piomart Documentation

Release latest

Sep 25, 2018

Contents

piomart is on pip. This is the most recommended way to install it.

```
pip install piomart
```

CHAPTER 1

Manual installation

```
git clone git@gitlab.com:diegocrespo/piomart.git
```

```
python setup.py install
```

the required packages for this software are

- **python packages (python>=3.5)**
  - docopt>=0.6.2
  - setuptools>=30.0
  - numpy>=1.14.3
  - responses>=0.9.0
  - pandas>=0.23.0
  - requests>=2.18.4

CHAPTER 2

`gtf` - Downloading gtf files

Piomart supports the downloading of gtf files from the ensembl database. A common usecase is downloading the Homo_sapiens.GRCh38.93.gtf.gz from ensembl. This can be done by specifying

python piomart.py gtf –species homo_sapiens

python piomart.py gtf –species homo_sapiens –release release-93 -iu

The above command is a more complex use case. here we specify the specif release we want for our gtf file. -u tells piomart to unzip the resulting .gz file that is downloaded. -i is an interactive feature. Some analyses require a special version of the gtf. abinitio, chr, chr_patch_hapl_scaff, etc. -i will present youwith all the files in the gtf ftp directory for your species. You can download all of them, or pick (one at a time) which files you would like to download

{0: 'Homo_sapiens.GRCh38.93.abinitio.gtf.gz', 1: 'Homo_sapiens.GRCh38.93.chr.gtf.gz', 2: 'Homo_sapiens.GRCh38.93.chr_patch_hapl_scaff.gtf.gz', 3: 'Homo_sapiens.GRCh38.93.gtf.gz', 4: 'All Files'}

CHAPTER 3

Working offline

3.1 Introduction

piomart is a command line tool that is designed to make the process of annotating differential expression results easier. It can be used in a notebook, on the command line, in a Snakefile etc. Below is an example of using it on a command line to annotated a csv file of differential expression results. At the end is an example how it would look integrated into a Snakefile.

3.2 Downloading the gtf file

To download the gtf file use the *gtf* command. To create the json file needed the gtf file needs to be unzipped

```
piomart gtf –species homo_sapiens -u
```

While this is all you need to download and unzip over ftp the gtf file, it is a good idea to specify the release as well to ensure reproducibility, using the *–release <ver>* option. If no release is specified, the data is pulled from the http://ftp.ensembl.org/pub/current_gtf/ directory which is updated every time a new release is published. it is also a good idea to give a name to the file using –output, or else you will be left with whatever the name of the file in the directory is. Currently it's Homo_sapiens.GRCh38.93.gtf.gz

3.3 Creating the json file

After the gtf file for the species of interest is download, it's time to use to create the json file which is parsed to get the gene symbols and other information the command

```
piomart json -f Homo_sapiens.GRCh38.93.gtf -o homo_sapiens.json
```

Will parse the gtf file *Homo_sapiens.GRCh38.93.gtf.gz* and create the json file *homo_sapiens.json* if no output file is specified *.json* will be added to the input file creating Homo_sapiens.GRCh38.93.gtf.json so it is recommended to specify an output.

3.4 Using the info Command

The info command is useful for getting information on on one or two genes. Using our previously downloaded *homo_sapiens.json*

piomart info ENSG00000278384 –offline -f homo_sapiens.json

will produce the output in the terminal

gene_id ENSG00000278384

gene_version 1

gene_name AL354822.1

gene_source ensembl

gene_biotype protein_coding

seqname GL000218.1

source ensembl

feature gene

start 51867

end 54893

score .

strand -

frame .

Any number of genes can be specified using spaces. *info* can also take a text file which contains id's as well.

3.5 Appending data to a csv file using dataframe

When used in offline mode the *dataframe* command will append up to all the information for each gene in the gtf file. The most common way to use it

piomart dataframe MyCsv.csv –offline -f homo_sapiens.json –columns=gene_name,gene_id –output=Mycsv_with_symbols.csv

piomart assumes that the index column is the first column, and that that column contains Ensembl ids. If the first column is not the index column. It can be specified with –index using either the column name, or integer.

If your Ensembl ids in the csv column contain versions, information about that will only be appended if the version of the gene in the csv file matches the data in the json file. If that is not the case, then the original Ensembl id with version will be returned. If all's you are interested is getting the information regardelss of version *–inexact* can be passed on the command line. –inexact will convert all Ensemble ids with versions to just Ensemble ids. If the id is in the json file then the information will be appended. If it does not exist in the json file, it usually means the Ensembl id has been deprecated and every field after gene_id and gene_name that is specified will have 'deprecated' in it. Instead of a gene_name the id will be returned with '_d' appended to it.

There is one unique case with paralogs. ENSG00000197976.11 and gene ENSG00000197976.11_PAR_Y are both the same gene, but one is a paralog. Both genes will be present in the returned file, but insterad of "_d" being appended to the gene "_PAR_Y" will be returned. So you will see "ENSG00000197976.11_PAR_Y" in your csv file. All other fields will contain the world "paralog" in them.

3.6 *Bonus* Using Piomart in a Snakefile

Below is an example of using piomart on some deseq2 results to generate annoated csv files

deseq_files = [“onset_deseq2_results”, “cortical_deseq2_results”,”cag_repeat_deseq2_results”]

deseq_out = expand(“{file}_annotated.csv”, file=deseq_files)

rule all:

input: deseq_out

rule download_gtf:

output: “homo_sapiens_grch38.json”

shell: “piomart gtf -u –species homo_sapiens –release release-93 –output homo_sapiens_grch38.gtf “

“&& piomart json -f homo_sapiens_grch38.gtf -o {output[0]}”

rule annotated_deseq_results:

input: “homo_sapiens_grch38.json”,

“{file}.csv”

output: “{file}_annotated.csv”

shell: “piomart dataframe {input[1]} –offline -f {input[0]} –columns=gene_name –output={output[0]}”

CHAPTER 4

module - importing piomart

it is possible to piomart in a script or notebook.

```
from piomart.piomart import EnsembleClient as pio
```

The function *add_to_dataframe* takes a pandas dataframe, and a list of colnames which correspond to the names of the columns in a gtf file as well as the values in the attribute column. So long as the index column is the gene ids, *add_to_dataframe* will append the gene symbols. If the symbol's version doesn't match the version in the gtf file, then the Ensembl id is used instead of the gene symbol. If the gene symbol without the id is not in the gtf file, this usually means it is deprecated. Therefore the Ensembl id is used, and "_d" is appended to it to give the user an indication that they should check out the id

if you have you have a counts matrix you would like to add gene name information to, you can add it to the dataframe by creatinga small function that looks like this.

```
def add_gene_info(dataframe, columns=[]): func = pio(True, "homo_sapiens.json") df =
    func.add_to_dataframe(dataframe
        ,columns)
    df.set_index("gene_name",drop=True,inplace=True) return df
```

and then passing the df with the column of interest to the function

```
df = add_gene_info(my_df, ["gene_name"])
```

which will change the dataframe as shown below.

gene_ids | sample_1 | sample_2 | sample_3 |

|———————+———————-+———————-+———————-| | ENSG000001 | 100 | 200 | 300 | | ENSG000002 | 200 | 100 | 300 | | ENSG000003 | 200 | 400 | 300 | | ENSG000004 | 300 | 400 | 100 |

gene_ids | sample_1 | sample_2 | sample_3 | gene_name |

|————————+————————-+————————-+————————-|————————| | ENSG000001 | 100 | 200 | 300 | GENE1 | | ENSG000002 | 200 | 100 | 300 | GENE2 | | ENSG000003 | 200 | 400 | 300 | GENE3 | | ENSG000004 | 300 | 400 | 100 | GENE4 |

func = pio(True, "homo_sapiens.json") tells piomart that you want to use it offline, with the homo_sapiens.json file (created using the *gtf* and *json* command)

df = func.add_to_dataframe(dataframe,columns) simply passes your dataframe (indexed to ensembl ids!) and columns of interest to the add_to_dataframe method of the ensembl client class.