



pybio: basic genomics toolset

Installation

Quick Start

Examples

Adding Ensembl genomes

Adding custom genomes

Retrieving genomic sequences

Annotating genomic positions

aimux module

Dependencies

Genomic coordinates

Authors

Issues and Suggestions

## pybio: basic genomics toolset

**pybio** is a Python framework for handling genomics operations and a direct interface to Ensembl genome assemblies and annotations.

Downloading an Ensembl genome + annotation:

```
Unset
# install over PyPi
pip install pybio

# install directly from this repository
pip install git+https://github.com/grexor/pybio.git@master

# download homo sapiens genome
pybio genome homo_sapiens
```

Features include genome+annotation download from Ensembl and processing with STAR and salmon, support of Fasta, Fastq and bedGraph file formats, motif sequence searches.

## Installation

The easiest way to install **pybio** is running:

Unset

```
pip install pybio
```

Note that on some systems, **pip** is installing the executable scripts under `~/.local/bin`. However this folder is not in the PATH which will result in “command not found” if you try to run “pybio” on the command line. To fix this, please execute “`export PATH=$PATH:~/.local/bin`” (and add this to your `.profile`). Another suggestion is to install inside a virtual environment (using `virtualenv`).

If you would like instead to **install the latest developmental version** from this repository:

Unset

```
# clone pybio GitHub repository
git clone https://github.com/grexor/pybio.git

# install
pip install .
```

## Quick Start

pybio is strongly integrated with Ensembl and provides genomic loci search for diverse annotated features (genes -> transcripts -> exons + 5UTR + 3UTR).

Let's say we are interested in the human genome. First download and prepare the genome with a single command:

Unset

```
pybio genome homo_sapiens
```

Searching a genomic position for features is easy in python, for example:

Python

```
import pybio
result = pybio.core.genomes.annotate("homo_sapiens", "1", "+", 11012344)
genes, transcripts, exons, UTR5, UTR3 = result
```

This will return a list of feature objects (genes, transcripts, exons, 3'-UTR and 5'-UTR) (check [pybio/core/genomes.py](#) classes to see details of these objects).

If you would like to know all genes that span the provided position, you could then write:

```
Python
for gene in genes:
    print(gene.gene_id, gene.gene_name, gene.start, gene.stop)
```

And to list all transcripts of each gene, you could extend the code like this:

```
Python
for gene in genes:
    print(gene.gene_id, gene.gene_name, gene.start, gene.stop)
    for transcript in gene.transcripts:
        print(transcript.transcript_id)
```

However you could also start directly with transcripts, and print to which genes are transcripts assigned to:

```
Python
for transcript in transcripts:
    print(transcript.gene.gene_id, transcript.transcript_id)
```

And an intuitive **graph representation of relationships** between feature objects:

```
Unset
gene <-> transcript_1 <-> exon_1
                        <-> exon_2
                        ...
                        <-> utr5
                        <-> utr3
<-> transcript_2 <-> exon_1
                        <-> exon_2
                        ...
                        <-> utr5
                        <-> utr3
```

Representation of relationships between feature objects:

```
Unset
gene = Gene instance object
    .transcripts = list of all transcript objects of the gene

transcript = Transcript instance object
    .gene = points to the gene of the transcript
    .exons = list of all exon objects of the transcript
    .utr5/utr3 = points to the UTR5 / UTR3 of the transcript

exon = Exon instance object
    .transcript = points to the transcript of the exon

utr5/utr3 = Utr5 / Utr3 instance object
    .transcript = points to the transcript of the UTR5/UTR3
```

## Examples

Here we provide basic pybio usage examples.

### Adding Ensembl genomes

To download Ensembl genomes simply run a few commands on the command line. For example:

```
Unset
# downloads Ensembl homo_sapiens assembly and annotation (latest version)
pybio genome homo_sapiens

# downloads Ensembl homo_sapiens assembly and annotation (version 109)
pybio genome homo_sapiens 109

# list all available elephant genomes matching *elephant*
pybio genome elephant
```

The above will download the FASTA sequence and GTF annotation. If you have **STAR** and **salmon** installed on your system, **pybio** will also build an index of the genome for both.

Data will be stored in the folder specified in the file `pybio.config`. The genomes folder structure is as follows:

```
Unset
homo_sapiens.assembly.ensembl109      # FASTA files of the genome
homo_sapiens.annotation.ensembl109    # Annotation in GTF and TAB format
homo_sapiens.assembly.ensembl109.star  # STAR index, GTF annotation aware
homo_sapiens.transcripts.ensembl109    # Transcriptome, Ensembl cDNA FASTA
homo_sapiens.transcripts.ensembl109.salmon # Salmon index of the transcriptome
```

pybio also supports the download of Ensembl Genomes (Ensembl Fungi, Ensembl Plants, Ensembl Protists, Ensembl Metazoa). You simply provide the name of the species on the command line to automatically download the genome, the assembly and prepare STAR and salmon indices.

For example, to download the latest version of the Dictyostelium discoideum genome, you would write:

```
Unset
# search for genomes with "dicty" in the name of the species or description
# if only one species found, directly go ahead and process
pybio genome dicty

# also directly providing the exact genome species works
pybio genome dictyostelium_discoideum
```

Another example is download the latest Arabidopsis thaliana genome:

```
Unset
pybio genome arabidopsis_thaliana
```

To see all available species, simply run `pybio species`. Moreover, to see all available arabidopsis genomes, you could run:

```
Unset
pybio species arabidopsis

arabidopsis_halleri  Ahal2.2  ensemblgenomes  plants ensemblgenomes56
arabidopsis_lyrata   v.1.0    ensemblgenomes  plants ensemblgenomes56
arabidopsis_thaliana TAIR10    ensemblgenomes  plants ensemblgenomes56
```

## Adding custom genomes

To add custom genomes to the **pybio** environment, you would need a FASTA (assembly) and GTF (annotation) files and set a genome version (custom label for your genome version). An example run would be:

```
Unset
# add custom genome with "species" name, version "v1" from FASTA and GTF
pybio genome species -fasta sample.fasta -genome_version v1 -gtf sample.gtf
```

## Retrieving genomic sequences

To retrieve stretches of genomic sequence, we use the `seq(genome, chr, strand, position, upstream, downstream)` method:

```
Python
import pybio
seq = pybio.core.genomes.seq("homo_sapiens", "1", "+", 450000, -20, 20)
```

The above command fetches the chr 1 sequence from 450000-20..450000+20, the resulting sequence is of length 41, TACCCTGATTCTGAAACGAAAAAGCTTTACAAAATCCAAGA.

## Annotating genomic positions

Given a genomic position, we can quickly retrieve features that are present at the position: the gene, transcript, exon and UTR 5/3 information. If there are several features (genes, transcripts, exons, UTR regions) at the specified position, they are all reported by **pybio**.

```
Python
# annotate position
result = pybio.core.genomes.annotate("hg38", "1", "+", 11012344)
# expand the data to features
genes, transcripts, exons, utr5, utr3 = result
# print all genes that cover the position
for gene in genes:
    print(gene.gene_id, gene.gene_name, gene.start, gene.stop)

# [pybio] loading genome annotation for homo_sapiens with Ensembl version 109
# ENSG00000120948, TARDBP, 11012343, 11030527
```

We can also easily access all transcripts of each gene with `gene.transcripts` and all exons of each transcript with `transcript.exons`.

## aimux module

Simple module to demultiplex short-read data using samples+barcodes annotation. The input is a TAB delimited file of samples and barcodes, for example (`annotation.tab`):

```
Python
sample_id    barcode1    barcode2
sample1      ATTCGT      ACC
sample2      AGGTCC      ATT
...
```

Having R1, R2, I1 and I2 fastq files, aimux can be run with:

```
Python
pybio aimux -r1 r1.fastq.gz -r2 r2.fastq.gz -i1 i1.fastq.gz -i2 i2.fastq.gz
-annotation annotation.tab -barcodes barcode1:i1:RRRRRR_0_m1,barcode2:i2:RRR_0
-stats aimux.stats -output samples
```

The above will take the I1 sequence and match it with a reversed barcode1 of length 6 (RRRRRR), starting at position 0 in the I1 read (`_0`) and allowing 1 mismatch (`_m1`). At the same time, it will check the I2 sequence against barcode2, requiring a perfect match (no `_m` specified) and again starting at position 0. Output is written to `samples` (-output) and statistics to the file `aimux.stats`.

## Dependencies

Basic dependencies include [pysam](#), [numpy](#) and [samtools](#) and should be installed automatically by pip when you install pybio over `pip install pybio`.

Optional dependencies include [STAR](#) and [salmon](#) if you would like to build genome/transcriptome indices and align reads.

## Genomic coordinates

All genomic coordinates we operate with inside pybio are 0-indexed left+right inclusive. This means, when we say for example 100-103, this would include coordinates 100, 101, 102 and 103. The first coordinate of any given sequence is 0.

## Important

Refseq and Ensembl GTF files are 1-indexed. When we read files from refseq/ensembl, we subtract 1 on all coordinates to keep this in line with other coordinate structures inside pybio (which are all 0-indexed).

## Authors

pybio is developed and supported by [Gregor Rot](#).

## Issues and Suggestions

Use the [GitHub repository issues page](#) to report issues and leave suggestions.