.gz)$')`

**RE\_REMOTE** = `re.compile('^(?:https?|ftp|sftp)://(?:.*)')`

**RE\_SRR** = `re.compile('^([SED]RR[0-9]+)$')`

**choose\_fq\_columns()**

Configures the columns referencing the fastq sources

**choose\_id\_column()**

Configures column to use as index on runs

If explicitly configured via **KEY\_IDCOL**, verifies that the column exists and that it is unique. Otherwise chooses the leftmost unique column in the data.

**property data**

Pandas dataframe of runs

Lazy loading property, first call may take a while.

**encode\_barcode\_path(barcode\_file, run, pair)**

**property fq\_names**

Names of all FastQ files

**property fwd\_fq\_names**

Names of forward FastQ files (se and pe)

**property fwd\_pe\_fq\_names**

Names of forward FastQ files part of pair

**get\_fq\_names** (*only\_fwd=False, only\_rev=False, only\_pe=False, only\_se=False*)

Get pipeline names of fq files

**get\_ids** (*groups, match\_groups=None, match\_value=None*)

**property idcol**

**iter\_samples** (*variables=None*)

**minimize\_variables** (*groups*)

**property outputs**

Returns the set of outputs this stage is able to generate.

May return either a [set](#) or a [dict](#) with the dictionary values representing redirections in the case of virtual stages such as [Pipeline](#) or [Reference](#).

**Return type** [Union](#)[[Set](#)[[str](#)], [Dict](#)[[str](#), [str](#)]]

**property pe\_fq\_names**

Names of paired end FastQ files

**property project\_name**

**raw\_reads\_source\_path** (*args, kwargs*)

**property rev\_pe\_fq\_names**

Names of reverse FastQ files part of pair

**property runs**

Pandas dataframe index of runs

Lazy loading property, first call may take a while.

**property se\_fq\_names**

Names of single end FastQ files

**property source\_cfg**

**source\_path** (*target, pair, nosplit=False*)

Get path for FQ file for run and pair

**unsplit\_path** (*barcode\_id, pairname*)

**property variables**

**class** `ymp.stage.project.SQLiteProjectData` (*cfg, name='data'*)

Bases: [object](#)

**column** (*col*)

**columns** ()

**property db\_url**

**dump** ()

**duplicate\_rows** (*column*)

**get** (*idcol, row, col*)

**groupby\_dedup** (*cols*)

**identifying\_columns** ()

```

property nrows
query (*args)
rows (col)
string_columns ()

```

### ymp.stage.reference module

**class** `ymp.stage.reference.Archive` (*name, dirname, tar, url, strip, files*)

Bases: `object`

```

dirname = None
files = None
get_files ()
hash = None
make_unpack_rule (baserule)
name = None
strip_components = None
tar = None

```

**class** `ymp.stage.reference.Reference` (*name, cfg*)

Bases: `ymp.stage.base.ConfigStage`

Represents (remote) reference file/database configuration

```

add_files (rsc, local_path)
get_file (filename)
get_path (_stack)

```

On disk location for this stage given `stack`.

Called by `StageStack` to determine the real path for virtual stages (which must override this function).

```
make_unpack_rules (baserule)
```

**property outputs**

Returns the set of outputs this stage is able to generate.

May return either a `set` or a `dict` with the dictionary values representing redirections in the case of virtual stages such as `Pipeline` or `Reference`.

**Return type** `Union[Set[str], Dict[str, str]]`

## ymp.stage.stack module

**class** `ymp.stage.stack.StageStack` (*path*, *stage=None*)

Bases: `object`

The “head” of a processing chain - a stack of stages

**all\_targets** ()

**complete** (*incomplete*)

**property** **defined\_in**

**classmethod** **get** (*path*, *stage=None*)

Cached access to StageStack

### Parameters

- **path** – Stage path
- **stage** – Stage object at head of stack

**property** **path**

On disk location of files provided by this stack

**prev** (*args=None*, *kwargs=None*)

Directory of previous stage

**resolve\_prevs** ()

**target** (*args*, *kwargs*)

Finds the target in the prev stage matching current target

**property** **targets**

Returns the current targets

**used\_stacks** = {}

`ymp.stage.stack.find_stage` (*name*)

`ymp.stage.stack.norm_wildcards` (*pattern*)

## ymp.stage.stage module

**class** `ymp.stage.stage.Param` (*stage*, *key*, *name*, *value=None*, *default=None*)

Bases: `object`

Stage Parameter (base class)

**property** **constraint**

**pattern** (*show\_constraint=True*)

String to add to filenames passed to Snakemake

I.e. a pattern of the form {wildcard, constraint}

**class** `ymp.stage.stage.ParamChoice` (*\*args*, *\*\*kwargs*)

Bases: `ymp.stage.stage.Param`

Stage Choice Parameter

**param\_func** ()

Returns function that will extract parameter value from wildcards

```
class ymp.stage.stage.ParamFlag(*args, **kwargs)
```

Bases: `ymp.stage.stage.Param`

Stage Flag Parameter

```
param_func()
```

Returns function that will extract parameter value from wildcards

```
class ymp.stage.stage.ParamInt(*args, **kwargs)
```

Bases: `ymp.stage.stage.Param`

Stage Int Parameter

```
param_func()
```

Returns function that will extract parameter value from wildcards

```
class ymp.stage.stage.Stage(name, altname=None, env=None, doc=None)
```

Bases: `ymp.snakemake.WorkflowObject`, `ymp.stage.base.BaseStage`

Creates a new stage

While entered using `with`, several stage specific variables are expanded within rules:

- `{:this:}` – The current stage directory
- `{:that:}` – The alternate output stage directory
- `{:prev:}` – The previous stage’s directory

#### Parameters

- **name** (`str`) – Name of this stage
- **altname** (`Optional[str]`) – Alternate name of this stage (used for stages with multiple output variants, e.g. `filter_x` and `remove_x`).
- **doc** (`Optional[str]`) – See `Stage.doc`
- **env** (`Optional[str]`) – See `Stage.env`

```
active = None
```

Currently active stage (“entered”)

```
add_param(key, typ, name, value=None, default=None)
```

Add parameter to stage

#### Example

```
>>> with Stage("test") as S
>>> S.add_param("N", "int", "nval", default=50)
>>> rule:
>>>     shell: "echo {param.nval}"
```

This would add a stage “test”, optionally callable as “testN123”, printing “50” or in the case of “testN123” printing “123”.

#### Parameters

- **char** – The character to use in the Stage name
- **typ** – The type of the parameter (int, flag)
- **param** – Name of parameter in params

- **value** – value `{param.xyz}` should be set to if param given
- **default** – default value for `{{param.xyz}}` if no param given

**env** (*name*)

Add package specifications to Stage environment

---

**Note:** This sets the environment for all rules within the stage, which leads to errors with Snakemake rule types not supporting conda environments

---

**Parameters** **name** (*str*) – Environment name or filename

```
>>> Env("blast", packages="blast =2.7*")
>>> with Stage("test") as S:
>>>     S.env("blast")
>>>     rule testing:
>>>         ...
```

```
>>> with Stage("test", env="blast") as S:
>>>     rule testing:
>>>         ...
```

```
>>> with Stage("test") as S:
>>>     rule testing:
>>>         conda: "blast"
>>>         ...
```

**Return type** None

**get\_inputs** ()

Returns the set of inputs required by this stage

This function must return a copy, to ensure internal data is not modified.

**match** (*name*)

Check if the *name* can refer to this stage

As component of a *StageStack*, a stage may be identified by alternative names and may also be parametrized by suffix modifiers. Stage types supporting this behavior must override this function.

**property outputs**

Returns the set of outputs this stage is able to generate.

May return either a *set* or a *dict* with the dictionary values representing redirections in the case of virtual stages such as *Pipeline* or *Reference*.

**prev** (*args*, *kwargs*)

Gathers `{:prev:}` calls from rules

**require** (*\*\*kwargs*)

Override inferred stage inputs

In theory, this should not be needed. But it's simpler for now.

**satisfy\_inputs** (*other\_stage*, *inputs*)

**Return type** *Dict*[*str*, *str*]



**that** (*args=None, kwargs=None*)  
 Alternate directory of current stage

Used for splitting stages

**this** (*args=None, kwargs=None*)  
 Directory of current stage

**wc2path** (*wc*)

**wildcards** (*args=None, kwargs=None*)

`ymp.stage.stage.norm_wildcards` (*pattern*)

## 6.1.2 Submodules

### 6.1.3 ymp.blast module

Parsers for blast output formats 6 (CSV) and 7 (CSV with comments between queries).

**class** `ymp.blast.BlastParser`

Bases: `object`

Base class for BLAST parsers

**FIELD\_MAP** = {'% identity': 'pident', 'alignment length': 'length', 'bit score': 'bi

**FIELD\_TYPE** = {'bitscore': <class 'float'>, 'evalue': <class 'float'>, 'gapopen': <c

**tupleofint** ()

**class** `ymp.blast.Fmt6Parser` (*fileobj*)

Bases: `ymp.blast.BlastParser`

Parser for BLAST format 6 (CSV)

**Hit**

alias of `BlastHit`

**field\_types** = [None, None, <class 'float'>, <class 'int'>, <class 'int'>, <class 'int'

**fields** = ['qseqid', 'sseqid', 'pident', 'length', 'mismatch', 'gapopen', 'qstart', 'qe  
 Default field types

**get\_fields** ()

**class** `ymp.blast.Fmt7Parser` (*fileobj*)

Bases: `ymp.blast.BlastParser`

Parses BLAST results in format '7' (CSV with comments)

**DATABASE** = '# Database: '

**FIELDS** = '# Fields: '

**HITSFOUND** = ' hits found'

**QUERY** = '# Query: '

**get\_fields** ()

Returns list of available field names

Format 7 specifies which columns it contains in comment lines, allowing this parser to be agnostic of the selection of columns made when running BLAST.

**Return type** `List[str]`

**Returns** List of field names (e.g. ['sacc', 'qacc', 'evaluate'])

**isfirsthit()**

Returns `True` if the current hit is the first hit for the current query

**Return type** `bool`

`ymp.blast.reader` (*fileobj*, *t=7*)

Creates a reader for files in BLAST format

```
>>> with open(blast_file) as infile:
>>>     reader = blast.reader(infile)
>>>     for hit in reader:
>>>         print(hit)
```

**Parameters**

- **fileobj** – iterable yielding lines in blast format
- **t** (`int`) – number of blast format type

**Return type** `BlastParser`

## 6.1.4 ymp.blast2gff module

## 6.1.5 ymp.cluster module

Module handling talking to cluster management systems

```
>>> python -m ymp.cluster slurm status <jobid>
```

**class** `ymp.cluster.ClusterMS`

Bases: `object`

**class** `ymp.cluster.Lsf`

Bases: `ymp.cluster.ClusterMS`

Talking to LSF

**states** = {'DONE': 'success', 'EXIT': 'failed', 'PEND': 'running', 'POST\_DONE': 'success'}

**static status** (*jobid*)

**static submit** (*args*)

**class** `ymp.cluster.Slurm`

Bases: `ymp.cluster.ClusterMS`

Talking to Slurm

**states** = {'BOOT\_FAIL': 'failed', 'CANCELLED': 'failed', 'COMPLETED': 'success', 'COMPL'

**static status** (*jobid*)

Print status of job @param jobid to stdout (as needed by snakemake)

Anectotal benchmarking shows 200ms per invocation, half used by Python startup and half by calling sacct. Using `scontrol show job` instead of `sacct -pbs` is faster by 80ms, but finished jobs are purged after unknown time window.

`ymp.cluster.error` (\*args, \*\*kwargs)

## 6.1.6 ymp.common module

Collection of shared utility classes and methods

**class** ymp.common.AttrDict

Bases: dict

AttrDict adds accessing stored keys as attributes to dict

**class** ymp.common.Cache (root)

Bases: object

**close** ()

**commit** ()

**get\_cache** (name, clean=False, \*args, \*\*kwargs)

**load** (cache, key)

**load\_all** (cache)

**store** (cache, key, obj)

**class** ymp.common.CacheDict (cache, name, \*args, loadfunc=None, itemloadfunc=None, itemdata=None, \*\*kwargs)

Bases: ymp.common.AttrDict

**get** (k[, d]) → D[k] if k in D, else d. d defaults to None.

**items** () → a set-like object providing a view on D's items

**keys** () → a set-like object providing a view on D's keys

**values** () → an object providing a view on D's values

**class** ymp.common.MkdirDict

Bases: ymp.common.AttrDict

Creates directories as they are requested

ymp.common.ensure\_list (obj)

Wrap obj in a list as needed

ymp.common.flatten (item)

Flatten lists without turning strings into letters

ymp.common.is\_container (obj)

Check if object is container, considering strings not containers

ymp.common.parse\_number (s="")

Basic 1k 1m 1g 1t parsing.

- assumes base 2
- returns “byte” value
- accepts “1kib”, “1kb” or “1k”

### 6.1.7 ymp.config module

**class** ymp.config.ConfigExpander(*config\_mgr*)

Bases: *ymp.snakemake.ColonExpander*

**class** Formatter(*expander*)

Bases: *ymp.snakemake.FormatExpander.Formatter*, *ymp.string.PartialFormatter*

**get\_value** (*field\_name*, *args*, *kwargs*)

**expands\_field** (*field*)

Checks if this expander should expand a Rule field type

**Parameters** *field* – the field to check

**Returns** True if *field* should be expanded.

**class** ymp.config.ConfigMgr(*root*, *conffiles*)

Bases: *object*

Manages workflow configuration

This is a singleton object of which only one instance should be around at a given time. It is available in the rules files as *icfg* and via *ymp.get\_config()* elsewhere.

ConfigMgr loads and maintains the workflow configuration as given in the *ymp.yml* files located in the workflow root directory, the user config folder (*~/.ymp*) and the installation *etc* folder.

**CONF\_DEFAULT\_FNAME** = *'/home/docs/checkouts/readthedocs.org/user\_builds/ymp/checkouts/stab'*

**CONF\_FNAME** = *'ymp.yml'*

**CONF\_USER\_FNAME** = *'/home/docs/.ymp/ymp.yml'*

**KEY\_PIPELINES** = *'pipelines'*

**KEY\_PROJECTS** = *'projects'*

**KEY\_REFERENCES** = *'references'*

**RULE\_MAIN\_FNAME** = *'/home/docs/checkouts/readthedocs.org/user\_builds/ymp/checkouts/stab'*

**property** *absdir*

Dictionary of absolute paths of named YMP directories

**classmethod** *activate()*

**property** *cluster*

The YMP cluster configuration.

**property** *conda*

**property** *dir*

Dictionary of relative paths of named YMP directories

The directory paths are relative to the YMP root workdir.

**property** *ensuredir*

Dictionary of absolute paths of named YMP directories

Directories will be created on the fly as they are requested.

**expand** (*item*, *\*\*kwargs*)

**classmethod find\_config()**

Locates ymp config files and ymp root

The root ymp work dir is determined as the first (parent) directory containing a file named `ConfigMgr.CONF_FNAME` (default `ymp.yml`).

The stack of config files comprises 1. the default config `ConfigMgr.CONF_DEFAULT_FNAME` (etc/defaults.yml in the ymp package directory), 2. the user config `ConfigMgr.CONF_USER_FNAME` (~/.ymp/ymp.yml) and 3. the `yml.yml` in the ymp root.

**Returns** Root working directory conffiles: list of active configuration files

**Return type** root

**classmethod instance()**

Returns the active Ymp ConfigMgr instance

**property limits**

The YMP limits configuration.

**mem** (*base='0', per\_thread=None, unit='m'*)

Clamp memory to configuration limits

**Params:** base: base memory requested per\_thread: additional mem required per allocated thread unit: output unit (b, k, m, g, t)

**property pairnames****property pipeline**

Configure pipelines

**property platform**

Name of current platform (macos or linux)

**property ref**

Configure references

**property shell**

The shell used by YMP

Change by adding e.g. `shell: /path/to/shell` to `ymp.yml`.

**property snakefiles**

Snakefiles used under this config in parsing order

**classmethod unload()****class ymp.config.OverrideExpander** (*cfgmgr*)

Bases: `ymp.snakemake.BaseExpander`

Apply rule attribute overrides from ymp.yml config

**Example**

Set the `wordsize` parameter in the `bmtagger_bitmask` rule to 12:

Listing 1: ymp.yml

```
overrides:
  rules:
    bmtagger_bitmask:
      params:
        wordsize: 12
```

**expand** (*rule*, *ruleinfo*, *\*\*kwargs*)

Expands RuleInfo object and children recursively.

Will call :meth:format (via :meth:format\_annotated) on `str` items encountered in the tree and wrap encountered functions to be called once the wildcards object is available.

Set `ymp.print_rule = 1` before a `rule:` statement in snakefiles to enable debug logging of recursion.

#### Parameters

- **rule** – The :class:snakemake.rules.Rule object to be populated with the data from the RuleInfo object passed from *item*
- **item** – The item to be expanded. Initially a :class:snakemake.workflow.RuleInfo object into which is recursively descendet. May ultimately be `None`, `str`, `function`, `int`, `float`, `dict`, `list` or `tuple`.
- **expand\_args** – Parameters passed on late expansion (when the dag tries to instantiate the *rule* into a job).
- **rec** – Recursion level

### 6.1.8 ymp.dna module

`ymp.dna.nuc2aa(seq)`

`ymp.dna.nuc2num(seq)`

### 6.1.9 ymp.download module

**class** `ymp.download.DownloadThread`

Bases: `object`

**get** (*url*, *dest*, *md5*)

**main** ()

**terminate** ()

**class** `ymp.download.FileDownloader` (*block\_size=4096*, *timeout=300*, *parallel=4*, *loglevel=30*, *alturls=None*, *retry=3*)

Bases: `object`

Manages download of a set of URLs

Downloads happen concurrently using asynchronous network IO.

#### Parameters

- **block\_size** (*int*) – Byte size of chunks to download
- **timeout** (*int*) – Aiohttp cumulative timeout
- **parallel** (*int*) – Number of files to download in parallel
- **loglevel** (*int*) – Log level for messages send to logging (Errors are send with `loglevel+10`)
- **alturls** – List of regexps modifying URLs
- **retry** (*int*) – Number of times to retry download

**error** (*msg*, \**args*, \*\**kwargs*)

Send error to logger

Message is sent with a log level 10 higher than the default for this object.

**Return type** None

**get** (*urls*, *dest*, *md5s=None*)

Download a list of URLs

**Parameters**

- **urls** (`Union[str, List[str]]`) – List of URLs
- **dest** (`str`) – Destination folder
- **md5s** (`Optional[List[str]]`) – List of MD5 sums to check

**Return type** None

**log** (*msg*, \**args*, *modlvl=0*, \*\**kwargs*)

Send message to logger

Honors loglevel set for the FileDownloader object.

**Parameters**

- **msg** (`str`) – The log message
- **modlvl** (`int`) – Added to default logging level for object

**Return type** None

**static make\_bar\_format** (*desc\_width=20*, *count\_width=0*, *rate=False*, *eta=False*,  
*have\_total=True*)

Construct bar\_format for tqdm

**Parameters**

- **desc\_width** (`int`) – minimum space allocated for description
- **count\_width** (`int`) – min space for counts
- **rate** (`bool`) – show rate to right of progress bar
- **eta** (`bool`) – show eta to right of progress bar
- **have\_total** (`bool`) – whether a total exists (required to add percentage)

**Return type** `str`

## 6.1.10 ymp.env module

This module manages the conda environments.

**class** `ymp.env.CondaPathExpander` (*config*, \**args*, \*\**kwargs*)

Bases: `ymp.snakemake.BaseExpander`

Applies search path for conda environment specifications

File names supplied via rule: `conda: "some.yml"` are replaced with absolute paths if they are found in any searched directory. Each `search_paths` entry is appended to the directory containing the top level Snakefile and the directory checked for the filename. Thereafter, the stack of including Snakefiles is traversed backwards. If no file is found, the original name is returned.

**expands\_field** (*field*)

Checks if this expander should expand a Rule field type

**Parameters** *field* – the field to check

**Returns** True if *field* should be expanded.

**format** (*conda\_env*, \**args*, \*\**kwargs*)

Format *item* using \**args* and \*\**kwargs*

```
class ymp.env.Env(env_file=None, dag=None, singularity_img=None, container_img=None,  
                  cleanup=None, name=None, packages=None, base='none', channels=None,  
                  rule=None)
```

Bases: `ymp.snakemake.WorkflowObject`, `snakemake.deployment.conda.Env`

Represents YMP conda environment

Snakemake expects the conda environments in a per-workflow directory configured by `conda_prefix`. YMP sets this value by default to `~/ .ymp/conda`, which has a greater chance of being on the same file system as the conda cache, allowing for hard linking of environment files.

Within the folder `conda_prefix`, each environment is created in a folder named by the hash of the environment definition file's contents and the `conda_prefix` path. This class inherits from `snakemake.deployment.conda.Env` to ensure that the hash we use is identical to the one Snakemake will use during workflow execution.

The class provides additional features for updating environments, creating environments dynamically and executing commands within those environments.

---

**Note:** This is not called from within the execution. Snakemake instantiates its own Env object purely based on the filename.

---

Creates an inline defined conda environment

**Parameters**

- **name** (`Optional[str]`) – Name of conda environment (and basename of file)
- **packages** (`Union[list, str, None]`) – package(s) to be installed into environment. Version constraints can be specified in each package string separated from the package name by whitespace. E.g. "blast =2.6\*"
- **channels** (`Union[list, str, None]`) – channel(s) to be selected for the environment
- **base** (`str`) – Select a set of default channels and packages to be added to the newly created environment. Sets are defined in `conda.defaults` in `yaml.yaml`

**create** (*dryrun=False*, *force=False*)

Ensure the conda environment has been created

Inherits from `snakemake.conda.Env.create`

**Behavior of super class** The environment is installed in a folder in `conda_prefix` named according to a hash of the `environment.yaml` defining the environment and the value of `conda-prefix` (`Env.hash`). The latter is included as installed environments cannot be moved.

- If this folder (`Env.path`) exists, nothing is done.
- If a folder named according to the hash of just the contents of `environment.yaml` exists, the environment is created by unpacking the tar balls in that folder.



**Handling pre-computed environment specs** In addition to freezing environments by maintaining a copy of the package binaries, we allow maintaining a copy of the package binary URLs, from which the archive folder is populated on demand.

If a file `{Env.name}.txt` exists in `conda.spec` **FIXME**

**export** (*stream*, *typ*='yml')

Freeze environment

**static get\_installed\_env\_hashes** ()

**property installed**

**run** (*command*)

Execute command in environment

Returns exit code of command run.

**set\_prefix** (*prefix*)

**update** ()

Update conda environment

## 6.1.11 ymp.exceptions module

Exceptions raised by YMP

**exception** `ymp.exceptions.YmpConfigError` (*obj*, *msg*, *key*=None, *exc*=None)

Bases: `ymp.exceptions.YmpNoStackException`

Indicates an error in the ymp.yml config files

**Parameters**

- **obj** (*object*) – Subtree of config causing error
- **msg** (*str*) – The message to display
- **key** (*object*) – Key indicating part of obj causing error
- **exc** (*Optional[Exception]*) – Upstream exception causing error

**exception** `ymp.exceptions.YmpException`

Bases: `Exception`

Base class of all YMP Exceptions

**exception** `ymp.exceptions.YmpNoStackException` (*message*)

Bases: `ymp.exceptions.YmpException`, `click.exceptions.ClickException`

Exception that does not lead to stack trace on CLI

Inheriting from `ClickException` makes `click` print only the `self.msg` value of the exception, rather than allowing Python to print a full stack trace.

This is useful for exceptions indicating usage or configuration errors. We use this, instead of `click.UsageError` and friends so that the exceptions can be caught and handled explicitly where needed.

Note that `click` will call the `show` method on this object to print the exception. The default implementation from `click` will just prefix the `msg` with `Error:`.

**FIXME: This does not work if the exception is raised from within** the `snakemake` workflow as `snake-make.snakemake` catches and reformats exceptions.

**exception** `ymp.exceptions.YmpRuleError(obj, msg)`

Bases: `ymp.exceptions.YmpNoStackException`

Indicates an error in the rules files

This could e.g. be a Stage or Environment defined twice.

**Parameters**

- **obj** (`object`) – The object causing the exception. Must have `lineno` and `filename` as these will be shown as part of the error message on the command line.
- **msg** (`str`) – The message to display

**show** ()

**Return type** `None`

**exception** `ymp.exceptions.YmpStageError(msg)`

Bases: `ymp.exceptions.YmpNoStackException`

Indicates an error in the requested stage stack

**show** ()

**Return type** `None`

**exception** `ymp.exceptions.YmpSystemError(message)`

Bases: `ymp.exceptions.YmpNoStackException`

Indicates problem running YMP with available system software

**exception** `ymp.exceptions.YmpUsageError(message)`

Bases: `ymp.exceptions.YmpNoStackException`

**exception** `ymp.exceptions.YmpWorkflowError(message)`

Bases: `ymp.exceptions.YmpNoStackException`

Indicates an error during workflow execution

E.g. failures to expand dynamic variables

## 6.1.12 ymp.gff module

Implements simple reader and writer for GFF (general feature format) files.

Unfinished

- only supports one version, GFF 3.2.3.
- no escaping

**class** `ymp.gff.Attributes(ID, Name, Alias, Parent, Target, Gap, Derives_From, Note, Dbxref, Ontology_term, Is_circular)`

Bases: `tuple`

Create new instance of `Attributes(ID, Name, Alias, Parent, Target, Gap, Derives_From, Note, Dbxref, Ontology_term, Is_circular)`

**property** `Alias`

Alias for field number 2

**property** `Dbxref`

Alias for field number 8

**property Derives\_From**

Alias for field number 6

**property Gap**

Alias for field number 5

**property ID**

Alias for field number 0

**property Is\_circular**

Alias for field number 10

**property Name**

Alias for field number 1

**property Note**

Alias for field number 7

**property Ontology\_term**

Alias for field number 9

**property Parent**

Alias for field number 3

**property Target**

Alias for field number 4

**class** ymp.gff.**Feature** (*seqid, source, type, start, end, score, strand, phase, attributes*)

Bases: `tuple`

Create new instance of Feature(seqid, source, type, start, end, score, strand, phase, attributes)

**property attributes**

Alias for field number 8

**property end**

Alias for field number 4

**property phase**

Alias for field number 7

**property score**

Alias for field number 5

**property seqid**

Alias for field number 0

**property source**

Alias for field number 1

**property start**

Alias for field number 3

**property strand**

Alias for field number 6

**property type**

Alias for field number 2

**class** ymp.gff.**reader** (*fileobj*)

Bases: `object`

**class** ymp.gff.**writer** (*fileobj*)

Bases: `object`

**write** (*feature*)

### 6.1.13 ymp.helpers module

This module contains helper functions.

Not all of these are currently in use

**class** ymp.helpers.**OrderedDictMaker**

Bases: `object`

odict creates OrderedDict objects in a dict-literal like syntax

```
>>> my_ordered_dict = odict[
>>>     'key': 'value'
>>> ]
```

Implementation: odict uses the python slice syntax which is similar to dict literals. The `[]` operator is implemented by overriding `__getitem__`. Slices passed to the operator as `object[start1:stop1:step1, start2:...]`, are passed to the implementation as a list of objects with start, stop and step members. odict simply creates an OrderedDict by iterating over that list.

ymp.helpers.**update\_dict** (*dst, src*)

Recursively update dictionary *dst* with *src*

- Treats a `list` as atomic, replacing it with new list.
- Dictionaries are overwritten by item
- None is replaced by empty dict if necessary

### 6.1.14 ymp.map2otu module

**class** ymp.map2otu.**MapfileParser** (*minid=0*)

Bases: `object`

**read** (*mapfiles*)

**write** (*outfile*)

**class** ymp.map2otu.**emirge\_info** (*line*)

Bases: `object`

ymp.map2otu.**main** ()

### 6.1.15 ymp.nuc2aa module

ymp.nuc2aa.**fasta\_dna2aa** (*inf, outf*)

ymp.nuc2aa.**nuc2aa** (*seq*)

ymp.nuc2aa.**nuc2num** (*seq*)

## 6.1.16 ymp.snakemake module

Extends Snakemake Features

**class** ymp.snakemake.BaseExpander

Bases: `object`

Base class for Snakemake expansion modules.

Subclasses should override the `:meth:expand` method if they need to work on the entire RuleInfo object or the `:meth:format` and `:meth:expands_field` methods if they intend to modify specific fields.

**expand** (*rule, item, expand\_args=None, rec=-1, cb=False*)

Expands RuleInfo object and children recursively.

Will call `:meth:format` (via `:meth:format_annotated`) on `str` items encountered in the tree and wrap encountered functions to be called once the wildcards object is available.

Set `ymp.print_rule = 1` before a `rule:` statement in snakefiles to enable debug logging of recursion.

### Parameters

- **rule** – The `:class:snakemake.rules.Rule` object to be populated with the data from the RuleInfo object passed from *item*
- **item** – The item to be expanded. Initially a `:class:snakemake.workflow.RuleInfo` object into which is recursively descendet. May ultimately be `None`, `str`, `function`, `int`, `float`, `dict`, `list` or `tuple`.
- **expand\_args** – Parameters passed on late expansion (when the dag tries to instantiate the *rule* into a job).
- **rec** – Recursion level

**expand\_dict** (*rule, item, expand\_args, rec*)

**expand\_func** (*rule, item, expand\_args, rec, debug*)

**expand\_list** (*rule, item, expand\_args, rec, cb*)

**expand\_ruleinfo** (*rule, item, expand\_args, rec*)

**expand\_str** (*rule, item, expand\_args, rec, cb*)

**expand\_tuple** (*rule, item, expand\_args, rec, cb*)

**expands\_field** (*field*)

Checks if this expander should expand a Rule field type

**Parameters** **field** – the field to check

**Returns** True if *field* should be expanded.

**format** (*item, \*args, \*\*kwargs*)

Format *item* using *\*args* and *\*\*kwargs*

**format\_annotated** (*item, expand\_args*)

Wrapper for `:meth:format` preserving `AnnotatedString` flags

Calls `:meth:format` to format *item* into a new string and copies flags from original item.

This is used by `:meth:expand`

**link\_workflow** (*workflow*)

Called when the Expander is associated with a workflow

May be called multiple times if a new workflow object is created.

**exception** `ymp.snakemake.CircularReferenceException` (*deps*, *rule*)

Bases: `ymp.exceptions.YmpRuleError`

Exception raised if parameters in rule contain a circular reference

**class** `ymp.snakemake.ColonExpander`

Bases: `ymp.snakemake.FormatExpander`

Expander using `{:xyz:}` formatted variables.

**regex** = `re.compile('\n \\{:\n (?=(\n \\s*\n (?P<name>(?:. (?!\s*\\:)))*.)\n \\s*\n )`

**spec** = `'{:{}:}'`

**class** `ymp.snakemake.DefaultExpander` (*\*\*kwargs*)

Bases: `ymp.snakemake.InheritanceExpander`

Adds default values to rules

The implementation simply makes all rules inherit from a defaults rule.

Creates DefaultExpander

Each parameter passed is considered a RuleInfo default value. Where applicable, Snakemake's argtuples (`[]`, `{}`) must be passed.

**get\_super** (*rule*, *ruleinfo*)

Find rule parent

#### Parameters

- **rule** (`Rule`) – Rule object being built
- **ruleinfo** (`RuleInfo`) – RuleInfo object describing rule being built

**Returns** name of parent rule and RuleInfo describing parent rule or (None, None).

**Return type** 2-Tuple

**exception** `ymp.snakemake.ExpandLateException`

Bases: `Exception`

**class** `ymp.snakemake.ExpandableWorkflow` (*\*args*, *\*\*kwargs*)

Bases: `snakemake.workflow.Workflow`

Adds hook for additional rule expansion methods to Snakemake

Constructor for ExpandableWorkflow overlay attributes

This may be called on an already initialized Workflow object.

**classmethod** **activate** ()

Installs the ExpandableWorkflow

Replaces the Workflow object in the `snakemake.workflow` module with an instance of this class and initializes default expanders (the `snakemake` syntax).

**add\_rule** (*name=None*, *lineno=None*, *snakefile=None*, *checkpoint=False*)

Add a rule.

#### Parameters

- **name** – name of the rule

- **lineno** – line number within the snakefile where the rule was defined
- **snakefile** – name of file in which rule was defined

**classmethod** `clear()`

**classmethod** `ensure_global_workflow()`

**get\_rule** (*name=None*)

Get rule by name. If name is none, the last created rule is returned.

**Parameters** **name** – the name of the rule

**global\_workflow** = `<ymp.snakemake.ExpandableWorkflow object>`

**classmethod** `load_workflow (snakefile='/home/docs/checkouts/readthedocs.org/user_builds/ymp/checkouts/stable/src/ymp')`

**classmethod** `register_expanders (*expanders)`

Register an object the `expand()` function of which will be called on each `RuleInfo` object before it is passed on to `snakemake`.

**rule** (*name=None, lineno=None, snakefile=None, checkpoint=None*)

Intercepts “rule:” Here we have the entire `ruleinfo` object

**class** `ymp.snakemake.FormatExpander`

Bases: `ymp.snakemake.BaseExpander`

Expander using a custom formatter object.

**class** `Formatter (expander)`

Bases: `ymp.string.ProductFormatter`

**parse** (*format\_string*)

**format** (*\*args, \*\*kwargs*)

Format *item* using *\*args* and *\*\*kwargs*

**get\_names** (*pattern*)

**regex** = `re.compile('\n \\{\n (?=(\n (?P<name>[^\{\}]++)\n ))\\1\n \\}\n ', re.VERBOSE)`

**spec** = `'{{{}}}'`

**exception** `ymp.snakemake.InheritanceException (msg, rule, parent, include=None, lineno=None, snakefile=None)`

Bases: `snakemake.exceptions.RuleException`

Exception raised for errors during rule inheritance

Creates a new instance of `RuleException`.

Arguments *message* – the exception message *include* – iterable of other exceptions to be included *lineno* – the line the exception originates *snakefile* – the file the exception originates

**class** `ymp.snakemake.InheritanceExpander`

Bases: `ymp.snakemake.BaseExpander`

Adds class-like inheritance to `Snakemake` rules

To avoid redundancy between closely related rules, e.g. rules for single ended and paired end data, YMP allows `Snakemake` rules to inherit from another rule.

## Example

Derived rules are always created with an implicit `ruleorder` statement, making Snakemake prefer the parent rule if either parent or child rule could be used to generate the requested output file(s).

Derived rules initially contain the same attributes as the parent rule. Each attribute assigned to the child rule overrides the matching attribute in the parent. Where attributes may contain named and unnamed values, specifying a named value overrides only the value of that name while specifying an unnamed value overrides all unnamed values in the parent attribute.

**KEYWORD** = `'ymp: extends'`

Comment keyword enabling inheritance

**expand** (*rule*, *ruleinfo*)

Expands RuleInfo object and children recursively.

Will call `:meth:format` (via `:meth:format_annotated`) on `str` items encountered in the tree and wrap encountered functions to be called once the wildcards object is available.

Set `ymp.print_rule = 1` before a `rule:` statement in snakefiles to enable debug logging of recursion.

### Parameters

- **rule** – The `:class:snakemake.rules.Rule` object to be populated with the data from the RuleInfo object passed from *item*
- **item** – The item to be expanded. Initially a `:class:snakemake.workflow.RuleInfo` object into which is recursively descended. May ultimately be `None`, `str`, `function`, `int`, `float`, `dict`, `list` or `tuple`.
- **expand\_args** – Parameters passed on late expansion (when the dag tries to instantiate the *rule* into a job).
- **rec** – Recursion level

**get\_code\_line** (*rule*)

Returns the source line defining *rule*

**Return type** `str`

**get\_super** (*rule*, *ruleinfo*)

Find rule parent

### Parameters

- **rule** (`Rule`) – Rule object being built
- **ruleinfo** (`RuleInfo`) – RuleInfo object describing rule being built

**Returns** name of parent rule and RuleInfo describing parent rule or (`None`, `None`).

**Return type** 2-Tuple

**class** `ymp.snakemake.NamedList` (*fromtuple=*`None`, *\*\*kwargs*)

Bases: `snakemake.io.Namedlist`

Extended version of Snakemake's `Namedlist`

- Fixes array assignment operator: Writing a field via `[]` operator updates the value accessed via `.` operator.
- Adds *fromtuple* to constructor: Builds from Snakemake's typical (*args*, *kwargs*) tuples as present in ruleinfo structures.



- Adds `update_tuple` method: Updates values in `(args, kwargs)` tuples as present in `ruleinfo` structures.

Create the object.

Arguments `toclone` – another `Namedlist` that shall be cloned fromdict – a dict that shall be converted to a

Unexpected indentation.

`Namedlist` (keys become names)

**get\_names** (*\*args, \*\*kwargs*)

Export `get_names` as public func

**update\_tuple** (*totuple*)

Update values in `(args, kwargs)` tuple. The tuple must be the same as used in the constructor and must not have been modified.

**class** `ymp.snakemake.RecursiveExpander`

Bases: `ymp.snakemake.BaseExpander`

Recursively expands `{xyz}` wildcards in Snakemake rules.

**expand** (*rule, ruleinfo*)

Recursively expand wildcards within `RuleInfo` object

**expands\_field** (*field*)

Returns true for all fields but `shell:`, `message:` and `wildcard_constraints`.

We don't want to mess with the regular expressions in the fields in `wildcard_constraints:`, and there is little use in expanding `message:` or `shell:` as these already have all wildcards applied just before job execution (by `format_wildcards()`).

**class** `ymp.snakemake.SnakemakeExpander`

Bases: `ymp.snakemake.BaseExpander`

Expand wildcards in strings returned from functions.

Snakemake does not do this by default, leaving wildcard expansion to the functions provided themselves. Since we never want `{input}` to be in a string returned as a file, we expand those always.

**expands\_field** (*field*)

Checks if this expander should expand a Rule field type

**Parameters** `field` – the field to check

**Returns** True if `field` should be expanded.

**format** (*item, \*args, \*\*kwargs*)

Format `item` using `*args` and `**kwargs`

**class** `ymp.snakemake.WorkflowObject` (*\*args, \*\*kwargs*)

Bases: `object`

Base for extension classes defined from snakefiles

This currently encompasses `ymp.env.Env` and `ymp.stage.Stage`.

This mixin sets the properties `filename` and `lineno` according to the definition source in the rules file. It also maintains a registry within the Snakemake workflow object and provides an accessor method to this registry.

**property** `defined_in`

**filename**

Name of file in which object was defined

Type `str`

**classmethod** `get_registry` (*clean=False*)

Return all objects of this class registered with current workflow

**lineno**

Line number of object definition

Type `int`

**classmethod** `new_registry` ()

**register** ()

Add self to registry

`ymp.snakemake.check_snakemake` ()

Return type `bool`

`ymp.snakemake.get_workflow` ()

Get active workflow, loading one if necessary

`ymp.snakemake.load_workflow` (*snakefile*)

Load new workflow

`ymp.snakemake.make_rule` (*name=None, lineno=None, snakefile=None, \*\*kwargs*)

`ymp.snakemake.networkx` ()

`ymp.snakemake.print_ruleinfo` (*rule, ruleinfo, func=<bound method Logger.debug of <Logger ymp.snakemake (WARNING)>>*)

Logs contents of Rule and RuleInfo objects.

**Parameters**

- **rule** (`Rule`) – Rule object to be printed
- **ruleinfo** (`RuleInfo`) – Matching RuleInfo object to be printed
- **func** – Function used for printing (default is `log.error`)

`ymp.snakemake.ruleinfo_fields` = {'benchmark': {'apply\_wildcards': True, 'format': 'string'}}  
describes attributes of `snakemake.workflow.RuleInfo`

## 6.1.17 ymp.snakemakelexer module

### `ymp.snakemakelexer`

**class** `ymp.snakemakelexer.SnakemakeLexer` (*\*args, \*\*kws*)

Bases: `pygments.lexers.python.PythonLexer`

**name** = 'Snakemake'

**tokens** = {'globalkeyword': [(<pygments.lexer.words object>, Token.Keyword)], 'root':

## 6.1.18 ymp.sphinxext module

This module contains a [Sphinx](#) extension for documenting YMP stages and [Snakemake](#) rules.

The *SnakemakeDomain* (name **sm**) provides the following directives:

```
.. sm:rule:: name
    Describes a Snakemake rule

.. sm:stage:: name
    Describes a YMP Stage
```

Both directives accept an optional `source` parameter. If given, a link to the source code of the stage or rule definition will be added. The format of the string passed is `filename:line`. Referenced Snakefiles will be highlighted with pygments and added to the documentation when building HTML.

The extension also provides an autodoc-like directive:

```
.. autosnake:: filename
    Generates documentation from Snakefile filename.
```

```
class ymp.sphinxext.AutoSnakefileDirective(name, arguments, options, content,
                                             lineno, content_offset, block_text, state,
                                             state_machine)
```

Bases: `docutils.parsers.rst.Directive`

Implements RST directive `.. autosnake:: filename`

The directive extracts docstrings from rules in snakefile and auto-generates documentation.

```
has_content = False
    This rule does not accept content
```

**Type** `bool`

```
load_workflow(file_path)
    Load the Snakefile
```

**Return type** `ExpandableWorkflow`

```
parse_doc(doc, source, idt=0)
    Convert doc string to StringList
```

**Parameters**

- **doc** (`str`) – Documentation text
- **source** (`str`) – Source filename
- **idt** (`int`) – Result indentation in characters (default 0)

**Return type** `StringList`

**Returns** `StringList` of re-indented documentation wrapped in newlines

```
parse_rule(rule, idt=0)
    Convert Rule to StringList
```

**Parameters**

- **rule** (`Rule`) – Rule object
- **idt** (`int`) – Result indentation in characters (default 0)

**Retuns:** `StringList` containing formatted Rule documentation

**Return type** `StringList`

**parse\_stage** (*stage*, *idt=0*)

**Return type** `StringList`

**required\_arguments** = 1

This rule needs one argument (the filename)

**Type** `int`

**run** ()

Entry point

**tpl\_rule** = '`.. sm:rule:: {name}`'

Template for generated Rule RSt

**Type** `str`

**tpl\_source** = '`:source: {filename}:{lineno}`'

Template option source

**Type** `str`

**tpl\_stage** = '`.. sm:stage:: {name}`'

Template for generated Stage RSt

**Type** `str`

**ymp.sphinxext.BASEPATH** = '`/home/docs/checkouts/readthedocs.org/user_builds/ymp/checkouts/st`'

Path in which YMP package is located

**Type** `str`

**class** `ymp.sphinxext.DomainTocTreeCollector`

Bases: `sphinx.environment.collectors.EnvironmentCollector`

Add Sphinx Domain entries to the TOC

**clear\_doc** (*app*, *env*, *docname*)

Clear data from environment

If we have cached data in environment for document *docname*, we should clear it here.

**Return type** `None`

**get\_ref** (*node*)

**Return type** `Optional[Node]`

**locate\_in\_toc** (*app*, *node*)

**Return type** `Optional[Node]`

**make\_heading** (*node*)

**Return type** `List[Node]`

**merge\_other** (*app*, *env*, *docnames*, *other*)

Merge with results from parallel processes

Called if Sphinx is processing documents in parallel. We should merge this from *other* into *env* for all *docnames*.

**Return type** `None`

**process\_doc** (*app, doctree*)

Process doctree

This is called by `read-doctree`, so after the doctree has been loaded. The signal is processed in registered first order, so we are called after built-in extensions, such as the `sphinx.environment.collectors.toc` extension building the TOC.

**Return type** `None`

**select\_doc\_nodes** (*doctree*)

Select the nodes for which entries in the TOC are desired

This is a separate method so that it might be overridden by subclasses wanting to add other types of nodes to the TOC.

**Return type** `List[Node]`

**select\_toc\_location** (*app, node*)

Select location in TOC where *node* should be referenced

**Return type** `Node`

**toc\_insert** (*docname, tocnode, node, heading*)

**Return type** `None`

**class** `ymp.sphinxext.SnakemakeDomain` (*env*)

Bases: `sphinx.domains.Domain`

Snakemake language domain

**clear\_doc** (*docname*)

Delete objects derived from file *docname*

**data\_version** = 0

**directives** = {'rule': <class 'ymp.sphinxext.SnakemakeRule'>, 'stage': <class 'ymp.sp

**get\_objects** ()

Return an iterable of “object descriptions”.

Object descriptions are tuples with six items:

**name** Fully qualified name.

**dispname** Name to display when searching/linking.

**type** Object type, a key in `self.object_types`.

**docname** The document where it is to be found.

**anchor** The anchor name for the object.

**priority** How “important” the object is (determines placement in search results). One of:

1 Default priority (placed before full-text matches).

0 Object is important (placed before default-priority objects).

2 Object is unimportant (placed after full-text matches).

-1 Object should not show up in search at all.

**initial\_data** = {'objects': {}}

**label** = 'Snakemake'

**name** = 'sm'

```
object_types = {'rule': <sphinx.domains.ObjType object>, 'stage': <sphinx.domains.Obj
```

```
resolve_xref (env, fromdocname, builder, typ, target, node, contnode)
```

Resolve the pending\_xref *node* with the given *typ* and *target*.

This method should return a new node, to replace the xref node, containing the *contnode* which is the markup content of the cross-reference.

If no resolution can be found, None can be returned; the xref node will then given to the **:event:`missing-reference`** event, and if that yields no resolution, replaced by *contnode*.

Unknown interpreted text role “event”.

The method can also raise `sphinx.environment.NoUri` to suppress the **:event:`missing-reference`** event being emitted.

Unknown interpreted text role “event”.

```
roles = {'rule': <sphinx.roles.XRefRole object>, 'stage': <sphinx.roles.XRefRole obj
```

```
class ymp.sphinxext.SnakemakeRule (name, arguments, options, content, lineno, content_offset,  
                                   block_text, state, state_machine)
```

Bases: `ymp.sphinxext.YmpObjectDescription`

Directive `sm:rule::` describing a Snakemake rule

```
typename = 'rule'
```

```
class ymp.sphinxext.YmpObjectDescription (name, arguments, options, content, lineno, con-  
                                           tent_offset, block_text, state, state_machine)
```

Bases: `sphinx.directives.ObjectDescription`

Base class for RSt directives in SnakemakeDomain

Since this inherits from Sphinx’ `ObjectDescription`, content generated by the directive will always be inside an `addnodes.desc`.

**Parameters** `source` – Specify source position as `file:line` to create link

```
add_source_link (signode)
```

Add link to source code to *signode*

**Return type** `None`

```
add_target_and_index (name, sig, signode)
```

Add cross-reference IDs and entries to `self.indexnode`

**Return type** `None`

```
get_index_text (typename, name)
```

Formats object for entry into index

**Return type** `str`

```
handle_signature (sig, signode)
```

Parse rule signature *sig* into RST nodes and append them to *signode*.

The return value identifies the object and is passed to `add_target_and_index()` unchanged

**Parameters**

- **sig** (`str`) – Signature string (i.e. string passed after directive)
- **signode** (`desc`) – Node created for object signature

**Return type** `str`

**Returns** Normalized signature (white space removed)

```

    option_spec = {'source': <function unchanged>}
    typename = '[object name]'
class ymp.sphinxext.YmpStage(name, arguments, options, content, lineno, content_offset,
                             block_text, state, state_machine)
    Bases: ymp.sphinxext.YmpObjectDescription
    Directive sm:stage:: describing an YMP stage
    typename = 'stage'
ymp.sphinxext.collect_pages(app)
    Add Snakefiles to documentation (in HTML mode)
ymp.sphinxext.relpath(path)
    Make absolute path relative to BASEPATH
    Parameters path (str) – absolute path
    Return type str
    Returns path relative to BASEPATH
ymp.sphinxext.setup(app)
    Register the extension with Sphinx

```

### 6.1.19 ymp.string module

```

exception ymp.string.FormattingError(message, fieldname)
    Bases: AttributeError
class ymp.string.GetNameFormatter
    Bases: string.Formatter
    get_names(pattern)
class ymp.string.OverrideJoinFormatter
    Bases: string.Formatter
    Formatter with overridable join method
    The default formatter joins all arguments with "".join(args). This class overrides _vformat() with
    identical code, changing only that line to one that can be overridden by a derived class.
    join(args)
        Joins the expanded pieces of the template string to form the output.
        This function is equivalent to ''.join(args). By overriding it, alternative methods can be imple-
        mented, e.g. to create a list of strings, each corresponding to a the cross product of the expanded variables.
        Return type Union[List[str], str]
class ymp.string.PartialFormatter
    Bases: string.Formatter
    Formats what it can and leaves the remainder untouched
    get_field(field_name, args, kwargs)
class ymp.string.ProductFormatter
    Bases: ymp.string.OverrideJoinFormatter
    String Formatter that creates a list of strings each expanded using one point in the cartesian product of all
    replacement values.

```

If none of the arguments evaluate to lists, the result is a string, otherwise it is a list.

```
>>> ProductFormatter().format("{A} and {B}", A=[1,2], B=[3,4])
"1 and 3"
"1 and 4"
"2 and 3"
"2 and 4"
```

**format\_field** (*value*, *format\_spec*)

**join** (*args*)

Joins the expanded pieces of the template string to form the output.

This function is equivalent to `' '.join(args)`. By overriding it, alternative methods can be implemented, e.g. to create a list of strings, each corresponding to a the cross product of the expanded variables.

**Return type** `Union[List[str], str]`

**class** `ymp.string.QuotedElementFormatter` (*\*args*, *\*\*kwargs*)

Bases: `snakemake.utils.SequenceFormatter`

**class** `ymp.string.RegexFormatter` (*regex*)

Bases: `string.Formatter`

String Formatter accepting a regular expression defining the format of the expanded tags.

**get\_names** (*format\_string*)

Get set of field names in *format\_string*

**Return type** `Set[str]`

**parse** (*format\_string*)

Parse *format\_string* into tuples. Tuples contain *literal\_text*: text to copy *field\_name*: follwed by field name  
*format\_spec*: conversion:

`ymp.string.make_formatter` (*product=None*, *regex=None*, *partial=None*, *quoted=None*)

## 6.1.20 ymp.util module

`ymp.util.R` (*code=""*, *\*\*kwargs*)

Execute R code

This function executes the R code given as a string. Additional arguments are injected into the R environment. The value of the last R statement is returned.

The function requires rpy2 to be installed.

**Parameters**

- **code** (*str*) – R code to be executed
- **\*\*kwargs** (*dict*) – variables to inject into R globalenv

**Yields** value of last R statement

```
>>> R("1*1", input=input)
```

`ymp.util.Rmd` (*rmd*, *out*, *\*\*kwargs*)

`ymp.util.activate_R` ()

`ymp.util.fasta_names` (*fasta\_file*)



`ymp.util.file_not_empty` (*fn*)  
Checks if a file is not empty, accounting for gz minimum size 20

`ymp.util.filter_out_empty` (*\*args*)  
Removes empty sets of files from input file lists.

Takes a variable number of file lists of equal length and removes indices where any of the files is empty. Strings are converted to lists of length 1.

Returns a generator tuple.

Example: `r1, r2 = filter_out_empty(input.r1, input.r2)`

`ymp.util.glob_wildcards` (*pattern, files=None*)  
Glob the values of the wildcards by matching the given pattern to the filesystem. Returns a named tuple with a list of values for each wildcard.

`ymp.util.is_fq` (*path*)

`ymp.util.make_local_path` (*icfg, url*)

`ymp.util.read_propfiles` (*files*)

## 6.1.21 ymp.yaml module

**class** `ymp.yaml.AttrItemAccessMixin`  
Bases: `object`

Mixin class mapping dot to bracket access

Added to classes implementing `__getitem__`, `__setitem__` and `__delitem__`, this mixin will allow accessing items using dot notation. I.e. “object.xyz” is translated to “object[xyz]”.

**exception** `ymp.yaml.LayeredConfAccessError`  
Bases: `ymp.yaml.LayeredConfError`, `KeyError`, `IndexError`

Can’t access

**exception** `ymp.yaml.LayeredConfError`  
Bases: `Exception`

Error in LayeredConf

**class** `ymp.yaml.LayeredConfProxy` (*maps, parent=None, key=None*)  
Bases: `ymp.yaml.MultiMapProxy`

Layered configuration

**save** (*outstream=None, layer=0*)

**exception** `ymp.yaml.LayeredConfWriteError`  
Bases: `ymp.yaml.LayeredConfError`

Can’t write

**exception** `ymp.yaml.MixedTypeError`  
Bases: `Exception`

Mixed types in proxy collection

**class** `ymp.yaml.MultiMapProxy` (*maps, parent=None, key=None*)  
Bases: `collections.abc.Mapping`, `ymp.yaml.MultiProxy`, `ymp.yaml.AttrItemAccessMixin`

Mapping Proxy for layered containers

**get** (*k*, *d*) → *D*[*k*] if *k* in *D*, else *d*. *d* defaults to *None*.

**items** () → a set-like object providing a view on *D*'s items

**keys** () → a set-like object providing a view on *D*'s keys

**values** () → an object providing a view on *D*'s values

**class** `ymp.yaml.MultiMapProxyItemsView` (*mapping*)

Bases: `ymp.yaml.MultiMapProxyMapView`, `collections.abc.ItemsView`

ItemsView for MultiMapProxy

**class** `ymp.yaml.MultiMapProxyKeysView` (*mapping*)

Bases: `ymp.yaml.MultiMapProxyMapView`, `collections.abc.KeysView`

KeysView for MultiMapProxy

**class** `ymp.yaml.MultiMapProxyMapView` (*mapping*)

Bases: `collections.abc.MappingView`

MapView for MultiMapProxy

**class** `ymp.yaml.MultiMapProxyValuesView` (*mapping*)

Bases: `ymp.yaml.MultiMapProxyMapView`, `collections.abc.ValuesView`

ValuesView for MultiMapProxy

**class** `ymp.yaml.MultiProxy` (*maps*, *parent=None*, *key=None*)

Bases: `object`

Base class for layered container structure

**add\_layer** (*name*, *container*)

**get\_files** ()

**get\_linenos** ()

**make\_map\_proxy** (*key*, *items*)

**make\_seq\_proxy** (*key*, *items*)

**remove\_layer** (*name*)

**to\_yaml** (*show\_source=False*)

**class** `ymp.yaml.MultiSeqProxy` (*maps*, *parent=None*, *key=None*)

Bases: `collections.abc.Sequence`, `ymp.yaml.MultiProxy`, `ymp.yaml.AttrItemAccessMixin`

Sequence Proxy for layered containers

**extend** (*item*)

`ymp.yaml.load` (*files*)

Load configuration files

Creates a `LayeredConfProxy` configuration object from a set of YAML files.

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### b

`ymp.blast`, 45  
`ymp.blast2gff`, 46

### c

`ymp.cli`, 31  
`ymp.cli.env`, 32  
`ymp.cli.init`, 32  
`ymp.cli.make`, 32  
`ymp.cli.shared_options`, 33  
`ymp.cli.show`, 34  
`ymp.cli.stage`, 35  
`ymp.cluster`, 46  
`ymp.common`, 47  
`ymp.config`, 48

### d

`ymp.dna`, 50  
`ymp.download`, 50

### e

`ymp.env`, 51  
`ymp.exceptions`, 53

### g

`ymp.gff`, 54

### h

`ymp.helpers`, 56

### m

`ymp.map2otu`, 56

### n

`ymp.nuc2aa`, 56

### s

`ymp.snakemake`, 57  
`ymp.snakemakelexer`, 62  
`ymp.sphinxext`, 63  
`ymp.stage`, 35  
`ymp.stage.base`, 35

`ymp.stage.expander`, 37  
`ymp.stage.groupby`, 37  
`ymp.stage.pipeline`, 37  
`ymp.stage.project`, 38  
`ymp.stage.reference`, 41  
`ymp.stage.stack`, 42  
`ymp.stage.stage`, 42  
`ymp.string`, 67

### u

`ymp.util`, 68

### y

`ymp`, 31  
`ymp.yaml`, 69



## Symbols

- F
  - ymp-env-prepare command line option, 14
  - ymp-make command line option, 19
  - ymp-submit command line option, 22
- J
  - ymp-submit command line option, 22
- N
  - ymp-env-prepare command line option, 14
  - ymp-make command line option, 19
  - ymp-submit command line option, 22
- P
  - ymp command line option, 9
  - ymp-env command line option, 10
  - ymp-env-activate command line option, 10
  - ymp-env-clean command line option, 11
  - ymp-env-export command line option, 11
  - ymp-env-install command line option, 12
  - ymp-env-list command line option, 13
  - ymp-env-prepare command line option, 14
  - ymp-env-remove command line option, 15
  - ymp-env-run command line option, 15
  - ymp-env-update command line option, 16
  - ymp-init command line option, 16
  - ymp-init-cluster command line option, 17
  - ymp-init-demo command line option, 17
  - ymp-init-project command line option, 18
  - ymp-make command line option, 18
  - ymp-show command line option, 20
  - ymp-stage command line option, 20
  - ymp-stage-list command line option, 21
  - ymp-submit command line option, 21, 22
- all
  - ymp-env-clean command line option, 11
  - ymp-env-list command line option, 13
- args <ARGS>
  - ymp-submit command line option, 23
- cluster-cores <N>
  - ymp-submit command line option, 22
- code
  - ymp-stage-list command line option, 21
- command <CMD>
  - ymp-submit command line option, 22
- conda-env-spec <conda\_env\_spec>
  - ymp-env-install command line option, 12
- conda-prefix <conda\_prefix>
  - ymp-env-install command line option, 12
- cores <CORES>
  - ymp-make command line option, 19
- cores <N>
  - ymp-submit command line option, 23
- create-missing
  - ymp-env-export command line option, 12
- dag
  - ymp-make command line option, 19
- debug
  - ymp-make command line option, 19
- debug-dag
  - ymp-make command line option, 19
- dest <FILE>
  - ymp-env-export command line option, 11
- drmaa
  - ymp-submit command line option, 22
- dry-run
  - ymp-env-install command line

option, 12

--dryrun

- ymp-env-prepare command line option, 14
- ymp-make command line option, 18
- ymp-submit command line option, 21

--dynamic

- ymp-env-list command line option, 13

--filetype <filetype>

- ymp-env-export command line option, 12

--force

- ymp-env-install command line option, 12
- ymp-env-prepare command line option, 14
- ymp-make command line option, 19
- ymp-submit command line option, 22

--forceall

- ymp-env-prepare command line option, 14
- ymp-make command line option, 19
- ymp-submit command line option, 22

--help

- ymp-show command line option, 20

--immediate

- ymp-submit command line option, 22

--install-completion

- ymp command line option, 9

--keepgoing

- ymp-env-prepare command line option, 14
- ymp-make command line option, 18
- ymp-submit command line option, 22

--latency-wait <T>

- ymp-submit command line option, 22

--lock

- ymp-env-prepare command line option, 14
- ymp-make command line option, 18
- ymp-submit command line option, 22

--log-file <log\_file>

- ymp command line option, 9
- ymp-env command line option, 10
- ymp-env-activate command line option, 10
- ymp-env-clean command line option, 11
- ymp-env-export command line option, 11
- ymp-env-install command line option, 12
- ymp-env-list command line option, 13
- ymp-env-prepare command line option, 14
- ymp-env-remove command line option, 15
- ymp-env-run command line option, 15
- ymp-env-update command line option, 16
- ymp-init command line option, 16
- ymp-init-cluster command line option, 17
- ymp-init-demo command line option, 17
- ymp-init-project command line option, 18
- ymp-make command line option, 18
- ymp-show command line option, 20
- ymp-stage command line option, 20
- ymp-stage-list command line option, 21
- ymp-submit command line option, 21

--long

- ymp-stage-list command line option, 21

--max-jobs-per-second <N>

- ymp-submit command line option, 22

--no-dynamic

- ymp-env-list command line option, 13

--no-lock

- ymp-env-prepare command line option, 14
- ymp-make command line option, 18
- ymp-submit command line option, 22

--no-static

- ymp-env-list command line option, 13

--nohup

- ymp-env-prepare command line option, 14
- ymp-make command line option, 19
- ymp-submit command line option, 22

--notemp

- ymp-env-prepare command line option, 14
- ymp-make command line option, 19
- ymp-submit command line option, 22

--overwrite

- ymp-env-export command line option, 12

--pdb

- ymp command line option, 9
- ymp-env command line option, 10
- ymp-env-activate command line option, 10
- ymp-env-clean command line option, 11



---

```

ymp-env-export command line option,
    11
ymp-env-install command line
    option, 12
ymp-env-list command line option, 13
ymp-env-prepare command line
    option, 14
ymp-env-remove command line option,
    15
ymp-env-run command line option, 15
ymp-env-update command line option,
    16
ymp-init command line option, 16
ymp-init-cluster command line
    option, 17
ymp-init-demo command line option,
    17
ymp-init-project command line
    option, 18
ymp-make command line option, 18
ymp-show command line option, 20
ymp-stage command line option, 20
ymp-stage-list command line option,
    21
ymp-submit command line option, 21
--printshellcmds
    ymp-env-prepare command line
        option, 14
    ymp-make command line option, 18
    ymp-submit command line option, 21
--profile <NAME>
    ymp-submit command line option, 22
--profile <profile>
    ymp command line option, 9
--quiet
    ymp command line option, 9
    ymp-env command line option, 10
    ymp-env-activate command line
        option, 10
    ymp-env-clean command line option,
        11
    ymp-env-export command line option,
        11
    ymp-env-install command line
        option, 12
    ymp-env-list command line option, 13
    ymp-env-prepare command line
        option, 14
    ymp-env-remove command line option,
        15
    ymp-env-run command line option, 15
    ymp-env-update command line option,
        16
    ymp-init command line option, 16
    ymp-init-cluster command line
        option, 17
    ymp-init-demo command line option,
        17
    ymp-init-project command line
        option, 18
    ymp-make command line option, 18
    ymp-show command line option, 20
    ymp-stage command line option, 20
    ymp-stage-list command line option,
        21
    ymp-submit command line option, 21
--reason
    ymp-env-prepare command line
        option, 14
    ymp-make command line option, 19
    ymp-submit command line option, 22
--reinstall <reinstall>
    ymp-env-update command line option,
        16
--rerun-incomplete
    ymp-env-prepare command line
        option, 14
    ymp-make command line option, 19
    ymp-submit command line option, 22
--reverse
    ymp-env-list command line option, 13
--ri
    ymp-env-prepare command line
        option, 14
    ymp-make command line option, 19
    ymp-submit command line option, 22
--rulegraph
    ymp-make command line option, 19
--scriptname <NAME>
    ymp-submit command line option, 23
--shadow-prefix <shadow_prefix>
    ymp-env-prepare command line
        option, 14
    ymp-make command line option, 19
    ymp-submit command line option, 22
--short
    ymp-stage-list command line option,
        21
--skip-missing
    ymp-env-export command line option,
        12
--snake-config <FILE>
    ymp-submit command line option, 22
--sort <sort_col>
    ymp-env-list command line option, 13
--source
    ymp-show command line option, 20
--static

```

```

    ymp-env-list command line option,13
--sync
    ymp-submit command line option,22
--touch
    ymp-env-prepare command line
        option,14
    ymp-make command line option,19
    ymp-submit command line option,22
--types
    ymp-stage-list command line option,
        21
--verbose
    ymp command line option,9
    ymp-env command line option,10
    ymp-env-activate command line
        option,10
    ymp-env-clean command line option,
        11
    ymp-env-export command line option,
        11
    ymp-env-install command line
        option,12
    ymp-env-list command line option,13
    ymp-env-prepare command line
        option,14
    ymp-env-remove command line option,
        15
    ymp-env-run command line option,15
    ymp-env-update command line option,
        16
    ymp-init command line option,16
    ymp-init-cluster command line
        option,17
    ymp-init-demo command line option,
        17
    ymp-init-project command line
        option,18
    ymp-make command line option,18
    ymp-show command line option,20
    ymp-stage command line option,20
    ymp-stage-list command line option,
        21
    ymp-submit command line option,21
--version
    ymp command line option,9
--wrapper <CMD>
    ymp-submit command line option,22
--yes
    ymp-init-cluster command line
        option,17
    ymp-init-project command line
        option,18
-a
    ymp-env-clean command line option,
        11
    ymp-env-list command line option,13
-c
    ymp-env-export command line option,
        12
    ymp-stage-list command line option,
        21
    ymp-submit command line option,22
-d
    ymp-env-export command line option,
        11
    ymp-submit command line option,22
-e
    ymp-env-install command line
        option,12
-f
    ymp-env-export command line option,
        12
    ymp-env-install command line
        option,12
    ymp-env-prepare command line
        option,14
    ymp-make command line option,19
    ymp-submit command line option,22
-h
    ymp-show command line option,20
-i
    ymp-submit command line option,22
-j
    ymp-make command line option,19
    ymp-submit command line option,23
-k
    ymp-env-prepare command line
        option,14
    ymp-make command line option,18
    ymp-submit command line option,22
-l
    ymp-stage-list command line option,
        21
    ymp-submit command line option,22
-n
    ymp-env-install command line
        option,12
    ymp-env-prepare command line
        option,14
    ymp-make command line option,18
    ymp-submit command line option,21
-p
    ymp-env-install command line
        option,12
    ymp-env-prepare command line
        option,14
    ymp-make command line option,18
    ymp-submit command line option,21

```

- q
  - ymp command line option, 9
  - ymp-env command line option, 10
  - ymp-env-activate command line option, 10
  - ymp-env-clean command line option, 11
  - ymp-env-export command line option, 11
  - ymp-env-install command line option, 12
  - ymp-env-list command line option, 13
  - ymp-env-prepare command line option, 14
  - ymp-env-remove command line option, 15
  - ymp-env-run command line option, 15
  - ymp-env-update command line option, 16
  - ymp-init command line option, 16
  - ymp-init-cluster command line option, 17
  - ymp-init-demo command line option, 17
  - ymp-init-project command line option, 18
  - ymp-make command line option, 18
  - ymp-show command line option, 20
  - ymp-stage command line option, 20
  - ymp-stage-list command line option, 21
  - ymp-submit command line option, 21
- r
  - ymp-env-list command line option, 13
  - ymp-env-prepare command line option, 14
  - ymp-make command line option, 19
  - ymp-submit command line option, 22
- s
  - ymp-env-export command line option, 12
  - ymp-env-list command line option, 13
  - ymp-show command line option, 20
  - ymp-stage-list command line option, 21
  - ymp-submit command line option, 22
- t
  - ymp-env-export command line option, 12
  - ymp-env-prepare command line option, 14
  - ymp-make command line option, 19
  - ymp-stage-list command line option, 21
- ymp-submit command line option, 22
- v
  - ymp command line option, 9
  - ymp-env command line option, 10
  - ymp-env-activate command line option, 10
  - ymp-env-clean command line option, 11
  - ymp-env-export command line option, 11
  - ymp-env-install command line option, 12
  - ymp-env-list command line option, 13
  - ymp-env-prepare command line option, 14
  - ymp-env-remove command line option, 15
  - ymp-env-run command line option, 15
  - ymp-env-update command line option, 16
  - ymp-init command line option, 16
  - ymp-init-cluster command line option, 17
  - ymp-init-demo command line option, 17
  - ymp-init-project command line option, 18
  - ymp-make command line option, 18
  - ymp-show command line option, 20
  - ymp-stage command line option, 20
  - ymp-stage-list command line option, 21
  - ymp-submit command line option, 21
- y
  - ymp-init-cluster command line option, 17
  - ymp-init-project command line option, 18
- 87 (*rule*), 29
- 88 (*rule*), 29
- 89 (*rule*), 29
- 90 (*rule*), 29
- 91 (*rule*), 29
- 92 (*rule*), 29
- A**
  - absdir() (*ymp.config.ConfigMgr* property), 48
  - activate() (*ymp.config.ConfigMgr* class method), 48
  - activate() (*ymp.snakemake.ExpandableWorkflow* class method), 58
  - activate\_R() (*in module ymp.util*), 68
  - active (*ymp.stage.stage.Stage* attribute), 43
  - add\_files() (*ymp.stage.reference.Reference* method), 41

[add\\_layer\(\)](#) (*ymp.yaml.MultiProxy method*), 70  
[add\\_param\(\)](#) (*ymp.stage.stage.Stage method*), 43  
[add\\_rule\(\)](#) (*ymp.snakemake.ExpandableWorkflow method*), 58  
[add\\_source\\_link\(\)](#) (*ymp.sphinxext.YmpObjectDescription method*), 66  
[add\\_target\\_and\\_index\(\)](#) (*ymp.sphinxext.YmpObjectDescription method*), 66  
[Alias\(\)](#) (*ymp.gff.Attributes property*), 54  
[align\\_mafft](#) (*rule*), 29  
[all\\_targets\(\)](#) (*ymp.stage.stack.StageStack method*), 42  
[annotate\\_blast](#) (*stage*), 25  
[annotate\\_diamond](#) (*stage*), 25  
[annotate\\_prodigal](#) (*stage*), 25  
[annotate\\_tblastn](#) (*stage*), 25  
[Archive](#) (*class in ymp.stage.reference*), 41  
[assemble\\_megahit](#) (*stage*), 25  
[assemble\\_metaspades](#) (*stage*), 25  
[assemble\\_trinity](#) (*stage*), 25  
[AttrDict](#) (*class in ymp.common*), 47  
[Attributes](#) (*class in ymp.gff*), 54  
[attributes\(\)](#) (*ymp.gff.Feature property*), 55  
[AttrItemAccessMixin](#) (*class in ymp.yaml*), 69  
[autosnake](#) (*directive*), 63  
[AutoSnakefileDirective](#) (*class in ymp.sphinxext*), 63

## B

[BaseExpander](#) (*class in ymp.snakemake*), 57  
[BASEPATH](#) (*in module ymp.sphinxext*), 64  
[BaseStage](#) (*class in ymp.stage.base*), 35  
[bin\\_metabat2](#) (*stage*), 25  
[blast7\\_all](#) (*rule*), 29  
[blast7\\_coverage\\_per\\_otu](#) (*rule*), 30  
[blast7\\_eval\\_hist](#) (*rule*), 29  
[blast7\\_eval\\_plot](#) (*rule*), 29  
[blast7\\_extract](#) (*rule*), 29  
[blast7\\_extract\\_merge](#) (*rule*), 29  
[blast7\\_merge](#) (*rule*), 29  
[blast7\\_reports](#) (*rule*), 29  
[BlastParser](#) (*class in ymp.blast*), 45

## C

[Cache](#) (*class in ymp.common*), 47  
[CacheDict](#) (*class in ymp.common*), 47  
[can\\_provide\(\)](#) (*ymp.stage.base.BaseStage method*), 35  
[can\\_provide\(\)](#) (*ymp.stage.pipeline.Pipeline method*), 37  
[categorize\\_by\\_function](#) (*rule*), 30  
[cdhit\\_fna\\_single](#) (*rule*), 29

[cfg](#) (*ymp.stage.base.ConfigStage attribute*), 36  
[check](#) (*stage*), 26  
[check\\_snakemake\(\)](#) (*in module ymp.snakemake*), 62  
[choose\\_fq\\_columns\(\)](#) (*ymp.stage.project.Project method*), 39  
[choose\\_id\\_column\(\)](#) (*ymp.stage.project.Project method*), 39  
[CircularReferenceException](#), 58  
[clear\(\)](#) (*ymp.snakemake.ExpandableWorkflow class method*), 59  
[clear\\_doc\(\)](#) (*ymp.sphinxext.DomainTocTreeCollector method*), 64  
[clear\\_doc\(\)](#) (*ymp.sphinxext.SnakemakeDomain method*), 65  
[close\(\)](#) (*ymp.common.Cache method*), 47  
[cluster\(\)](#) (*ymp.config.ConfigMgr property*), 48  
[cluster\\_cdhit](#) (*stage*), 26  
[ClusterMS](#) (*class in ymp.cluster*), 46  
[collect\\_pages\(\)](#) (*in module ymp.sphinxext*), 67  
[ColonExpander](#) (*class in ymp.snakemake*), 58  
[column\(\)](#) (*ymp.stage.project.PandasProjectData method*), 38  
[column\(\)](#) (*ymp.stage.project.SQLiteProjectData method*), 40  
[columns\(\)](#) (*ymp.stage.project.PandasProjectData method*), 38  
[columns\(\)](#) (*ymp.stage.project.SQLiteProjectData method*), 40  
[combine\\_with\\_ref](#) (*rule*), 29  
[COMMAND](#)  
     [ymp-env-run](#) command line option, 16  
[command\(\)](#) (*in module ymp.cli.shared\_options*), 34  
[command\(\)](#) (*ymp.cli.shared\_options.Group method*), 33  
[commit\(\)](#) (*ymp.common.Cache method*), 47  
[complete\(\)](#) (*ymp.cli.make.TargetParam class method*), 32  
[complete\(\)](#) (*ymp.cli.show.ConfigPropertyParam method*), 34  
[complete\(\)](#) (*ymp.stage.stack.StageStack method*), 42  
[conda\(\)](#) (*ymp.config.ConfigMgr property*), 48  
[CondaPathExpander](#) (*class in ymp.env*), 51  
[CONF\\_DEFAULT\\_FNAME](#) (*ymp.config.ConfigMgr attribute*), 48  
[CONF\\_FNAME](#) (*ymp.config.ConfigMgr attribute*), 48  
[CONF\\_USER\\_FNAME](#) (*ymp.config.ConfigMgr attribute*), 48  
[ConfigExpander](#) (*class in ymp.config*), 48  
[ConfigExpander.Formatter](#) (*class in ymp.config*), 48  
[ConfigMgr](#) (*class in ymp.config*), 48  
[ConfigPropertyParam](#) (*class in ymp.cli.show*), 34  
[ConfigStage](#) (*class in ymp.stage.base*), 36  
[constraint\(\)](#) (*ymp.stage.stage.Param property*), 42

`convert()` (*ymp.cli.show.ConfigPropertyParam method*), 34  
`convert_to_closed_ref(rule)`, 30  
`correct_bbmap(stage)`, 26  
`count_diamond(stage)`, 26  
`count_stringtie(stage)`, 26  
`coverage_samtools(stage)`, 26  
`create()` (*ymp.env.Env method*), 52

## D

`data()` (*ymp.stage.project.Project property*), 39  
`data_version` (*ymp.sphinxext.SnakemakeDomain attribute*), 65  
`DATABASE` (*ymp.blast.Fmt7Parser attribute*), 45  
`db_url()` (*ymp.stage.project.SQLiteProjectData property*), 40  
`Dbxref()` (*ymp.gff.Attributes property*), 54  
`debug()` (*in module ymp.cli.make*), 32  
`dedup_bbmap(stage)`, 26  
`DefaultExpander` (*class in ymp.snakemake*), 58  
`defined_in()` (*ymp.snakemake.WorkflowObject property*), 61  
`defined_in()` (*ymp.stage.base.ConfigStage property*), 36  
`defined_in()` (*ymp.stage.stack.StageStack property*), 42  
`Derives_From()` (*ymp.gff.Attributes property*), 54  
`dir()` (*ymp.config.ConfigMgr property*), 48  
`directives` (*ymp.sphinxext.SnakemakeDomain attribute*), 65  
`dirname` (*ymp.stage.reference.Archive attribute*), 41  
`doc()` (*ymp.stage.base.BaseStage method*), 35  
`docstring` (*ymp.stage.base.BaseStage attribute*), 35  
`DomainToCtreeCollector` (*class in ymp.sphinxext*), 64  
`download_file_ftp(rule)`, 29  
`download_file_http(rule)`, 29  
`DownloadThread` (*class in ymp.download*), 50  
`dump()` (*ymp.stage.project.PandasProjectData method*), 38  
`dump()` (*ymp.stage.project.SQLiteProjectData method*), 40  
`duplicate_rows()` (*ymp.stage.project.PandasProjectData method*), 38  
`duplicate_rows()` (*ymp.stage.project.SQLiteProjectData method*), 40  
`dust_bbmap(stage)`, 26

## E

`emirge_info` (*class in ymp.map2otu*), 56  
`emit()` (*ymp.cli.shared\_options.TqdmHandler method*), 34  
`enable_debug()` (*in module ymp.cli.shared\_options*), 34  
`encode_barcode_path()` (*ymp.stage.project.Project method*), 39  
`end()` (*ymp.gff.Feature property*), 55  
`ensure_global_workflow()` (*ymp.snakemake.ExpandableWorkflow class method*), 59  
`ensure_list()` (*in module ymp.common*), 47  
`ensuredir()` (*ymp.config.ConfigMgr property*), 48  
`Env` (*class in ymp.env*), 52  
`env()` (*ymp.stage.stage.Stage method*), 44  
`env_wait(rule)`, 30  
`ENVNAME`  
   `ymp-env-activate` command line option, 10  
   `ymp-env-run` command line option, 16  
`ENVNAMES`  
   `ymp-env-clean` command line option, 11  
   `ymp-env-export` command line option, 12  
   `ymp-env-install` command line option, 13  
   `ymp-env-list` command line option, 13  
   `ymp-env-remove` command line option, 15  
   `ymp-env-update` command line option, 16  
`error()` (*in module ymp.cluster*), 46  
`error()` (*ymp.download.FileDownloader method*), 50  
`expand()` (*ymp.config.ConfigMgr method*), 48  
`expand()` (*ymp.config.OverrideExpander method*), 49  
`expand()` (*ymp.snakemake.BaseExpander method*), 57  
`expand()` (*ymp.snakemake.InheritanceExpander method*), 60  
`expand()` (*ymp.snakemake.RecursiveExpander method*), 61  
`expand_dict()` (*ymp.snakemake.BaseExpander method*), 57  
`expand_func()` (*ymp.snakemake.BaseExpander method*), 57  
`expand_list()` (*ymp.snakemake.BaseExpander method*), 57  
`expand_ruleinfo()` (*ymp.snakemake.BaseExpander method*), 57  
`expand_ruleinfo()` (*ymp.stage.expander.StageExpander method*), 37  
`expand_str()` (*ymp.snakemake.BaseExpander method*), 57  
`expand_str()` (*ymp.stage.expander.StageExpander method*), 37  
`expand_tuple()` (*ymp.snakemake.BaseExpander method*), 57

ExpandableWorkflow (class in *ymp.snakemake*), 58  
 ExpandLateException, 58  
 expands\_field() (*ymp.config.ConfigExpander* method), 48  
 expands\_field() (*ymp.env.CondaPathExpander* method), 51  
 expands\_field() (*ymp.snakemake.BaseExpander* method), 57  
 expands\_field() (*ymp.snakemake.RecursiveExpander* method), 61  
 expands\_field() (*ymp.snakemake.SnakemakeExpander* method), 61  
 expands\_field() (*ymp.stage.expander.StageExpander* method), 37  
 export() (*ymp.env.Env* method), 53  
 extend() (*ymp.yaml.MultiSeqProxy* method), 70  
 extract\_reads(stage), 26  
 extract\_seqs(stage), 26

## F

faa\_fastp(rule), 29  
 fasta\_dna2aa() (in module *ymp.nuc2aa*), 56  
 fasta\_names() (in module *ymp.util*), 68  
 fasta\_to\_fastp\_gz(rule), 29  
 fastq\_dump(rule), 29  
 Feature (class in *ymp.gff*), 55  
 FIELD\_MAP (*ymp.blast.BlastParser* attribute), 45  
 FIELD\_TYPE (*ymp.blast.BlastParser* attribute), 45  
 field\_types (*ymp.blast.Fmt6Parser* attribute), 45  
 fields (*ymp.blast.Fmt6Parser* attribute), 45  
 FIELDS (*ymp.blast.Fmt7Parser* attribute), 45  
 file\_not\_empty() (in module *ymp.util*), 68  
 FileDownloader (class in *ymp.download*), 50  
 filename (*ymp.snakemake.WorkflowObject* attribute), 61  
 filename (*ymp.stage.base.ConfigStage* attribute), 36  
 files (*ymp.stage.reference.Archive* attribute), 41  
 filter\_bmtagger(stage), 27  
 filter\_out\_empty() (in module *ymp.util*), 69  
 find\_config() (*ymp.config.ConfigMgr* class method), 48  
 find\_stage() (in module *ymp.stage.stack*), 42  
 flatten() (in module *ymp.common*), 47  
 Fmt6Parser (class in *ymp.blast*), 45  
 Fmt7Parser (class in *ymp.blast*), 45  
 format() (*ymp.cli.shared\_options.LogFormatter* method), 33  
 format() (*ymp.env.CondaPathExpander* method), 52  
 format() (*ymp.snakemake.BaseExpander* method), 57  
 format() (*ymp.snakemake.FormatExpander* method), 59  
 format() (*ymp.snakemake.SnakemakeExpander* method), 61

format\_annotated() (*ymp.snakemake.BaseExpander* method), 57  
 format\_bbmap(stage), 27  
 format\_field() (*ymp.string.ProductFormatter* method), 68  
 FormatExpander (class in *ymp.snakemake*), 59  
 FormatExpander.Formatter (class in *ymp.snakemake*), 59  
 FormattingError, 67  
 fq2fa(rule), 30  
 fq\_names() (*ymp.stage.project.Project* property), 39  
 fwd\_fq\_names() (*ymp.stage.project.Project* property), 39  
 fwd\_pe\_fq\_names() (*ymp.stage.project.Project* property), 40

## G

Gap() (*ymp.gff.Attributes* property), 55  
 get() (*ymp.common.CacheDict* method), 47  
 get() (*ymp.download.DownloadThread* method), 50  
 get() (*ymp.download.FileDownloader* method), 51  
 get() (*ymp.stage.project.PandasProjectData* method), 38  
 get() (*ymp.stage.project.SQLiteProjectData* method), 40  
 get() (*ymp.stage.stack.StageStack* class method), 42  
 get() (*ymp.yaml.MultiMapProxy* method), 69  
 get\_all\_targets() (*ymp.stage.base.BaseStage* method), 35  
 get\_all\_targets() (*ymp.stage.pipeline.Pipeline* method), 38  
 get\_cache() (*ymp.common.Cache* method), 47  
 get\_code\_line() (*ymp.snakemake.InheritanceExpander* method), 60  
 get\_config() (in module *ymp*), 31  
 get\_env() (in module *ymp.cli.env*), 32  
 get\_envs() (in module *ymp.cli.env*), 32  
 get\_field() (*ymp.string.PartialFormatter* method), 67  
 get\_fields() (*ymp.blast.Fmt6Parser* method), 45  
 get\_fields() (*ymp.blast.Fmt7Parser* method), 45  
 get\_file() (*ymp.stage.reference.Reference* method), 41  
 get\_files() (*ymp.stage.reference.Archive* method), 41  
 get\_files() (*ymp.yaml.MultiProxy* method), 70  
 get\_fq\_names() (*ymp.stage.project.Project* method), 40  
 get\_ids() (*ymp.stage.project.Project* method), 40  
 get\_index\_text() (*ymp.sphinxext.YmpObjectDescription* method), 66  
 get\_inputs() (*ymp.stage.base.BaseStage* method), 36



- [get\\_inputs\(\)](#) (*ymp.stage.stage.Stage* method), 44  
[get\\_installed\\_env\\_hashes\(\)](#) (*ymp.env.Env* static method), 53  
[get\\_linenos\(\)](#) (*ymp.yaml.MultiProxy* method), 70  
[get\\_names\(\)](#) (*ymp.snakemake.FormatExpander* method), 59  
[get\\_names\(\)](#) (*ymp.snakemake.NamedList* method), 61  
[get\\_names\(\)](#) (*ymp.string.GetNameFormatter* method), 67  
[get\\_names\(\)](#) (*ymp.string.RegexFormatter* method), 68  
[get\\_objects\(\)](#) (*ymp.sphinxext.SnakemakeDomain* method), 65  
[get\\_path\(\)](#) (*ymp.stage.base.BaseStage* method), 36  
[get\\_path\(\)](#) (*ymp.stage.pipeline.Pipeline* method), 38  
[get\\_path\(\)](#) (*ymp.stage.reference.Reference* method), 41  
[get\\_ref\(\)](#) (*ymp.sphinxext.DomainTocTreeCollector* method), 64  
[get\\_registry\(\)](#) (*ymp.snakemake.WorkflowObject* class method), 62  
[get\\_rule\(\)](#) (*ymp.snakemake.ExpandableWorkflow* method), 59  
[get\\_super\(\)](#) (*ymp.snakemake.DefaultExpander* method), 58  
[get\\_super\(\)](#) (*ymp.snakemake.InheritanceExpander* method), 60  
[get\\_value\(\)](#) (*ymp.config.ConfigExpander.Formatter* method), 48  
[get\\_value\(\)](#) (*ymp.stage.expander.StageExpander.Formatter* method), 37  
[get\\_value\\_\(\)](#) (*ymp.stage.expander.StageExpander.Formatter* method), 37  
[get\\_workflow\(\)](#) (in module *ymp.snakemake*), 62  
[GetNameFormatter](#) (class in *ymp.string*), 67  
[glob\\_wildcards\(\)](#) (in module *ymp.util*), 69  
[global\\_workflow](#) (*ymp.snakemake.ExpandableWorkflow* attribute), 59  
[Group](#) (class in *ymp.cli.shared\_options*), 33  
[group\(\)](#) (in module *ymp.cli.shared\_options*), 34  
[GroupBy](#) (class in *ymp.stage.groupby*), 37  
[groupby\\_dedup\(\)](#) (*ymp.stage.project.PandasProjectData* method), 38  
[groupby\\_dedup\(\)](#) (*ymp.stage.project.SQLiteProjectData* method), 40  
[gunzip](#) (*rule*), 29
- ## H
- [handle\\_signature\(\)](#) (*ymp.sphinxext.YmpObjectDescription* method), 66  
[has\\_content](#) (*ymp.sphinxext.AutoSnakefileDirective* attribute), 63  
[hash](#) (*ymp.stage.reference.Archive* attribute), 41  
[have\\_command\(\)](#) (in module *ymp.cli.init*), 32  
[Hit](#) (*ymp.blast.Fmt6Parser* attribute), 45  
[HITSFOUND](#) (*ymp.blast.Fmt7Parser* attribute), 45  
[humann2](#) (*stage*), 27
- ## I
- [ID\(\)](#) (*ymp.gff.Attributes* property), 55  
[idcol\(\)](#) (*ymp.stage.project.Project* property), 40  
[identifying\\_columns\(\)](#) (*ymp.stage.project.PandasProjectData* method), 38  
[identifying\\_columns\(\)](#) (*ymp.stage.project.SQLiteProjectData* method), 40  
[index\\_bbmap](#) (*stage*), 27  
[index\\_blast](#) (*stage*), 27  
[index\\_bmtagger](#) (*stage*), 27  
[index\\_bowtie2](#) (*stage*), 27  
[index\\_diamond](#) (*stage*), 27  
[InheritanceException](#), 59  
[InheritanceExpander](#) (class in *ymp.snakemake*), 59  
[initial\\_data](#) (*ymp.sphinxext.SnakemakeDomain* attribute), 65  
[install\\_completion\(\)](#) (in module *ymp.cli*), 31  
[install\\_profiler\(\)](#) (in module *ymp.cli*), 31  
[installed\(\)](#) (*ymp.env.Env* property), 53  
[instance\(\)](#) (*ymp.config.ConfigMgr* class method), 49  
[Is\\_circular\(\)](#) (*ymp.gff.Attributes* property), 55  
[is\\_container\(\)](#) (in module *ymp.common*), 47  
[is\\_fq\(\)](#) (in module *ymp.util*), 69  
[iter\\_firsthit\(\)](#) (*ymp.blast.Fmt7Parser* method), 46  
[items\(\)](#) (*ymp.common.CacheDict* method), 47  
[items\(\)](#) (*ymp.yaml.MultiMapProxy* method), 70  
[iter\\_samples\(\)](#) (*ymp.stage.project.Project* method), 40
- ## J
- [join\(\)](#) (*ymp.string.OverrideJoinFormatter* method), 67  
[join\(\)](#) (*ymp.string.ProductFormatter* method), 68
- ## K
- [KEY\\_BCCOL](#) (*ymp.stage.project.Project* attribute), 39  
[KEY\\_DATA](#) (*ymp.stage.project.Project* attribute), 39  
[KEY\\_IDCOL](#) (*ymp.stage.project.Project* attribute), 39  
[KEY\\_PIPELINES](#) (*ymp.config.ConfigMgr* attribute), 48  
[KEY\\_PROJECTS](#) (*ymp.config.ConfigMgr* attribute), 48  
[KEY\\_READCOLS](#) (*ymp.stage.project.Project* attribute), 39  
[KEY\\_REFERENCES](#) (*ymp.config.ConfigMgr* attribute), 48  
[keys\(\)](#) (*ymp.common.CacheDict* method), 47  
[keys\(\)](#) (*ymp.yaml.MultiMapProxy* method), 70  
[KEYWORD](#) (*ymp.snakemake.InheritanceExpander* attribute), 60

## L

`label` (`ymp.sphinxext.SnakemakeDomain` attribute), 65  
`LayeredConfAccessError`, 69  
`LayeredConfError`, 69  
`LayeredConfProxy` (class in `ymp.yaml`), 69  
`LayeredConfWriteError`, 69  
`limits()` (`ymp.config.ConfigMgr` property), 49  
`lineno` (`ymp.snakemake.WorkflowObject` attribute), 62  
`lineno` (`ymp.stage.base.ConfigStage` attribute), 36  
`link_workflow()` (`ymp.snakemake.BaseExpander` method), 57  
`load()` (in module `ymp.yaml`), 70  
`load()` (`ymp.common.Cache` method), 47  
`load_all()` (`ymp.common.Cache` method), 47  
`load_data()` (`ymp.stage.project.PandasTableBuilder` method), 39  
`load_workflow()` (in module `ymp.snakemake`), 62  
`load_workflow()` (`ymp.snakemake.ExpandableWorkflow` class method), 59  
`load_workflow()` (`ymp.sphinxext.AutoSnakefileDirective` method), 63  
`locate_in_toc()` (`ymp.sphinxext.DomainTocTreeCollector` method), 64  
`Log` (class in `ymp.cli.shared_options`), 33  
`log()` (`ymp.download.FileDownloader` method), 51  
`log_options()` (in module `ymp.cli.shared_options`), 34  
`logfile_option()` (`ymp.cli.shared_options.Log` class method), 33  
`LogFormatter` (class in `ymp.cli.shared_options`), 33  
`Lsf` (class in `ymp.cluster`), 46

## M

`main()` (in module `ymp.map2otu`), 56  
`main()` (`ymp.download.DownloadThread` method), 50  
`make_bar_format()` (`ymp.download.FileDownloader` static method), 51  
`make_formatter()` (in module `ymp.string`), 68  
`make_heading()` (`ymp.sphinxext.DomainTocTreeCollector` method), 64  
`make_local_path()` (in module `ymp.util`), 69  
`make_map_proxy()` (`ymp.yaml.MultiProxy` method), 70  
`make_otu_table` (rule), 30  
`make_rule()` (in module `ymp.snakemake`), 62  
`make_seq_proxy()` (`ymp.yaml.MultiProxy` method), 70  
`make_unpack_rule()` (`ymp.stage.reference.Archive` method), 41  
`make_unpack_rules()` (`ymp.stage.reference.Reference` method), 41  
`map_bbmap` (stage), 27  
`map_bowtie2` (stage), 27  
`map_diamond` (stage), 27  
`map_hisat2` (stage), 27  
`map_star` (stage), 27  
`MapfileParser` (class in `ymp.map2otu`), 56  
`match()` (`ymp.stage.base.BaseStage` method), 36  
`match()` (`ymp.stage.stage.Stage` method), 44  
`mem()` (`ymp.config.ConfigMgr` method), 49  
`merge_other()` (`ymp.sphinxext.DomainTocTreeCollector` method), 64  
`metaphlan2` (stage), 27  
`minimize_variables()` (`ymp.stage.project.Project` method), 40  
`MixedTypeError`, 69  
`mkdir` (rule), 30  
`MkdirDict` (class in `ymp.common`), 47  
`mod_level()` (`ymp.cli.shared_options.Log` method), 33  
module  
    `ymp`, 31  
    `ymp.blast`, 45  
    `ymp.blast2gff`, 46  
    `ymp.cli`, 31  
    `ymp.cli.env`, 32  
    `ymp.cli.init`, 32  
    `ymp.cli.make`, 32  
    `ymp.cli.shared_options`, 33  
    `ymp.cli.show`, 34  
    `ymp.cli.stage`, 35  
    `ymp.cluster`, 46  
    `ymp.common`, 47  
    `ymp.config`, 48  
    `ymp.dna`, 50  
    `ymp.download`, 50  
    `ymp.env`, 51  
    `ymp.exceptions`, 53  
    `ymp.gff`, 54  
    `ymp.helpers`, 56  
    `ymp.map2otu`, 56  
    `ymp.nuc2aa`, 56  
    `ymp.snakemake`, 57  
    `ymp.snakemakelexer`, 62  
    `ymp.sphinxext`, 63  
    `ymp.stage`, 35  
    `ymp.stage.base`, 35  
    `ymp.stage.expander`, 37  
    `ymp.stage.groupby`, 37  
    `ymp.stage.pipeline`, 37  
    `ymp.stage.project`, 38  
    `ymp.stage.reference`, 41  
    `ymp.stage.stack`, 42  
    `ymp.stage.stage`, 42  
    `ymp.string`, 67  
    `ymp.util`, 68



`ymp.yaml`, 69  
`MultiMapProxy` (class in `ymp.yaml`), 69  
`MultiMapProxyItemsView` (class in `ymp.yaml`), 70  
`MultiMapProxyKeysView` (class in `ymp.yaml`), 70  
`MultiMapProxyMapView` (class in `ymp.yaml`), 70  
`MultiMapProxyValuesView` (class in `ymp.yaml`), 70  
`MultiProxy` (class in `ymp.yaml`), 70  
`MultiSeqProxy` (class in `ymp.yaml`), 70

## N

### NAME

`ymp-init-project` command line option, 18  
`name` (`ymp.snakemakelexer.SnakemakeLexer` attribute), 62  
`name` (`ymp.sphinxext.SnakemakeDomain` attribute), 65  
`name` (`ymp.stage.base.BaseStage` attribute), 36  
`name` (`ymp.stage.reference.Archive` attribute), 41  
`Name` () (`ymp.gff.Attributes` property), 55  
`NamedList` (class in `ymp.snakemake`), 60  
`networkx` () (in module `ymp.snakemake`), 62  
`new_registry` () (`ymp.snakemake.WorkflowObject` class method), 62  
`nohup` () (in module `ymp.cli.shared_options`), 34  
`noop` (rule), 30  
`norm_wildcards` () (in module `ymp.stage.stack`), 42  
`norm_wildcards` () (in module `ymp.stage.stage`), 45  
`normalize_16S` (rule), 30  
`Note` () (`ymp.gff.Attributes` property), 55  
`nrows` () (`ymp.stage.project.SQLiteProjectData` property), 40  
`nuc2aa` () (in module `ymp.dna`), 50  
`nuc2aa` () (in module `ymp.nuc2aa`), 56  
`nuc2num` () (in module `ymp.dna`), 50  
`nuc2num` () (in module `ymp.nuc2aa`), 56

## O

`object_types` (`ymp.sphinxext.SnakemakeDomain` attribute), 65  
`Ontology_term` () (`ymp.gff.Attributes` property), 55  
`option_spec` (`ymp.sphinxext.YmpObjectDescription` attribute), 66  
`OrderedDictMaker` (class in `ymp.helpers`), 56  
`otu_to_biom` (rule), 30  
`otu_to_qiime_txt` (rule), 30  
`outputs` () (`ymp.stage.base.BaseStage` property), 36  
`outputs` () (`ymp.stage.pipeline.Pipeline` property), 38  
`outputs` () (`ymp.stage.project.Project` property), 40  
`outputs` () (`ymp.stage.reference.Reference` property), 41  
`outputs` () (`ymp.stage.stage.Stage` property), 44  
`OverrideExpander` (class in `ymp.config`), 49

`OverrideJoinFormatter` (class in `ymp.string`), 67

## P

`pairnames` () (`ymp.config.ConfigMgr` property), 49  
`PandasProjectData` (class in `ymp.stage.project`), 38  
`PandasTableBuilder` (class in `ymp.stage.project`), 38  
`Param` (class in `ymp.stage.stage`), 42  
`param_func` () (`ymp.stage.stage.ParamChoice` method), 42  
`param_func` () (`ymp.stage.stage.ParamFlag` method), 43  
`param_func` () (`ymp.stage.stage.ParamInt` method), 43  
`ParamChoice` (class in `ymp.stage.stage`), 42  
`ParamFlag` (class in `ymp.stage.stage`), 42  
`ParamInt` (class in `ymp.stage.stage`), 43  
`Parent` () (`ymp.gff.Attributes` property), 55  
`parse` () (`ymp.snakemake.FormatExpander.Formatter` method), 59  
`parse` () (`ymp.string.RegexFormatter` method), 68  
`parse_doc` () (`ymp.sphinxext.AutoSnakefileDirective` method), 63  
`parse_number` () (in module `ymp.common`), 47  
`parse_rule` () (`ymp.sphinxext.AutoSnakefileDirective` method), 63  
`parse_stage` () (`ymp.sphinxext.AutoSnakefileDirective` method), 64  
`PartialFormatter` (class in `ymp.string`), 67  
`path` () (`ymp.stage.stack.StageStack` property), 42  
`pattern` () (`ymp.stage.stage.Param` method), 42  
`pe_fq_names` () (`ymp.stage.project.Project` property), 40  
`phase` () (`ymp.gff.Feature` property), 55  
`pick_closed_otus` (rule), 30  
`pick_open_otus` (rule), 30  
`Pipeline` (class in `ymp.stage.pipeline`), 37  
`pipeline` () (`ymp.config.ConfigMgr` property), 49  
`pipeline` () (`ymp.stage.pipeline.Pipeline` property), 38  
`platform` () (`ymp.config.ConfigMgr` property), 49  
`predict_metagenome` (rule), 30  
`prefetch` (rule), 29  
`prev` () (`ymp.stage.stack.StageStack` method), 42  
`prev` () (`ymp.stage.stage.Stage` method), 44  
`primermatch_bbmap` (stage), 27  
`print_rule` (in module `ymp`), 31  
`print_ruleinfo` () (in module `ymp.snakemake`), 62  
`process_doc` () (`ymp.sphinxext.DomainTocTreeCollector` method), 64  
`ProductFormatter` (class in `ymp.string`), 67  
`profile_centrifuge` (stage), 28  
`Project` (class in `ymp.stage.project`), 39  
`project_name` () (`ymp.stage.project.Project` property), 40

`properties()` (*ymp.cli.show.ConfigPropertyParams* property), 34

`PROPERTY`  
ymp-show command line option, 20

## Q

`qc_fastqc` (*stage*), 28

`qc_multiqc` (*stage*), 28

`qc_quast` (*stage*), 28

`quant_rsem` (*stage*), 28

`QUERY` (*ymp.blast.Fmt7Parser* attribute), 45

`query()` (*ymp.stage.project.SQLiteProjectData* method), 41

`quiet_option()` (*ymp.cli.shared\_options.Log* class method), 33

`QuotedElementFormatter` (class in *ymp.string*), 68

## R

`R()` (in module *ymp.util*), 68

`rarefy_table` (rule), 30

`raw_reads_source_path()`  
(*ymp.stage.project.Project* method), 40

`raxml_tree` (rule), 30

`RE_FILE` (*ymp.stage.project.Project* attribute), 39

`RE_REMOTE` (*ymp.stage.project.Project* attribute), 39

`RE_SRR` (*ymp.stage.project.Project* attribute), 39

`read()` (*ymp.map2otu.MapfileParser* method), 56

`read_propfiles()` (in module *ymp.util*), 69

`reader` (class in *ymp.gff*), 55

`reader()` (in module *ymp.blast*), 46

`RecursiveExpander` (class in *ymp.snakemake*), 61

`ref()` (*ymp.config.ConfigMgr* property), 49

`Reference` (class in *ymp.stage.reference*), 41

`references` (*stage*), 28

`regex` (*ymp.snakemake.ColonExpander* attribute), 58

`regex` (*ymp.snakemake.FormatExpander* attribute), 59

`RegexFormatter` (class in *ymp.string*), 68

`register()` (*ymp.snakemake.WorkflowObject* method), 62

`register_expanders()`  
(*ymp.snakemake.ExpandableWorkflow* class method), 59

`relpath()` (in module *ymp.sphinxext*), 67

`remove_bbmap` (*stage*), 28

`remove_layer()` (*ymp.yaml.MultiProxy* method), 70

`require()` (*ymp.stage.stage.Stage* method), 44

`required_arguments`  
(*ymp.sphinxext.AutoSnakefileDirective* attribute), 64

`resolve_prevs()` (*ymp.stage.stack.StageStack* method), 42

`resolve_xref()` (*ymp.sphinxext.SnakemakeDomain* method), 66

`rev_pe_fq_names()` (*ymp.stage.project.Project* property), 40

`Rmd()` (in module *ymp.util*), 68

`roles` (*ymp.sphinxext.SnakemakeDomain* attribute), 66

`rows()` (*ymp.stage.project.PandasProjectData* method), 38

`rows()` (*ymp.stage.project.SQLiteProjectData* method), 41

`rsem_index` (rule), 30

`rule()` (*ymp.snakemake.ExpandableWorkflow* method), 59

`RULE_MAIN_FNAME` (*ymp.config.ConfigMgr* attribute), 48

`ruleinfo_fields` (in module *ymp.snakemake*), 62

`run()` (*ymp.env.Env* method), 53

`run()` (*ymp.sphinxext.AutoSnakefileDirective* method), 64

`runs()` (*ymp.stage.project.Project* property), 40

## S

`satisfy_inputs()` (*ymp.stage.stage.Stage* method), 44

`save()` (*ymp.yaml.LayeredConfProxy* method), 69

`scnic_within_minsamp` (rule), 30

`scnic_within_sparcc_filter` (rule), 30

`score()` (*ymp.gff.Feature* property), 55

`se_fq_names()` (*ymp.stage.project.Project* property), 40

`select_doc_nodes()`  
(*ymp.sphinxext.DomainTocTreeCollector* method), 65

`select_toc_location()`  
(*ymp.sphinxext.DomainTocTreeCollector* method), 65

`seqid()` (*ymp.gff.Feature* property), 55

`set_logfile()` (*ymp.cli.shared\_options.Log* static method), 33

`set_prefix()` (*ymp.env.Env* method), 53

`setup()` (in module *ymp.sphinxext*), 67

`shell()` (*ymp.config.ConfigMgr* property), 49

`show()` (*ymp.exceptions.YmpRuleError* method), 54

`show()` (*ymp.exceptions.YmpStageError* method), 54

`show_help()` (in module *ymp.cli.show*), 35

`Slurm` (class in *ymp.cluster*), 46

`sm:rule` (directive), 63

`sm:stage` (directive), 63

`snake_params()` (in module *ymp.cli.make*), 32

`snakefiles()` (*ymp.config.ConfigMgr* property), 49

`snakemake_level_styles`  
(*ymp.cli.shared\_options.LogFormatter* attribute), 33

`snakemake_versions` (in module *ymp*), 31

`SnakemakeDomain` (class in *ymp.sphinxext*), 65

`SnakemakeExpander` (class in *ymp.snakemake*), 61

- SnakemakeLexer (class in *ymp.snakemakelexer*), 62  
 SnakemakeRule (class in *ymp.sphinxext*), 66  
 sort\_bam (stage), 28  
 source() (*ymp.gff.Feature* property), 55  
 source\_cfg() (*ymp.stage.project.Project* property), 40  
 source\_path() (*ymp.stage.project.Project* method), 40  
 spec (*ymp.snakemake.ColonExpander* attribute), 58  
 spec (*ymp.snakemake.FormatExpander* attribute), 59  
 split\_library (stage), 28  
 SQLiteProjectData (class in *ymp.stage.project*), 40  
 STAGE  
     ymp-stage-list command line option, 21  
 Stage (class in *ymp.stage.stage*), 43  
 StageExpander (class in *ymp.stage.expander*), 37  
 StageExpander.Formatter (class in *ymp.stage.expander*), 37  
 StageStack (class in *ymp.stage.stack*), 42  
 STAMP\_FILENAME (*ymp.stage.base.BaseStage* attribute), 35  
 star\_index (rule), 30  
 start() (*ymp.gff.Feature* property), 55  
 start\_snakemake() (in module *ymp.cli.make*), 32  
 states (*ymp.cluster.Lsf* attribute), 46  
 states (*ymp.cluster.Slurm* attribute), 46  
 status() (*ymp.cluster.Lsf* static method), 46  
 status() (*ymp.cluster.Slurm* static method), 46  
 store() (*ymp.common.Cache* method), 47  
 strand() (*ymp.gff.Feature* property), 55  
 string\_columns() (*ymp.stage.project.PandasProjectData* method), 38  
 string\_columns() (*ymp.stage.project.SQLiteProjectData* method), 41  
 strip\_components (*ymp.stage.reference.Archive* attribute), 41  
 submit() (*ymp.cluster.Lsf* static method), 46
- ## T
- tar (*ymp.stage.reference.Archive* attribute), 41  
 Target() (*ymp.gff.Attributes* property), 55  
 target() (*ymp.stage.stack.StageStack* method), 42  
 TARGET\_FILES  
     ymp-env-prepare command line option, 15  
     ymp-make command line option, 19  
     ymp-submit command line option, 23  
 TargetParam (class in *ymp.cli.make*), 32  
 targets() (*ymp.stage.stack.StageStack* property), 42  
 terminate() (*ymp.download.DownloadThread* method), 50  
 that() (*ymp.stage.stage.Stage* method), 44  
 this() (*ymp.stage.stage.Stage* method), 45
- ticktock (rule), 30  
 to\_yaml() (*ymp.yaml.MultiProxy* method), 70  
 toc\_insert() (*ymp.sphinxext.DomainTocTreeCollector* method), 65  
 tokens (*ymp.snakemakelexer.SnakemakeLexer* attribute), 62  
 tpl\_rule (*ymp.sphinxext.AutoSnakefileDirective* attribute), 64  
 tpl\_source (*ymp.sphinxext.AutoSnakefileDirective* attribute), 64  
 tpl\_stage (*ymp.sphinxext.AutoSnakefileDirective* attribute), 64  
 TqdmHandler (class in *ymp.cli.shared\_options*), 33  
 trim\_bbmap (stage), 28  
 trim\_sickle (stage), 28  
 trim\_trimmomatic (stage), 29  
 tupleofint() (*ymp.blast.BlastParser* method), 45  
 type() (*ymp.gff.Feature* property), 55  
 typename (*ymp.sphinxext.SnakemakeRule* attribute), 66  
 typename (*ymp.sphinxext.YmpObjectDescription* attribute), 67  
 typename (*ymp.sphinxext.YmpStage* attribute), 67
- ## U
- unload() (*ymp.config.ConfigMgr* class method), 49  
 unsplit\_path() (*ymp.stage.project.Project* method), 40  
 update() (*ymp.env.Env* method), 53  
 update\_dict() (in module *ymp.helpers*), 56  
 update\_tuple() (*ymp.snakemake.NamedList* method), 61  
 used\_stacks (*ymp.stage.stack.StageStack* attribute), 42
- ## V
- values() (*ymp.common.CacheDict* method), 47  
 values() (*ymp.yaml.MultiMapProxy* method), 70  
 variables() (*ymp.stage.project.Project* property), 40  
 verbose\_option() (*ymp.cli.shared\_options.Log* class method), 33
- ## W
- wc2path() (*ymp.stage.stage.Stage* method), 45  
 wildcards() (*ymp.stage.stage.Stage* method), 45  
 WorkflowObject (class in *ymp.snakemake*), 61  
 wrap() (in module *ymp.cli.stage*), 35  
 write() (*ymp.gff.writer* method), 55  
 write() (*ymp.map2otu.MapfileParser* method), 56  
 writer (class in *ymp.gff*), 55
- ## Y
- ymp  
     module, 31  
     ymp command line option

- P, 9
- install-completion, 9
- log-file <log\_file>, 9
- pdb, 9
- profile <profile>, 9
- quiet, 9
- verbose, 9
- version, 9
- q, 9
- v, 9
- ymp.blast
  - module, 45
- ymp.blast2gff
  - module, 46
- ymp.cli
  - module, 31
- ymp.cli.env
  - module, 32
- ymp.cli.init
  - module, 32
- ymp.cli.make
  - module, 32
- ymp.cli.shared\_options
  - module, 33
- ymp.cli.show
  - module, 34
- ymp.cli.stage
  - module, 35
- ymp.cluster
  - module, 46
- ymp.common
  - module, 47
- ymp.config
  - module, 48
- ymp.dna
  - module, 50
- ymp.download
  - module, 50
- ymp.env
  - module, 51
- ymp.exceptions
  - module, 53
- ymp.gff
  - module, 54
- ymp.helpers
  - module, 56
- ymp.map2otu
  - module, 56
- ymp.nuc2aa
  - module, 56
- ymp.snakemake
  - module, 57
- ymp.snakemakelexer
  - module, 62
- ymp.sphinxext
  - module, 63
- ymp.stage
  - module, 35
- ymp.stage.base
  - module, 35
- ymp.stage.expander
  - module, 37
- ymp.stage.groupby
  - module, 37
- ymp.stage.pipeline
  - module, 37
- ymp.stage.project
  - module, 38
- ymp.stage.reference
  - module, 41
- ymp.stage.stack
  - module, 42
- ymp.stage.stage
  - module, 42
- ymp.string
  - module, 67
- ymp.util
  - module, 68
- ymp.yaml
  - module, 69
- ymp-env command line option
  - P, 10
  - log-file <log\_file>, 10
  - pdb, 10
  - quiet, 10
  - verbose, 10
  - q, 10
  - v, 10
- ymp-env-activate command line option
  - P, 10
  - log-file <log\_file>, 10
  - pdb, 10
  - quiet, 10
  - verbose, 10
  - q, 10
  - v, 10
  - ENVNAME, 10
- ymp-env-clean command line option
  - P, 11
  - all, 11
  - log-file <log\_file>, 11
  - pdb, 11
  - quiet, 11
  - verbose, 11
  - a, 11
  - q, 11
  - v, 11
  - ENVNAMES, 11

## ymp-env-export command line option

-P, 11  
 --create-missing, 12  
 --dest <FILE>, 11  
 --filetype <filetype>, 12  
 --log-file <log\_file>, 11  
 --overwrite, 12  
 --pdb, 11  
 --quiet, 11  
 --skip-missing, 12  
 --verbose, 11  
 -c, 12  
 -d, 11  
 -f, 12  
 -q, 11  
 -s, 12  
 -t, 12  
 -v, 11  
 ENVNAMES, 12

## ymp-env-install command line option

-P, 12  
 --conda-env-spec <conda\_env\_spec>, 12  
 --conda-prefix <conda\_prefix>, 12  
 --dry-run, 12  
 --force, 12  
 --log-file <log\_file>, 12  
 --pdb, 12  
 --quiet, 12  
 --verbose, 12  
 -e, 12  
 -f, 12  
 -n, 12  
 -p, 12  
 -q, 12  
 -v, 12  
 ENVNAMES, 13

## ymp-env-list command line option

-P, 13  
 --all, 13  
 --dynamic, 13  
 --log-file <log\_file>, 13  
 --no-dynamic, 13  
 --no-static, 13  
 --pdb, 13  
 --quiet, 13  
 --reverse, 13  
 --sort <sort\_col>, 13  
 --static, 13  
 --verbose, 13  
 -a, 13  
 -q, 13  
 -r, 13  
 -s, 13

-v, 13

ENVNAMES, 13

## ymp-env-prepare command line option

-F, 14  
 -N, 14  
 -P, 14  
 --dryrun, 14  
 --force, 14  
 --forceall, 14  
 --keepgoing, 14  
 --lock, 14  
 --log-file <log\_file>, 14  
 --no-lock, 14  
 --nohup, 14  
 --notemp, 14  
 --pdb, 14  
 --printshellcmds, 14  
 --quiet, 14  
 --reason, 14  
 --rerun-incomplete, 14  
 --ri, 14  
 --shadow-prefix <shadow\_prefix>, 14  
 --touch, 14  
 --verbose, 14  
 -f, 14  
 -k, 14  
 -n, 14  
 -p, 14  
 -q, 14  
 -r, 14  
 -t, 14  
 -v, 14

TARGET\_FILES, 15

## ymp-env-remove command line option

-P, 15  
 --log-file <log\_file>, 15  
 --pdb, 15  
 --quiet, 15  
 --verbose, 15  
 -q, 15  
 -v, 15  
 ENVNAMES, 15

## ymp-env-run command line option

-P, 15  
 --log-file <log\_file>, 15  
 --pdb, 15  
 --quiet, 15  
 --verbose, 15  
 -q, 15  
 -v, 15  
 COMMAND, 16  
 ENVNAME, 16

## ymp-env-update command line option

-P, 16

--log-file <log\_file>, 16  
--pdb, 16  
--quiet, 16  
--reinstall <reinstall>, 16  
--verbose, 16  
-q, 16  
-v, 16  
ENVNAMES, 16  
ymp-init command line option  
-P, 16  
--log-file <log\_file>, 16  
--pdb, 16  
--quiet, 16  
--verbose, 16  
-q, 16  
-v, 16  
ymp-init-cluster command line option  
-P, 17  
--log-file <log\_file>, 17  
--pdb, 17  
--quiet, 17  
--verbose, 17  
--yes, 17  
-q, 17  
-v, 17  
-y, 17  
ymp-init-demo command line option  
-P, 17  
--log-file <log\_file>, 17  
--pdb, 17  
--quiet, 17  
--verbose, 17  
-q, 17  
-v, 17  
ymp-init-project command line option  
-P, 18  
--log-file <log\_file>, 18  
--pdb, 18  
--quiet, 18  
--verbose, 18  
--yes, 18  
-q, 18  
-v, 18  
-y, 18  
NAME, 18  
ymp-make command line option  
-F, 19  
-N, 19  
-P, 18  
--cores <CORES>, 19  
--dag, 19  
--debug, 19  
--debug-dag, 19  
--dryrun, 18  
--force, 19  
--forceall, 19  
--keepgoing, 18  
--lock, 18  
--log-file <log\_file>, 18  
--no-lock, 18  
--nohup, 19  
--notemp, 19  
--pdb, 18  
--printshellcmds, 18  
--quiet, 18  
--reason, 19  
--rerun-incomplete, 19  
--ri, 19  
--rulegraph, 19  
--shadow-prefix <shadow\_prefix>, 19  
--touch, 19  
--verbose, 18  
-f, 19  
-j, 19  
-k, 18  
-n, 18  
-p, 18  
-q, 18  
-r, 19  
-t, 19  
-v, 18  
TARGET\_FILES, 19  
ymp-show command line option  
-P, 20  
--help, 20  
--log-file <log\_file>, 20  
--pdb, 20  
--quiet, 20  
--source, 20  
--verbose, 20  
-h, 20  
-q, 20  
-s, 20  
-v, 20  
PROPERTY, 20  
ymp-stage command line option  
-P, 20  
--log-file <log\_file>, 20  
--pdb, 20  
--quiet, 20  
--verbose, 20  
-q, 20  
-v, 20  
ymp-stage-list command line option  
-P, 21  
--code, 21  
--log-file <log\_file>, 21  
--long, 21

---

```

--pdb, 21
--quiet, 21
--short, 21
--types, 21
--verbose, 21
-c, 21
-l, 21
-q, 21
-s, 21
-t, 21
-v, 21
STAGE, 21
ymp-submit command line option
-F, 22
-J, 22
-N, 22
-P, 21, 22
--args <ARGS>, 23
--cluster-cores <N>, 22
--command <CMD>, 22
--cores <N>, 23
--drmaa, 22
--dryrun, 21
--force, 22
--forceall, 22
--immediate, 22
--keepgoing, 22
--latency-wait <T>, 22
--lock, 22
--log-file <log_file>, 21
--max-jobs-per-second <N>, 22
--no-lock, 22
--nohup, 22
--notemp, 22
--pdb, 21
--printshellcmds, 21
--profile <NAME>, 22
--quiet, 21
--reason, 22
--rerun-incomplete, 22
--ri, 22
--scriptname <NAME>, 23
--shadow-prefix <shadow_prefix>, 22
--snake-config <FILE>, 22
--sync, 22
--touch, 22
--verbose, 21
--wrapper <CMD>, 22
-c, 22
-d, 22
-f, 22
-i, 22
-j, 23
-k, 22
-l, 22
-n, 21
-p, 21
-q, 21
-r, 22
-s, 22
-t, 22
-v, 21
TARGET_FILES, 23
YmpConfigError, 53
YmpConfigNotFound, 32
YmpException, 53
YmpNoStackException, 53
YmpObjectDescription (class in ymp.sphinxext),
    66
YmpRuleError, 53
YmpStage (class in ymp.sphinxext), 67
YmpStageError, 54
YmpSystemError, 54
YmpUsageError, 54
YmpWorkflowError, 54

```