COMMAND>_ Documentation

Marco Moretto

Feb 05, 2019

Table of Contents

COMMAND>_ is a web-based application used to download, collect and manage gene expression data from public databases.

Main features are.

- Easy installation and update using Docker Compose technology.
- Graphical User Interface (GUI) for parsing and importing gene expression data.
- Default Python scripts for easy parsing/importing of the most common microarray platforms (Affymetrix, Nimblegen, two-colors, etc.) and dedicated scripting editor for allowing flexible importing of any kind of gene expression data.
- Automatic pre-processing (downloading, trimming, mapping and counting) of bulk RNA-Seq data.
- Exporting of the collected data.

Note: Give it a try on https://command.fmach.it:4242 using:

- username: `guest`
- password: `demo`

Check out the *Use Cases*!

CHAPTER 1

What is COMMAND>_?

COMMAND>_[#f1]_ is an acronym that stands for **COM**pendia **MAN**agement **D**esktop. It is the software used for the creation of several gene expression compendia such as COLOMBOS[2] and VESPUCCI[3]. Despite being used since 2010 it has been made publicly available for anyone to use only in 2018, after having been completely rewritten. COMMAND>_ was originally conceived for the collection (and integration) of prokaryotes microarray experiments. As time goes by it has been evolved to allow also RNA-seq experiment to be imported and other species to be managed. With the current implementation COMMAND>_ is still meant for gene expression data collection but can be easily extended to support other kind of quantitative measurement technology (have a look at *COMMAND>_for developers*).

COMMAND>_ is a Python web application developed using the Django framework for the backend, while the web interface has been developed using ExtJS with a look and feel typical of desktop applications. With COMMAND>_ you can search and download experiment from public gene expression databases, such as GEO , ArrayExpress or SRA, parse downloaded files to extract only valuable information, preview parsed data and import experiment data into a database. The pivotal point is the usage of custom Python scripts to mine only the relevant information. Scripts can be created or modified directly within the interface and are responsible to parse input files and populate each part of the **data model** (see *Database schema*), i.e. measurement data and meta-data for *experiment*, *platforms* and *samples*.

For microarray platforms it would be necessary to map probes to genes but before this step genes have first to be imported. COMMAND>_ allow to perform both these steps. For the latter it would be simply a matter of uploading a FASTA file with gene sequences (see data_collection), while for the former a BLAST alignment followed by a two-step filtering will be performed. In this way the microarray gets annotated with the latest available information enhancing the homogeneity since all microarrays will be annotated using the same gene list (see also map_feature).

References

---

[2] Moretto, M. et al. (2015). COLOMBOS v3. 0: leveraging gene expression compendia for cross-species analyses. *Nucleic acids research*, 44(D1), D620-D623.

[3] Moretto, M. et al. (2016). VESPUCCI: exploring patterns of gene expression in grapevine. *Frontiers in plant science*, 7, 633.

CHAPTER 2

Getting start with COMMAND>_

2.1 Getting my user id and password

If you are using the public COMMAND>_ instance on https://command.fmach.it:4242 you can login using:

- username: `guest`
- password: `demo`

This is a user with restricted privileges meant for demonstration purpose only. If you have your running instance of COMMAND>_ (see *Deploy*) you will be able to first login using:

- username: `admin`
- password: `admin`

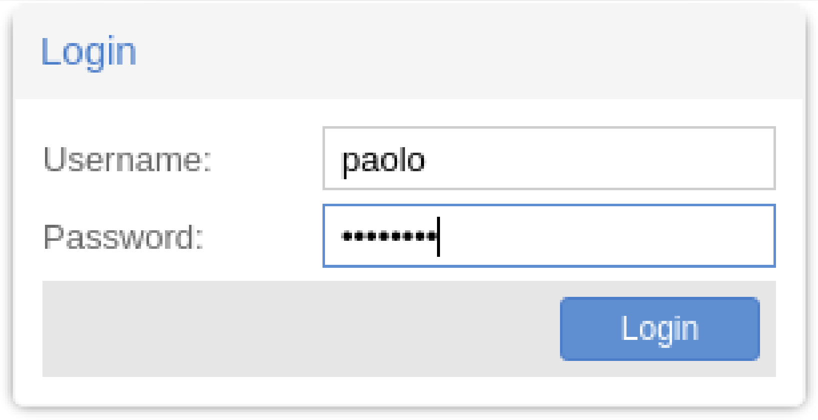
Now you can change the admin password, create new users and assign them privileges following the instructions in *Admin*.

2.2 Set up and select a compendium

The first thing to do is creating a new empty compendium. Go to Admin (top bar) > Compendium Manager > Create Compendium (bottom-left corner + icon) and follow the instructions at *Admin*.

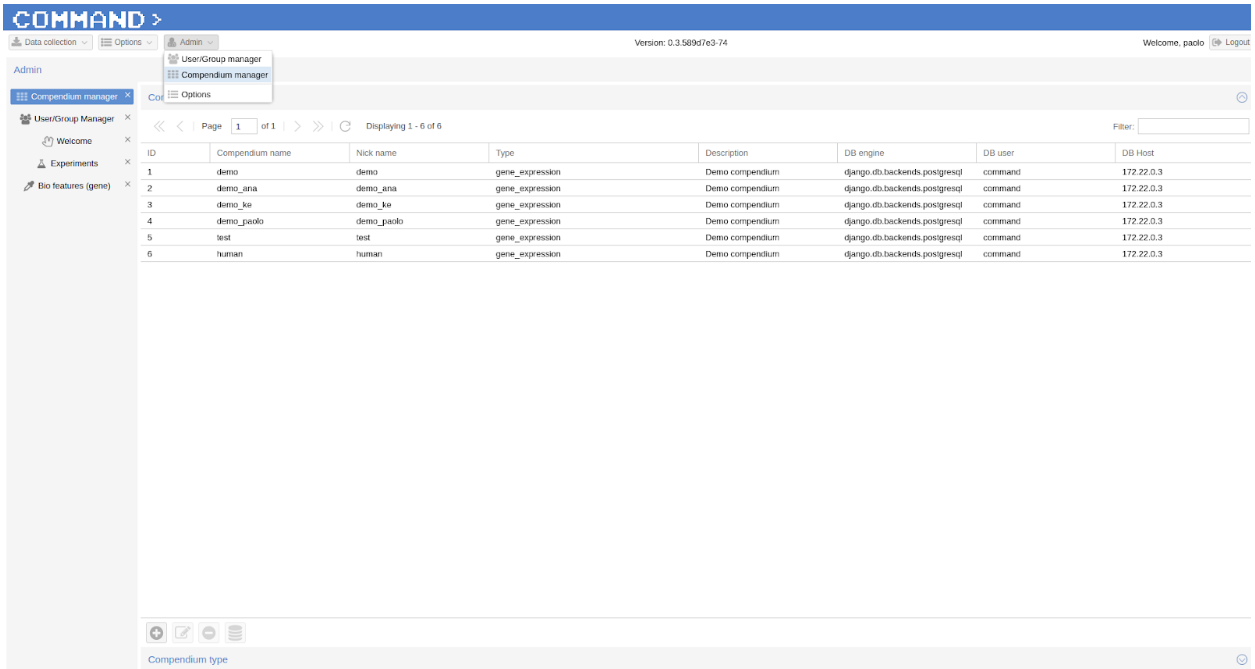
Now that a new compendium has been set up you need to retrieve a FASTA file containing the gene ids and sequences for the species you want to study.

Tip: For example you can visit the NCBI Nucleotide database and get the coding sequences for the organism of interest. This file is mandatory for blasting and mapping respectively in either microarray or RNA-Seq experiments. In order to import it into COMMAND>_ go to Data collection (on the top-left corner) > Biological features, then select Import biological features from the bottom-left + icon.

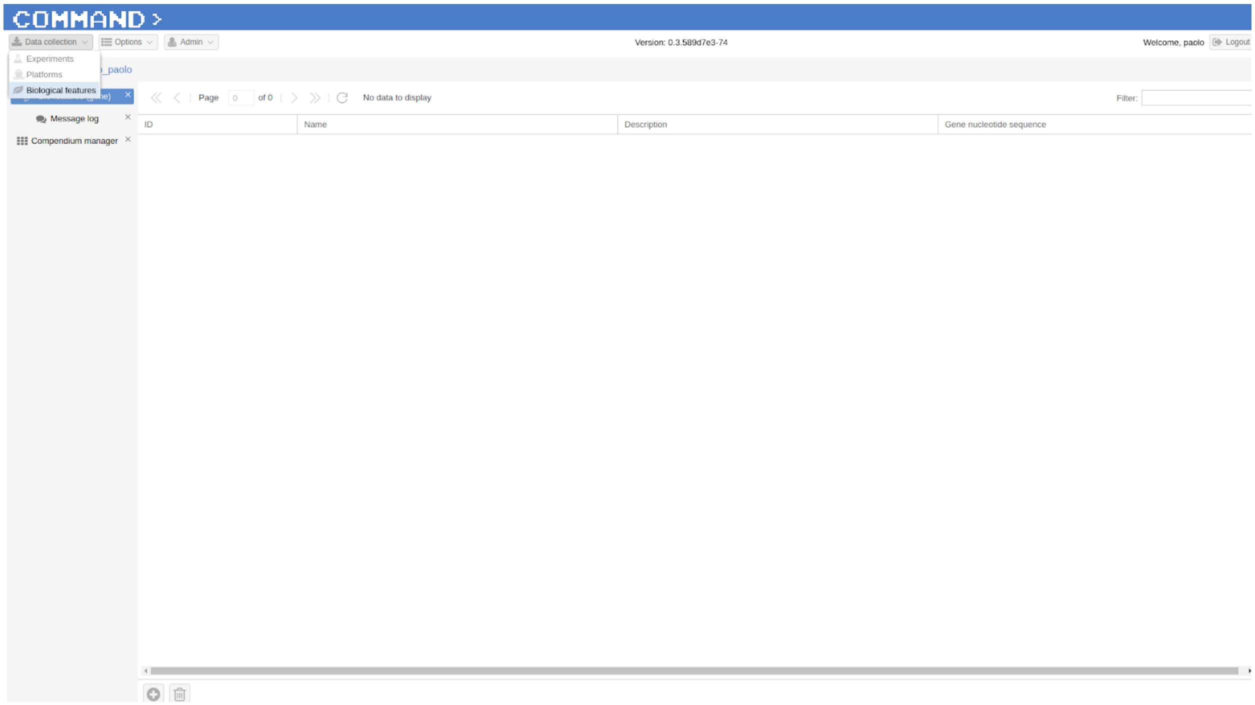


Username: paolo

Password: ••••••••



ID	Compendium name	Nick name	Type	Description	DB engine	DB user	DB Host
1	demo	demo	gene_expression	Demo compendium	django.db.backends.postgresql	command	172.22.0.3
2	demo_ana	demo_ana	gene_expression	Demo compendium	django.db.backends.postgresql	command	172.22.0.3
3	demo_ke	demo_ke	gene_expression	Demo compendium	django.db.backends.postgresql	command	172.22.0.3
4	demo_paolo	demo_paolo	gene_expression	Demo compendium	django.db.backends.postgresql	command	172.22.0.3
5	test	test	gene_expression	Demo compendium	django.db.backends.postgresql	command	172.22.0.3
6	human	human	gene_expression	Demo compendium	django.db.backends.postgresql	command	172.22.0.3



Now your gene annotation file has been imported and you can start looking for interesting experiments (both microarray and RNA-Seq) related to the organism of interest.

2.3 Searching public databases

After a new empty compendium has been created and a species of interest selected the user can start looking for collections of samples (from one or more experiments) from public databases (GEO, ArrayExpress or, in case of RNA-Seq experiments, SRA): Go to > Data collection (on the top-left corner) then > Experiments > New Experiment (on the bottom right corner) > from public DB.

- In the Search options field of the dialog ‘Download from Public DB’ select the DB (here GEO) and the term of interest, either a description (e.g. Leukemia b-cell, Vitis vinifera, erc.) or directly a GSE ID.
- From the list select an experiment of interest and click the download button.

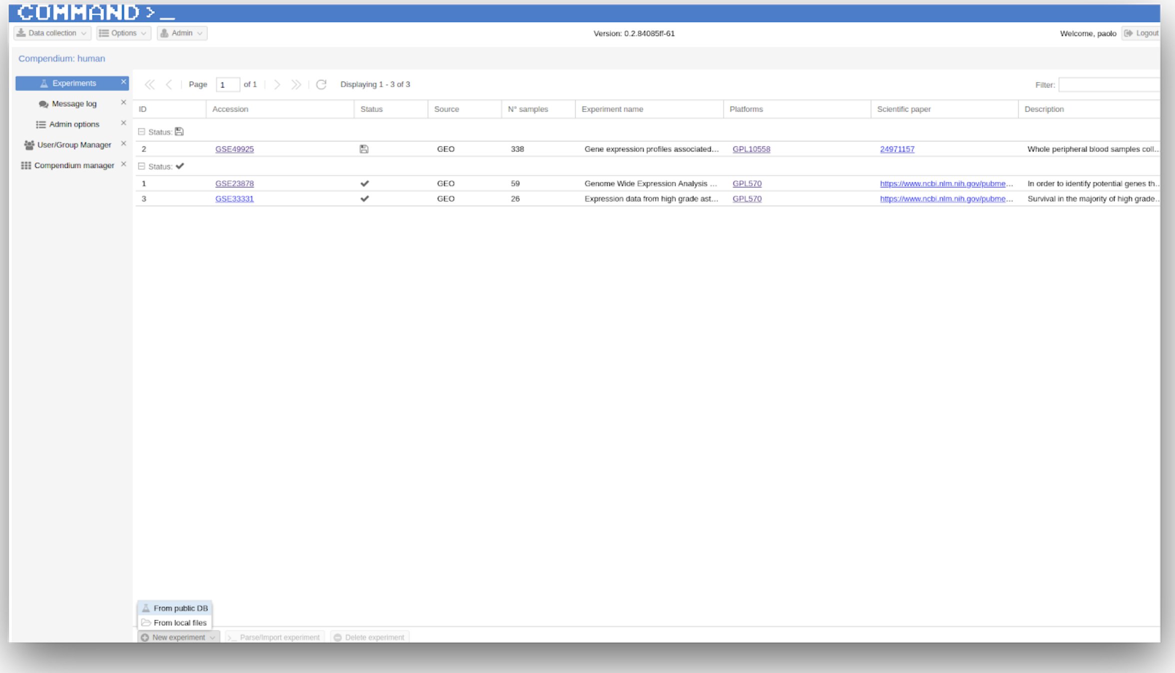
Tip: You can download multiple experiments at the same time.

- After a while, depending of the number of samples in the selected experiment(s) you have your experiment downloaded.

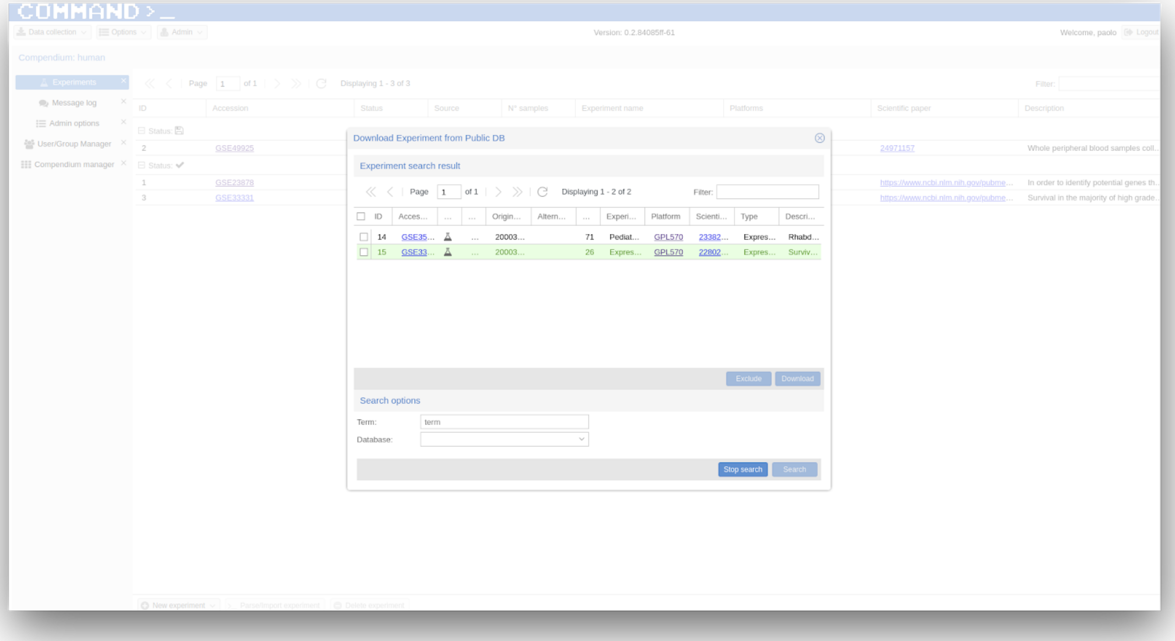
Tip:

- Check Message log frequently.
- Inspect the Experiments section to see which experiments are available, yet to be parsed or already imported.

Now you can start parse and import some experiment (see *Use Cases*).



ID	Accession	Status	Source	N° samples	Experiment name	Platforms	Scientific paper	Description
Status: 🖫								
2	GSE49925	🖫	GEO	338	Gene expression profiles associated…	GPL10558	24971157	Whole peripheral blood samples coll…
Status: ✓								
1	GSE23878	✓	GEO	59	Genome Wide Expression Analysis …	GPL570	https://www.ncbi.nlm.nih.gov/pubme…	In order to identify potential genes th…
3	GSE33331	✓	GEO	26	Expression data from high grade ast…	GPL570	https://www.ncbi.nlm.nih.gov/pubme…	Survival in the majority of high grade…



☐	ID	Acces…	…	…	Origin…	Altern…	…	Experi…	Platform	Scienti…	Type	Descri…
☐	14	GSE35…	⚗	…	20003…		71	Pediat…	GPL570	23382…	Expres…	Rhabd…
☐	15	GSE33…	⚗	…	20003…		26	Expres…	GPL570	22802…	Expres…	Surviv…

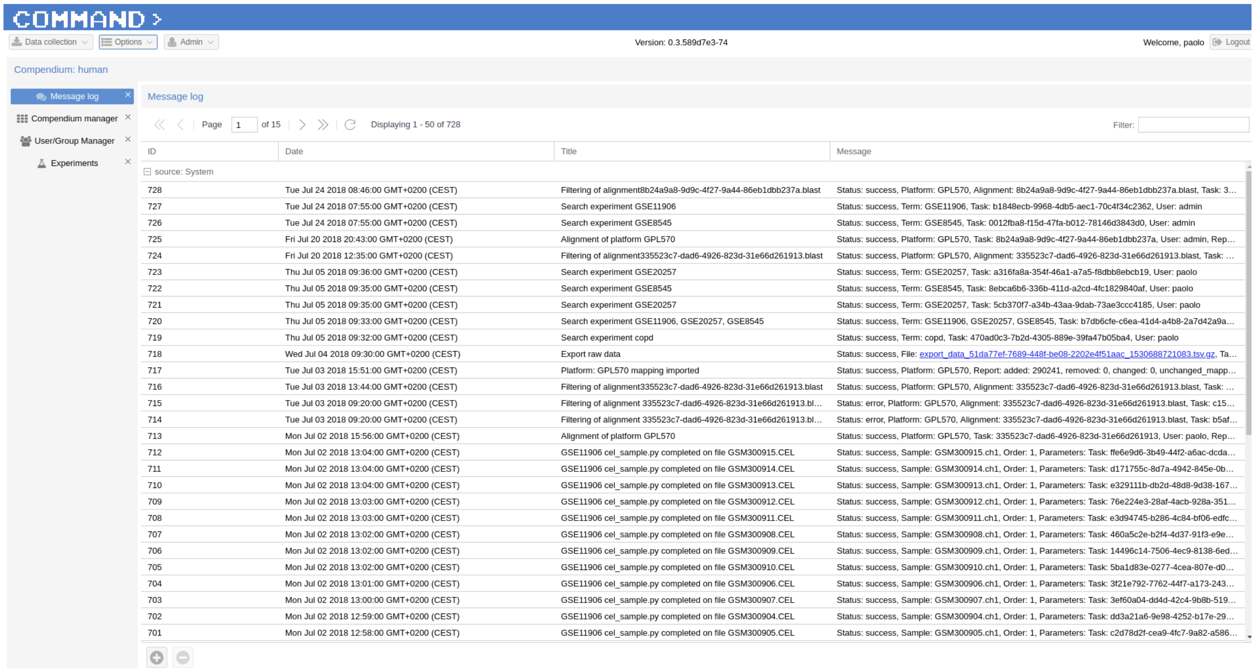
Term: term

Database:

CHAPTER 3

Message log

The Message Log page (Top > Options > Message Log) allows the user to take an eye on every activity of COMMAND>_. Check it frequently!



source: System

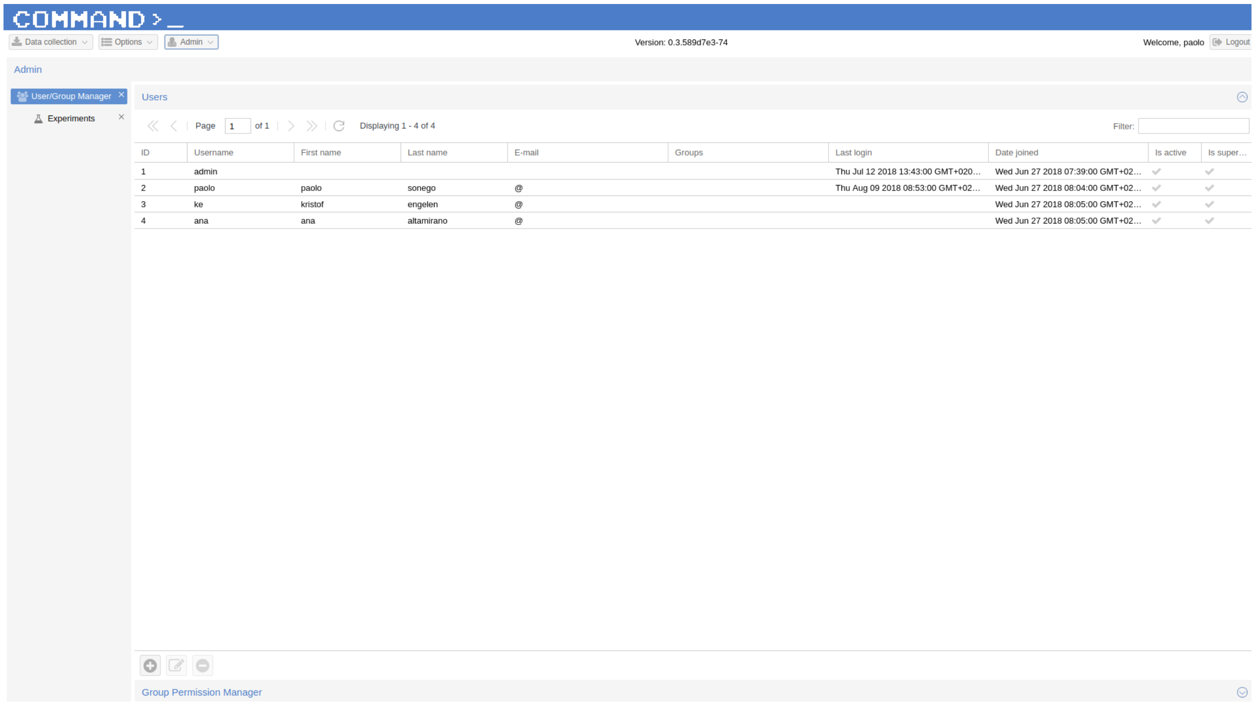
ID	Date	Title	Message
728	Tue Jul 24 2018 08:46:00 GMT+0200 (CEST)	Filtering of alignment8b24a9a8-9d9c-4f27-9a44-86eb1dbb237a.blast	Status: success, Platform: GPL570, Alignment: 8b24a9a8-9d9c-4f27-9a44-86eb1dbb237a.blast, Task: 3…
727	Tue Jul 24 2018 07:55:00 GMT+0200 (CEST)	Search experiment GSE11906	Status: success, Term: GSE11906, Task: b1848ecb-9968-4db5-aec1-70c4f34c2362, User: admin
726	Tue Jul 24 2018 07:55:00 GMT+0200 (CEST)	Search experiment GSE8545	Status: success, Term: GSE8545, Task: 0012fba8-f15d-47fa-b012-78146d3843d0, User: admin
725	Fri Jul 20 2018 20:43:00 GMT+0200 (CEST)	Alignment of platform GPL570	Status: success, Platform: GPL570, Task: 8b24a9a8-9d9c-4f27-9a44-86eb1dbb237a, User: admin, Rep…
724	Fri Jul 20 2018 12:35:00 GMT+0200 (CEST)	Filtering of alignment335523c7-dad6-4926-823d-31e66d261913.blast	Status: success, Platform: GPL570, Alignment: 335523c7-dad6-4926-823d-31e66d261913.blast, Task: …
723	Thu Jul 05 2018 09:36:00 GMT+0200 (CEST)	Search experiment GSE20257	Status: success, Term: GSE20257, Task: a316fa8a-354f-46a1-a7a5-f8dbb8ebcb19, User: paolo
722	Thu Jul 05 2018 09:35:00 GMT+0200 (CEST)	Search experiment GSE8545	Status: success, Term: GSE8545, Task: 8ebca6b6-336b-411d-a2cd-4fc1829840af, User: paolo
721	Thu Jul 05 2018 09:35:00 GMT+0200 (CEST)	Search experiment GSE20257	Status: success, Term: GSE20257, Task: 5cb370f7-a34b-43aa-9dab-73ae3ccc4185, User: paolo
720	Thu Jul 05 2018 09:33:00 GMT+0200 (CEST)	Search experiment GSE11906, GSE20257, GSE8545	Status: success, Term: GSE11906, GSE20257, GSE8545, Task: b7db6cfe-c6ea-41d4-a4b8-2a7d42a9a…
719	Thu Jul 05 2018 09:32:00 GMT+0200 (CEST)	Search experiment copd	Status: success, Term: copd, Task: 470ad0c3-7b2d-4305-889e-39fa47b05ba4, User: paolo
718	Wed Jul 04 2018 09:30:00 GMT+0200 (CEST)	Export raw data	Status: success, File: export_data_51da77ef-7689-448f-be08-2202e4f51aac_1530688721083.tsv.gz, Ta…
717	Tue Jul 03 2018 15:51:00 GMT+0200 (CEST)	Platform: GPL570 mapping imported	Status: success, Platform: GPL570, Report: added: 290241, removed: 0, changed: 0, unchanged_mapp…
716	Tue Jul 03 2018 13:44:00 GMT+0200 (CEST)	Filtering of alignment335523c7-dad6-4926-823d-31e66d261913.blast	Status: success, Platform: GPL570, Alignment: 335523c7-dad6-4926-823d-31e66d261913.blast, Task: …
715	Tue Jul 03 2018 09:20:00 GMT+0200 (CEST)	Filtering of alignment 335523c7-dad6-4926-823d-31e66d261913.bl…	Status: error, Platform: GPL570, Alignment: 335523c7-dad6-4926-823d-31e66d261913.blast, Task: c15…
714	Tue Jul 03 2018 09:20:00 GMT+0200 (CEST)	Filtering of alignment 335523c7-dad6-4926-823d-31e66d261913.bl…	Status: error, Platform: GPL570, Alignment: 335523c7-dad6-4926-823d-31e66d261913.blast, Task: b5af…
713	Mon Jul 02 2018 15:56:00 GMT+0200 (CEST)	Alignment of platform GPL570	Status: success, Platform: GPL570, Task: 335523c7-dad6-4926-823d-31e66d261913, User: paolo, Rep…
712	Mon Jul 02 2018 13:04:00 GMT+0200 (CEST)	GSE11906 cel_sample.py completed on file GSM300915.CEL	Status: success, Sample: GSM300915.ch1, Order: 1, Parameters: Task: ffe6e9d6-3b49-44f2-a6ac-dcda…
711	Mon Jul 02 2018 13:04:00 GMT+0200 (CEST)	GSE11906 cel_sample.py completed on file GSM300914.CEL	Status: success, Sample: GSM300914.ch1, Order: 1, Parameters: Task: d171755c-8d7a-4942-845e-0b…
710	Mon Jul 02 2018 13:04:00 GMT+0200 (CEST)	GSE11906 cel_sample.py completed on file GSM300913.CEL	Status: success, Sample: GSM300913.ch1, Order: 1, Parameters: Task: e329111b-db2d-48d8-9d38-167…
709	Mon Jul 02 2018 13:03:00 GMT+0200 (CEST)	GSE11906 cel_sample.py completed on file GSM300912.CEL	Status: success, Sample: GSM300912.ch1, Order: 1, Parameters: Task: 76e224e3-28af-4acb-928a-351…
708	Mon Jul 02 2018 13:03:00 GMT+0200 (CEST)	GSE11906 cel_sample.py completed on file GSM300911.CEL	Status: success, Sample: GSM300911.ch1, Order: 1, Parameters: Task: e3d94745-b286-4c84-bf06-edfc…
707	Mon Jul 02 2018 13:02:00 GMT+0200 (CEST)	GSE11906 cel_sample.py completed on file GSM300908.CEL	Status: success, Sample: GSM300908.ch1, Order: 1, Parameters: Task: 460a5c2e-b2f4-4d37-91f3-e9e…
706	Mon Jul 02 2018 13:02:00 GMT+0200 (CEST)	GSE11906 cel_sample.py completed on file GSM300909.CEL	Status: success, Sample: GSM300909.ch1, Order: 1, Parameters: Task: 14496c14-7506-4ec9-8138-6ed…
705	Mon Jul 02 2018 13:02:00 GMT+0200 (CEST)	GSE11906 cel_sample.py completed on file GSM300910.CEL	Status: success, Sample: GSM300910.ch1, Order: 1, Parameters: Task: 5ba1d83e-0277-4cea-807e-d0…
704	Mon Jul 02 2018 13:01:00 GMT+0200 (CEST)	GSE11906 cel_sample.py completed on file GSM300906.CEL	Status: success, Sample: GSM300906.ch1, Order: 1, Parameters: Task: 3f21e792-7762-44f7-a173-243…
703	Mon Jul 02 2018 13:00:00 GMT+0200 (CEST)	GSE11906 cel_sample.py completed on file GSM300907.CEL	Status: success, Sample: GSM300907.ch1, Order: 1, Parameters: Task: 3ef60a04-dd4d-42c4-9b8b-519…
702	Mon Jul 02 2018 12:59:00 GMT+0200 (CEST)	GSE11906 cel_sample.py completed on file GSM300904.CEL	Status: success, Sample: GSM300904.ch1, Order: 1, Parameters: Task: dd3a21a6-9e98-4252-b17e-29…
701	Mon Jul 02 2018 12:58:00 GMT+0200 (CEST)	GSE11906 cel_sample.py completed on file GSM300905.CEL	Status: success, Sample: GSM300905.ch1, Order: 1, Parameters: Task: c2d78d2f-cea9-4fc7-9a82-a586…

CHAPTER 4

Admin

The admin interface is visible only to admin users that have complete access to COMMAND>_ functionalities and compendia.

4.1 Users/Group manager



Version: 0.3.589d7e3-74

Displaying 1 - 4 of 4

ID	Username	First name	Last name	E-mail	Groups	Last login	Date joined	Is active	Is super…
1	admin					Thu Jul 12 2018 13:43:00 GMT+020…	Wed Jun 27 2018 07:39:00 GMT+02…	✓	✓
2	paolo	paolo	sonego	@		Thu Aug 09 2018 08:53:00 GMT+02…	Wed Jun 27 2018 08:04:00 GMT+02…	✓	✓
3	ke	kristof	engelen	@			Wed Jun 27 2018 08:05:00 GMT+02…	✓	✓
4	ana	ana	altamirano	@			Wed Jun 27 2018 08:05:00 GMT+02…	✓	✓

Fig. 1: User manager page

The user menu allow to create, remove and modify users. Moreover, an admin user can assign users to groups and set privileges to them. Group privileges are compendium-specific, i.e. we can for example restrict access only to some compendia and avoid users belonging to a group to see the others. For those compendia we can limit some functionalities, for example we could avoid users to run Python script or import experiments.

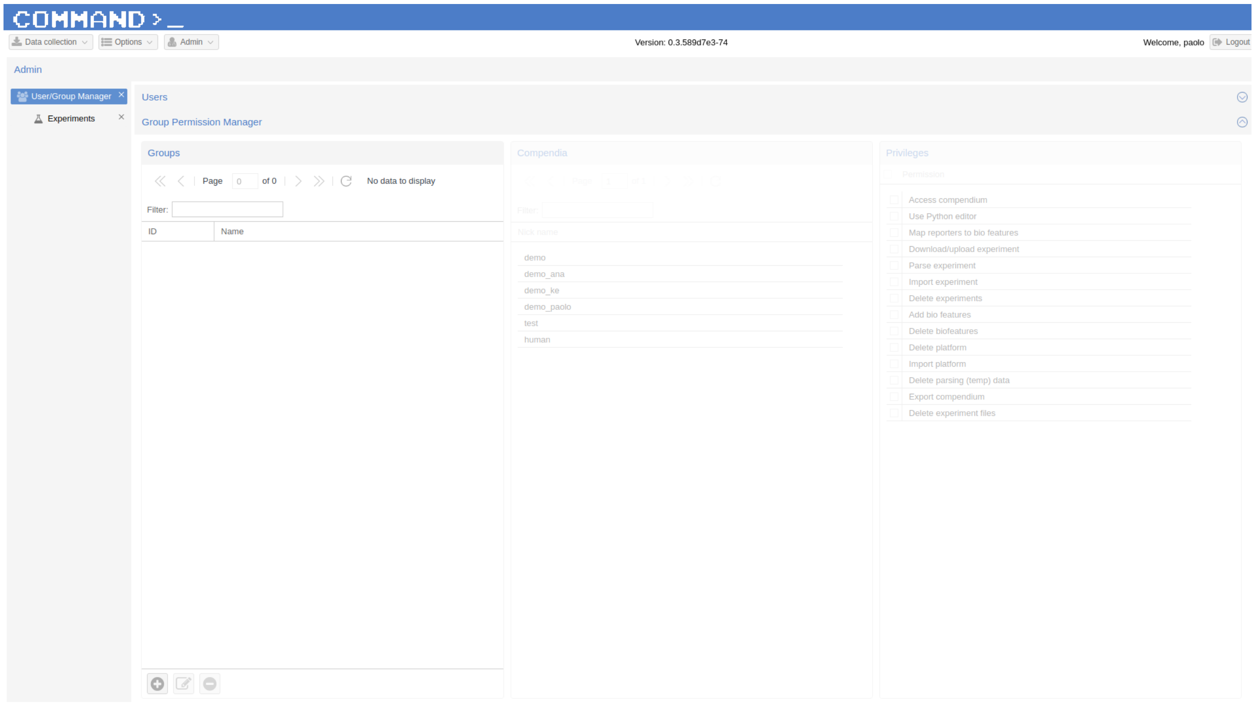


Fig. 2: Permission manager page

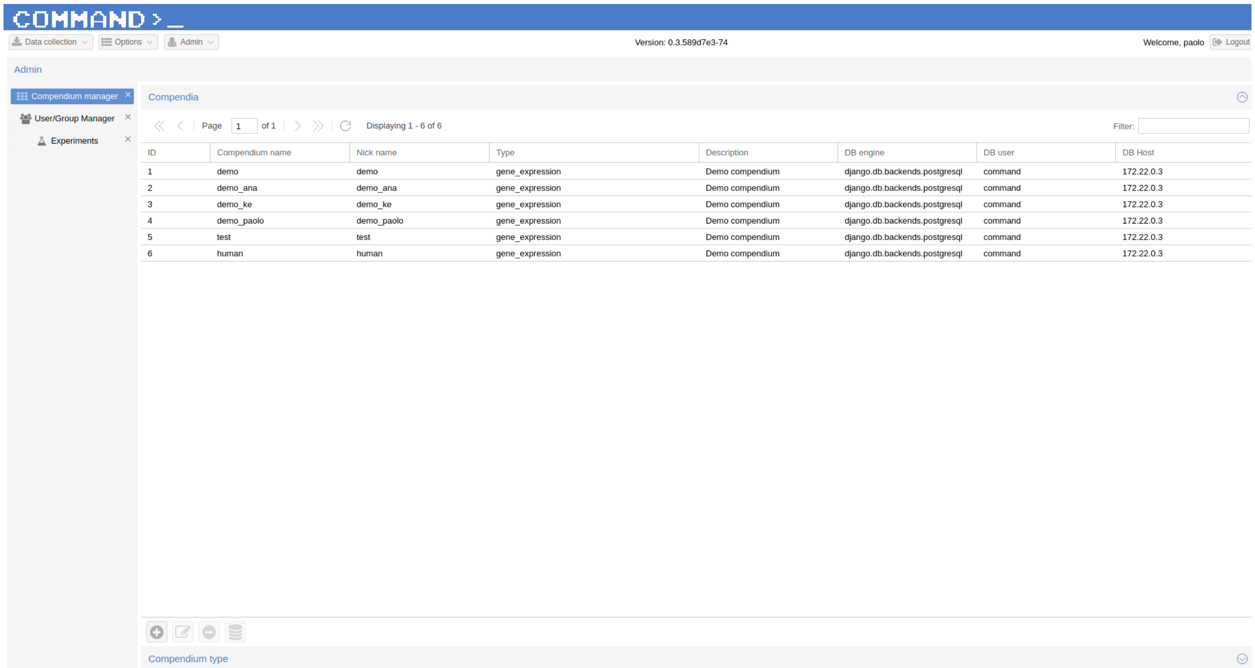
4.2 Compendium manager

From this page a compendium can be created, modified, deleted and initialized. From a technical point of view a *compendium* is nothing more than a database schema. When an admin user creates a new compendium he will be asked to add all the information necessary to connect to such database.

Once the connection information are saved and a new *compendium* appear in the grid, it would be possible to *initialize* it, i.e. to create the database schema.

Note: In this way it is possible to have compendia hosted on different database server. If the database do not exists yet it will be possible to have COMMAND>_ to create it on the fly but you will be asked to provide *username* and *password* for a **database admin user**. Default database admin user is *postgres* with password *postgres*.

The *Compendium Type* section is read-only and at the moment is filled only with *gene expression* since it is the only type of compendium you are allowed to create. To extend COMMAND>_ and allow other kind of quantitative data to be collected please have a look at *COMMAND>_for developers*.



ID	Compendium name	Nick name	Type	Description	DB engine	DB user	DB Host
1	demo	demo	gene_expression	Demo compendium	django.db.backends.postgresql	command	172.22.0.3
2	demo_ana	demo_ana	gene_expression	Demo compendium	django.db.backends.postgresql	command	172.22.0.3
3	demo_ke	demo_ke	gene_expression	Demo compendium	django.db.backends.postgresql	command	172.22.0.3
4	demo_paolo	demo_paolo	gene_expression	Demo compendium	django.db.backends.postgresql	command	172.22.0.3
5	test	test	gene_expression	Demo compendium	django.db.backends.postgresql	command	172.22.0.3
6	human	human	gene_expression	Demo compendium	django.db.backends.postgresql	command	172.22.0.3

Fig. 3: Compendium manager page

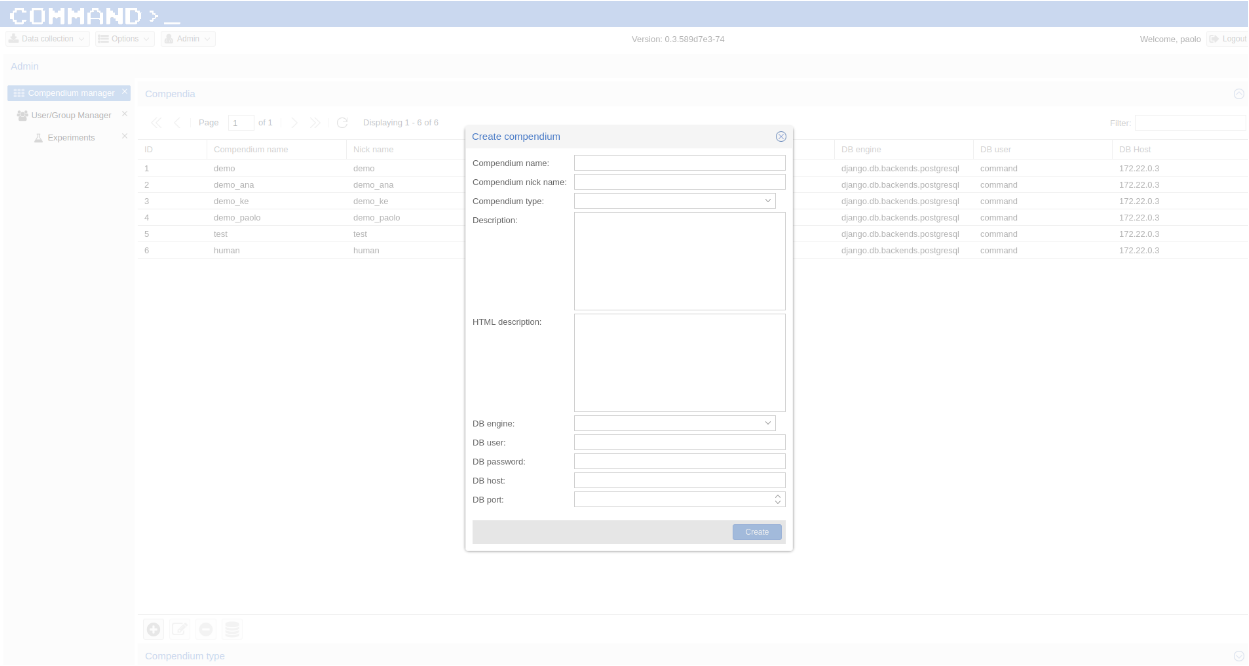


Fig. 4: New compendium page

CHAPTER 5

Deploy

COMMAND>_ is a complex application and relies on several other software components to work. In order to ease up the deployment process a `docker-compose.yml` file is provided, so assuming you have a working Docker Compose environment, the deployment process will be a matter of running a few commands.

In case you want to manually deploy COMMAND>_ in your environment there will be more steps you will need to take care of such as installing the web-server, the DBMS, etc.

5.1 Requirements

Have a look at the `requirements.txt` file for details. COMMAND>_ main dependencies are:

- Python 3
- Django
- PostgreSQL
- Celery
- Channels
- Numpy
- Pandas
- BioPython

COMMAND>_ uses several `external tools` that you'll need to download them separately:

- AffxFusion.jar
- kallisto
- BLAST+
- SRA-toolkit
- Trimmomatic

5.2 Docker Compose

Assuming that you have Docker Compose correctly installed, you should be able to perform the following steps:

```
# 1. clone the repository
git clone https://github.com/marcomoretto/command.git

# 2. copy external dependencies (check figure below)

# 3. build
docker-compose build

# 4. start docker
docker-compose up -d

# 5. create database schema
docker-compose exec web python manage.py migrate

# 6. create admin user
docker-compose exec web python manage.py init_admin

# 7. create initial options
docker-compose exec web python manage.py init_options

# 8. create demo compendium
docker-compose exec web python manage.py init_demo_compendium demo

# 9. run daphne
docker-compose exec -d daphne daphne -b 0.0.0.0 -p 8001 cport.asgi:channel_layer

# 10. run worker
docker-compose exec -d worker python3 manage.py runworker
```

That's it! You should be able to point your browser to http://localhost and login into COMMAND>_ using:

- username: `admin`
- password: `admin`

Note: You should have the following directory structure for the `external tools`

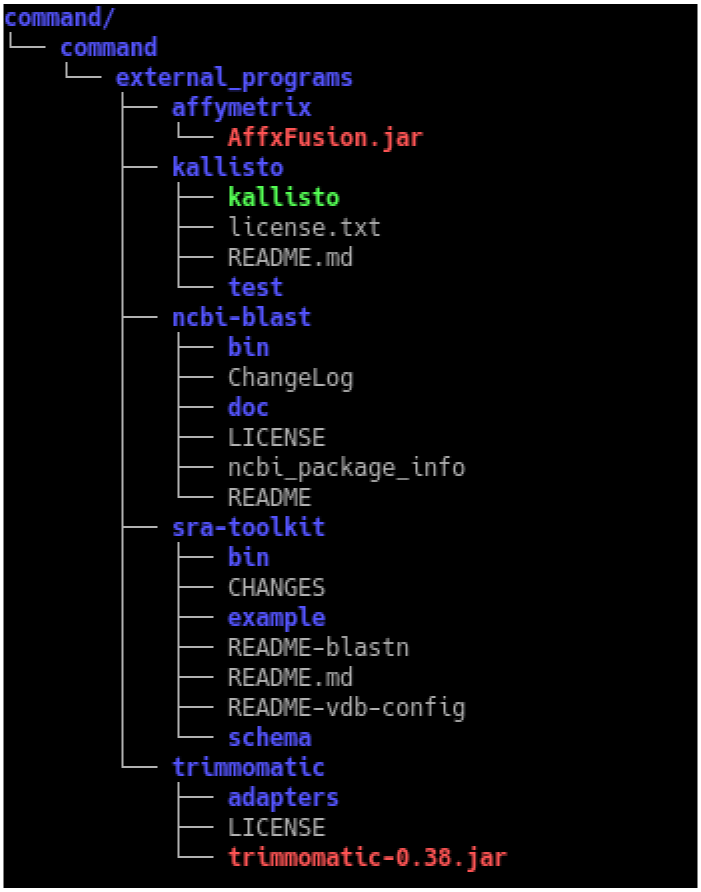
Note: You might need to rename the directory from `command` to `cport` before doing step # 2.

5.3 Manual Deploy

One easy way to understand what you need to do to manually deploy COMMAND>_ is to have a look at 2 files:

- the Dockerfile
- the docker-compose.yml file

In a nutshell, after having installed and configured Nginx (or another web-server to run Django applications), PostgreSQL, Redis, RabbitMQ and Celery, you'll have to run:



```
command/
└── command
    └── external_programs
        ├── affymetrix
        │   └── AffxFusion.jar
        ├── kallisto
        │   ├── kallisto
        │   ├── license.txt
        │   ├── README.md
        │   └── test
        ├── ncbi-blast
        │   ├── bin
        │   ├── ChangeLog
        │   ├── doc
        │   ├── LICENSE
        │   ├── ncbi_package_info
        │   └── README
        ├── sra-toolkit
        │   ├── bin
        │   ├── CHANGES
        │   ├── example
        │   ├── README-blastn
        │   ├── README.md
        │   ├── README-vdb-config
        │   └── schema
        └── trimmomatic
            ├── adapters
            ├── LICENSE
            └── trimmomatic-0.38.jar
```

```
pip3 install --upgrade pip
pip3 install Cython==0.28.1
pip3 install -r requirements.txt
```

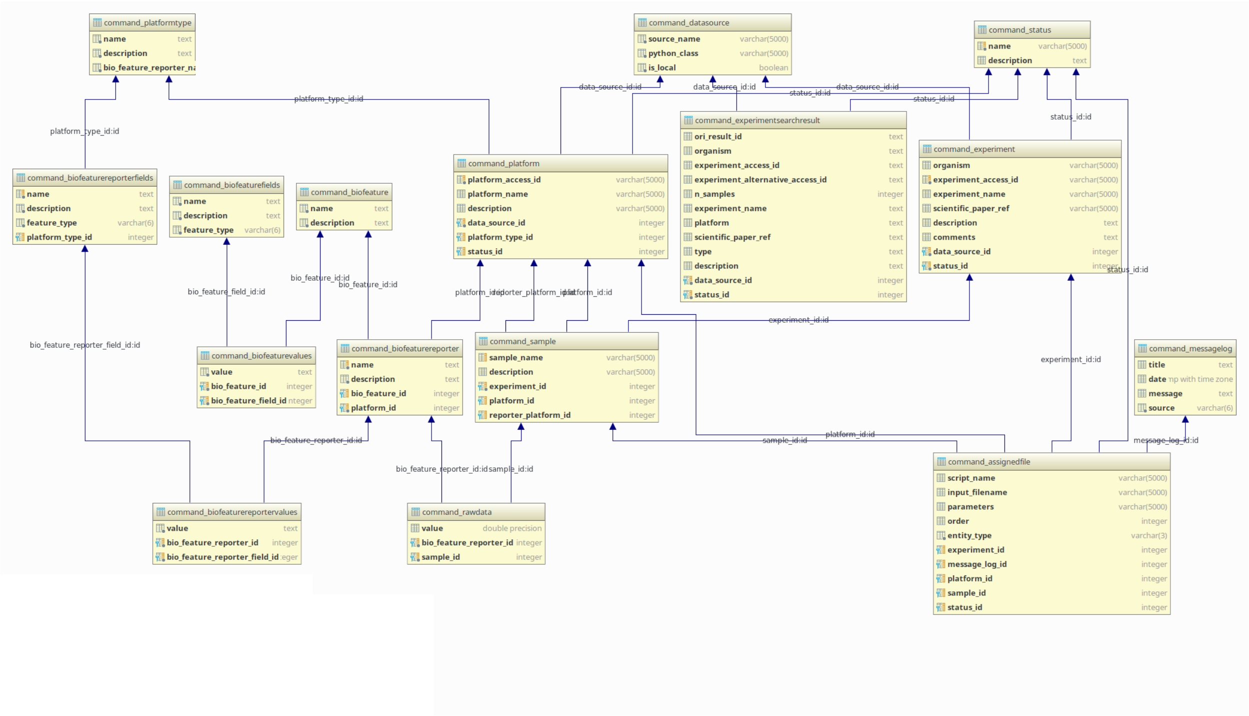
Now you should be ready configure Django (check the documentation for details), create the database schema and run the application.

```
python manage.py migrate

python manage.py init_admin

python manage.py init_options

python manage.py init_demo_compendium demo

daphne -b 0.0.0.0 -p 8001 cport.asgi:channel_layer

python3 manage.py runworker
```

Note: COMMAND>_ id a Django application so refer to the Django docs for database configuration https://docs.djangoproject.com/en/1.11/ref/settings/

CHAPTER 6

Database schema



CHAPTER 7

Use Cases

In this section we show how to both parse and import experiments from various gene expression platforms, technologies and sources (both public databases and local files) using the provided default scripts.

7.1 Use Case - Affymetrix from GEO

7.1.1 Import Gene Annotations

We want to look for experiments related to Yeast: the Saccharomyces Genome Database is the proper choice for retrieving sequences associated to Yeast's genes (from this link). Go to > Data collection (on the top left corner) then > Biological features > Import biological feature (+ symbol on the bottom left) > Type: FASTA , File name: select the annotation file you downloaded before > Import Biological features. Wait.

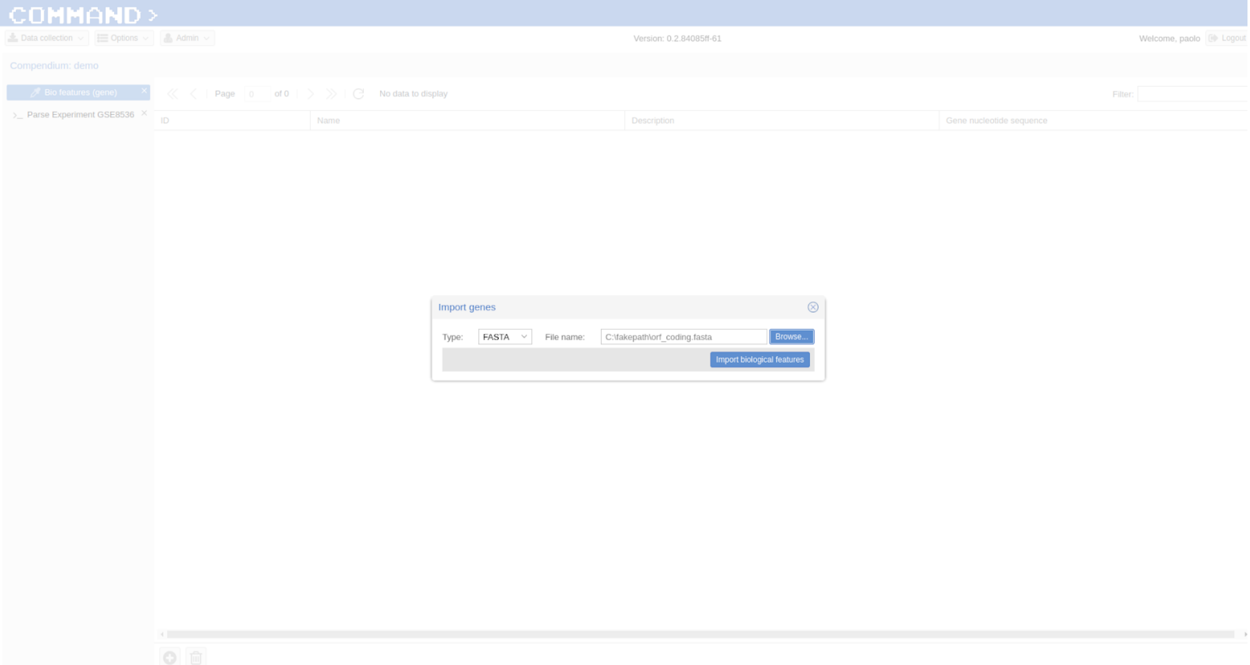
We start by selecting Experiments from Data collection (top left corner) then we highlight the experiment of interest (it was previously retrieved from GEO following *Searching public databases*), here GSE8536, an expression analyses study which inspects the response of Saccharomyces cerevisiae to stress throughout a 15-day wine fermentation.

7.1.2 Parse Experiment, Platform and Samples

Since we have a new platform (GPL90) never imported before into COMMAND>_ for this compendium, we retrieve the sequences associated to the Affymetrix probe ids (`YG_S98 probes`) for this platform from the Affymetrix Support sitewebsite.

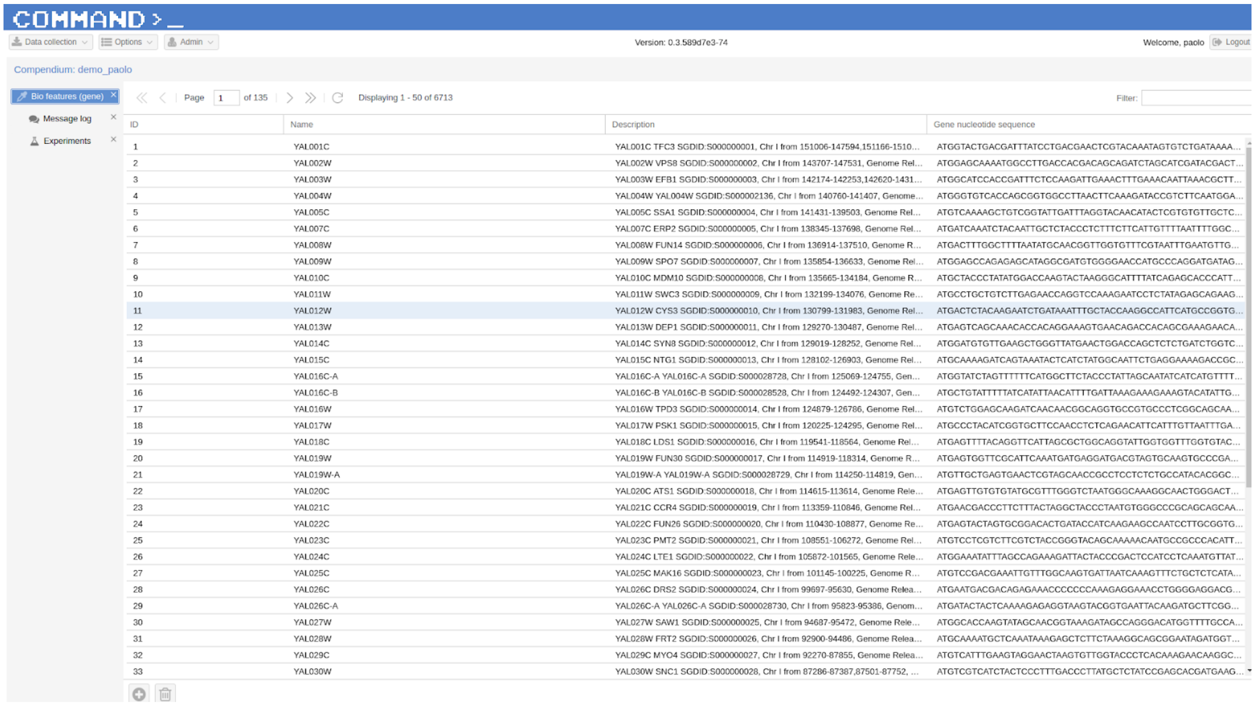
From Experiments (Data collection Menu) we highlight the selected experiment (GSE8536 here) and click the Parse/Import experiment from the bottom bar. On the main window you can see that the Experiment tab is populated with metadata gathered from the publicDB (GEO here). Now we can start parsing the Experiment, the Platform(s) and the Samples.

Being a dataset retrieved from GEO we take advantage of the .soft file downloaded (see GEO Documentation for a description of this type of file):

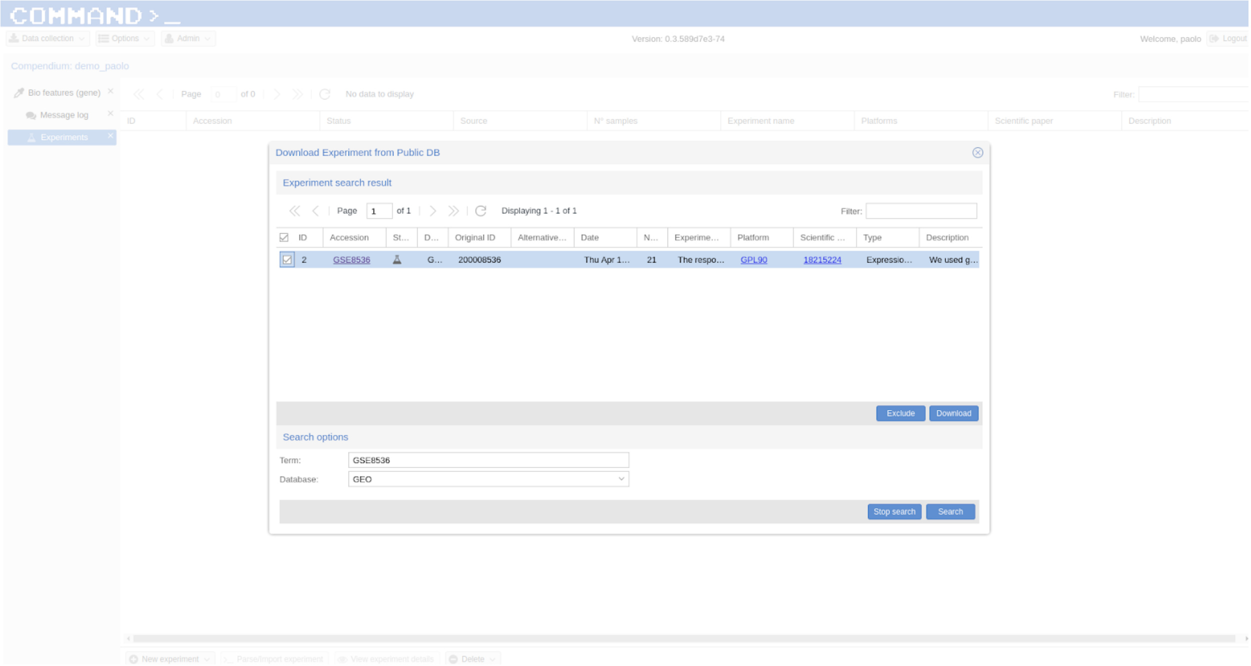


Type: FASTA

File name: C:\fakepath\orf_coding.fasta



ID	Name	Description	Gene nucleotide sequence
1	YAL001C	YAL001C TFC3 SGDID:S000000001, Chr I from 151006-147594,151166-1510...	ATGGTACTGACGATTTATCCTGACGAACTCGTACAAATAGTGTCTGATAAAA...
2	YAL002W	YAL002W VPS8 SGDID:S000000002, Chr I from 143707-147531, Genome Rel...	ATGGAGCAAAATGGCCTTGACCACGACAGCAGATCTAGCATCGATACGACT...
3	YAL003W	YAL003W EFB1 SGDID:S000000003, Chr I from 142174-142253,142620-1431...	ATGGCATCCACCGATTTCTCCAAGATTGAAACTTTGAAACAATTAAACGCTT...
4	YAL004W	YAL004W YAL004W SGDID:S000002136, Chr I from 140760-141407, Genome...	ATGGGTGTCACCAGCGGTGGCCTTAACTTCAAAGATACCGTCTTCAATGGA...
5	YAL005C	YAL005C SSA1 SGDID:S000000004, Chr I from 141431-139503, Genome Rel...	ATGTCAAAAGCTGTCGGTATTGATTTAGGTACAACATACTCGTGTGTTGCTC...
6	YAL007C	YAL007C ERP2 SGDID:S000000005, Chr I from 138345-137698, Genome Rel...	ATGATCAAATCTACAATTGCTCTACCCTCTTTCTTCATTGTTTTAATTTTGGC...
7	YAL008W	YAL008W FUN14 SGDID:S000000006, Chr I from 136914-137510, Genome R...	ATGACTTTGGCTTTTAATATGCAACGGTTGGTGTTTCGTAATTTGAATGTTG...
8	YAL009W	YAL009W SPO7 SGDID:S000000007, Chr I from 135854-136633, Genome Rel...	ATGGAGCCAGAGAGCATAGGCGATGTGGGGAACCATGCCCAGGATGATAG...
9	YAL010C	YAL010C MDM10 SGDID:S000000008, Chr I from 135665-134184, Genome R...	ATGCTACCCTATATGGACCAAGTACTAAGGGCATTTTATCAGAGCACCCATT...
10	YAL011W	YAL011W SWC3 SGDID:S000000009, Chr I from 132199-134076, Genome Re...	ATGCCTGCTGTCTTGAGAACCAGGTCCAAAGAATCCTCTATAGAGCAGAAG...
11	YAL012W	YAL012W CYS3 SGDID:S000000010, Chr I from 130799-131983, Genome Rel...	ATGACTCTACAAGAATCTGATAAATTTGCTACCAAGGCCATTCATGCCGGTG...
12	YAL013W	YAL013W DEP1 SGDID:S000000011, Chr I from 129270-130487, Genome Rel...	ATGAGTCAGCAAACACCACAGGAAAGTGAACAGACCACAGCGAAAGAACA...
13	YAL014C	YAL014C SYN8 SGDID:S000000012, Chr I from 129019-128252, Genome Rel...	ATGGATGTGTTGAAGCTGGGTTATGAACTGGACCAGCTCTCTGATCTGGTC...
14	YAL015C	YAL015C NTG1 SGDID:S000000013, Chr I from 128102-126903, Genome Rel...	ATGCAAAAGATCAGTAAATACTCATCTATGGCAATTCTGAGGAAAAGACCGC...
15	YAL016C-A	YAL016C-A YAL016C-A SGDID:S000028728, Chr I from 125069-124755, Gen...	ATGGTATCTAGTTTTTTCATGGCTTCTACCCTATTAGCAATATCATCATGTTTT...
16	YAL016C-B	YAL016C-B YAL016C-B SGDID:S000028528, Chr I from 124492-124307, Gen...	ATGCTGTATTTTTATCATATTAACATTTTGATTAAAGAAAGAAAGTACATATTG...
17	YAL016W	YAL016W TPD3 SGDID:S000000014, Chr I from 124879-126786, Genome Rel...	ATGTCTGGAGCAAGATCAACAACGGCAGGTGCCGTGCCCTCGGCAGCAA...
18	YAL017W	YAL017W PSK1 SGDID:S000000015, Chr I from 120225-124295, Genome Rel...	ATGCCCTACATCGGTGCTTCCAACCTCTCAGAACATTCATTTGTTAATTTGA...
19	YAL018C	YAL018C LDS1 SGDID:S000000016, Chr I from 119541-118564, Genome Rel...	ATGAGTTTTACAGGTTCATTAGCGCTGGCAGGTATTGGTGGTTTGGTGTAC...
20	YAL019W	YAL019W FUN30 SGDID:S000000017, Chr I from 114919-118314, Genome R...	ATGAGTGGTTCGCATTCAAATGATGAGGATGACGTAGTGCAAGTGCCCGA...
21	YAL019W-A	YAL019W-A YAL019W-A SGDID:S000028729, Chr I from 114250-114819, Gen...	ATGTTGCTGAGTGAACTCGTAGCAACCGCCTCCTCTCTGCCATACACGGC...
22	YAL020C	YAL020C ATS1 SGDID:S000000018, Chr I from 114615-113614, Genome Rele...	ATGAGTTGTGTGTATGCGTTTGGGTCTAATGGGCAAAGGCAACTGGGACT...
23	YAL021C	YAL021C CCR4 SGDID:S000000019, Chr I from 113359-110846, Genome Rel...	ATGAACGACCCTTCTTTACTAGGCTACCCTAATGTGGGCCCGCAGCAGCAA...
24	YAL022C	YAL022C FUN26 SGDID:S000000020, Chr I from 110430-108877, Genome Re...	ATGAGTACTAGTGCGGACACTGATACCATCAAGAAGCCAATCCTTGCGGTG...
25	YAL023C	YAL023C PMT2 SGDID:S000000021, Chr I from 108551-106272, Genome Rel...	ATGTCCTCGTCTTCGTCTACCGGGTACAGCAAAAACAATGCCGCCCACATT...
26	YAL024C	YAL024C LTE1 SGDID:S000000022, Chr I from 105872-101565, Genome Rele...	ATGGAAATATTTAGCCAGAAAGATTACTACCCGACTCCATCCTCAAATGTTAT...
27	YAL025C	YAL025C MAK16 SGDID:S000000023, Chr I from 101145-100225, Genome R...	ATGTCCGACGAAATTGTTTGGCAAGTGATTAATCAAAGTTTCTGCTCTCATA...
28	YAL026C	YAL026C DRS2 SGDID:S000000024, Chr I from 99697-95630, Genome Relea...	ATGAATGACGACAGAGAAACCCCCCCAAAGAGGAAACCTGGGGAGGACG...
29	YAL026C-A	YAL026C-A YAL026C-A SGDID:S000028730, Chr I from 95823-95386, Genom...	ATGATACTACTCAAAAGAGAGGTAAGTACGGTGAATTACAAGATGCTTCGG...
30	YAL027W	YAL027W SAW1 SGDID:S000000025, Chr I from 94687-95472, Genome Rele...	ATGGCACCAAGTATAGCAACGGTAAAGATAGCCAGGGACATGGTTTTGCCA...
31	YAL028W	YAL028W FRT2 SGDID:S000000026, Chr I from 92900-94486, Genome Relea...	ATGCAAAATGCTCAAATAAAGAGCTCTTCTAAAGGCAGCGGAATAGATGGT...
32	YAL029C	YAL029C MYO4 SGDID:S000000027, Chr I from 92270-87855, Genome Relea...	ATGTCATTTGAAGTAGGAACTAAGTGTTGGTACCCTCACAAAGAACAAGGC...
33	YAL030W	YAL030W SNC1 SGDID:S000000028, Chr I from 87286-87387,87501-87752, ...	ATGTCGTCATCTACTCCCTTTGACCCTTATGCTCTATCCGAGCACGATGAAG...



Download Experiment from Public DB

Experiment search result

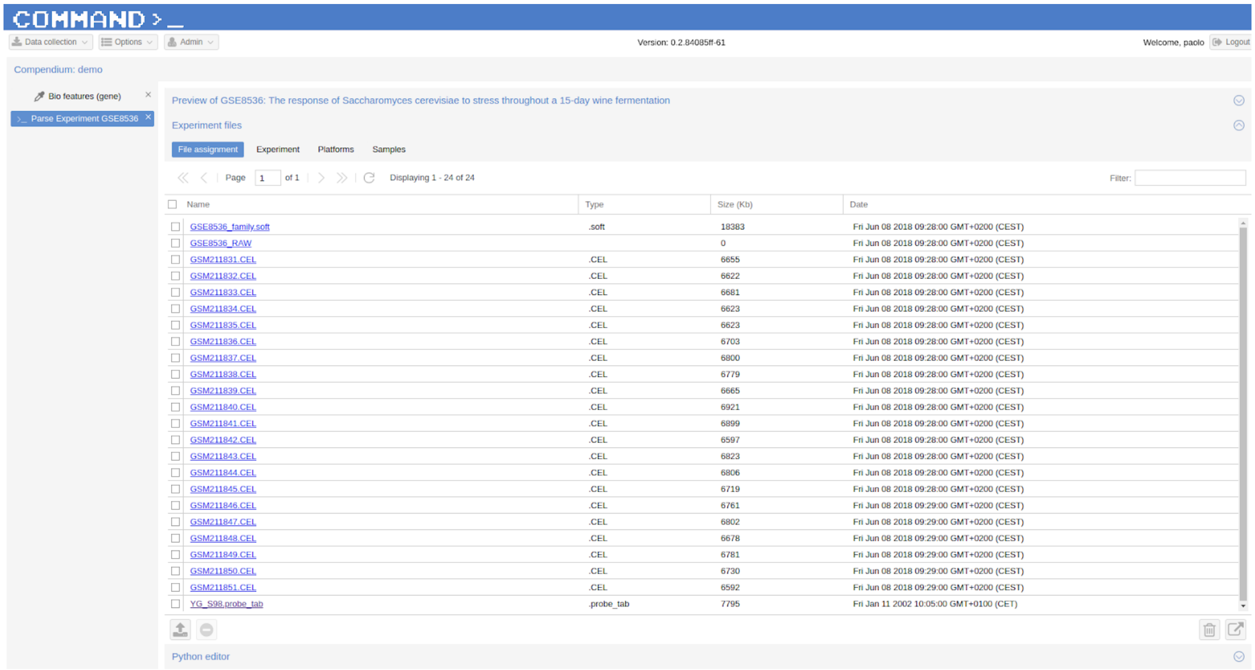
Page 1 of 1 Displaying 1 - 1 of 1 Filter:

☑	ID	Accession	St...	D...	Original ID	Alternative...	Date	N...	Experime...	Platform	Scientific ...	Type	Description
☑	2	GSE8536		G...	200008536		Thu Apr 1...	21	The respo...	GPL90	18215224	Expressio...	We used g...

Search options

Term: GSE8536

Database: GEO



Preview of GSE8536: The response of Saccharomyces cerevisiae to stress throughout a 15-day wine fermentation

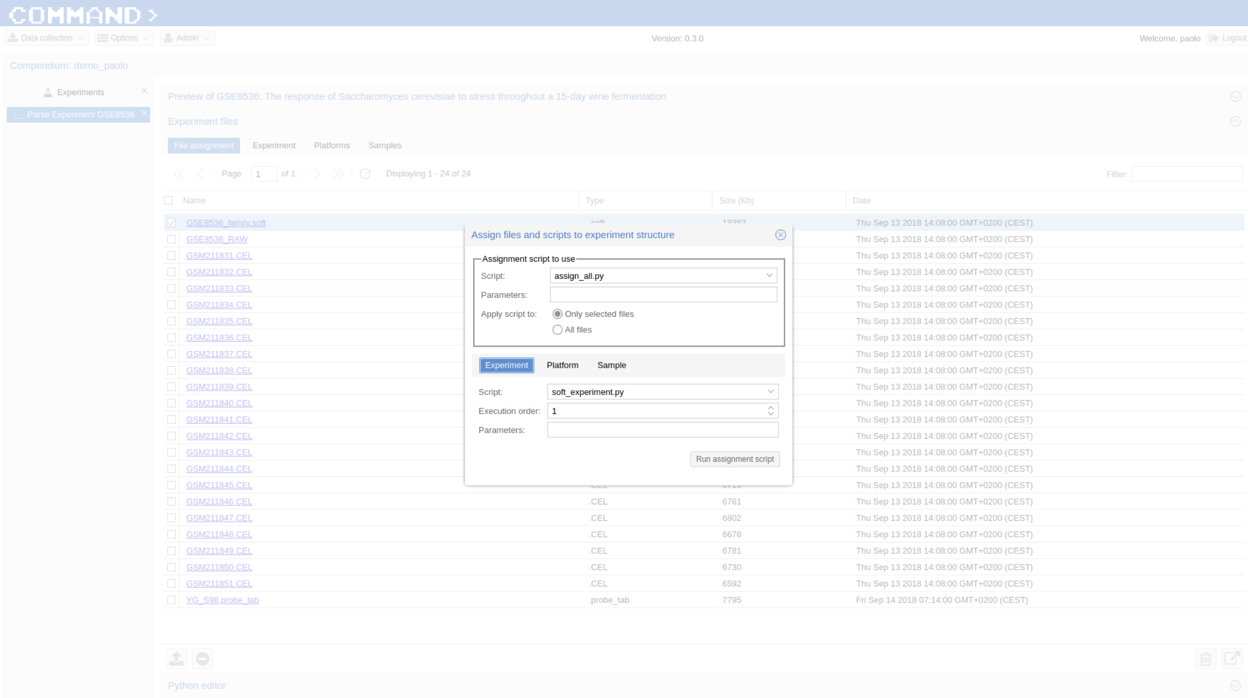
Experiment files

Page 1 of 1 Displaying 1 - 24 of 24 Filter:

☐	Name	Type	Size (Kb)	Date
☐	GSE8536_family.soft	.soft	18383	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSE8536_RAW		0	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211831.CEL	.CEL	6655	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211832.CEL	.CEL	6622	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211833.CEL	.CEL	6681	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211834.CEL	.CEL	6623	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211835.CEL	.CEL	6623	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211836.CEL	.CEL	6703	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211837.CEL	.CEL	6800	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211838.CEL	.CEL	6779	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211839.CEL	.CEL	6665	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211840.CEL	.CEL	6921	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211841.CEL	.CEL	6899	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211842.CEL	.CEL	6597	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211843.CEL	.CEL	6823	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211844.CEL	.CEL	6806	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211845.CEL	.CEL	6719	Fri Jun 08 2018 09:28:00 GMT+0200 (CEST)
☐	GSM211846.CEL	.CEL	6761	Fri Jun 08 2018 09:29:00 GMT+0200 (CEST)
☐	GSM211847.CEL	.CEL	6802	Fri Jun 08 2018 09:29:00 GMT+0200 (CEST)
☐	GSM211848.CEL	.CEL	6678	Fri Jun 08 2018 09:29:00 GMT+0200 (CEST)
☐	GSM211849.CEL	.CEL	6781	Fri Jun 08 2018 09:29:00 GMT+0200 (CEST)
☐	GSM211850.CEL	.CEL	6730	Fri Jun 08 2018 09:29:00 GMT+0200 (CEST)
☐	GSM211851.CEL	.CEL	6592	Fri Jun 08 2018 09:29:00 GMT+0200 (CEST)
☐	YG_S98.probe_tab	.probe_tab	7795	Fri Jan 11 2002 10:05:00 GMT+0100 (CET)

Select *GSE8536_family.soft* and click the *Use assignment script to assign files to experiment entities* icon on the bottom-right. A dialog will show-up:

- Script > assign_all.py > Only selected files
- Experiment tab > Script: > soft_experiment.py, Execution order: 1
- Platform tab > Script: > soft_platform.py, Execution order: 1
- Sample tab > Script: > soft_sample.py, Execution order: 1
- Run assignment script



Assign files and scripts to experiment structure

Assignment script to use

Script: assign_all.py

Parameters:

Apply script to: ● Only selected files ○ All files

Experiment | Platform | Sample

Script: soft_experiment.py

Execution order: 1

Parameters:

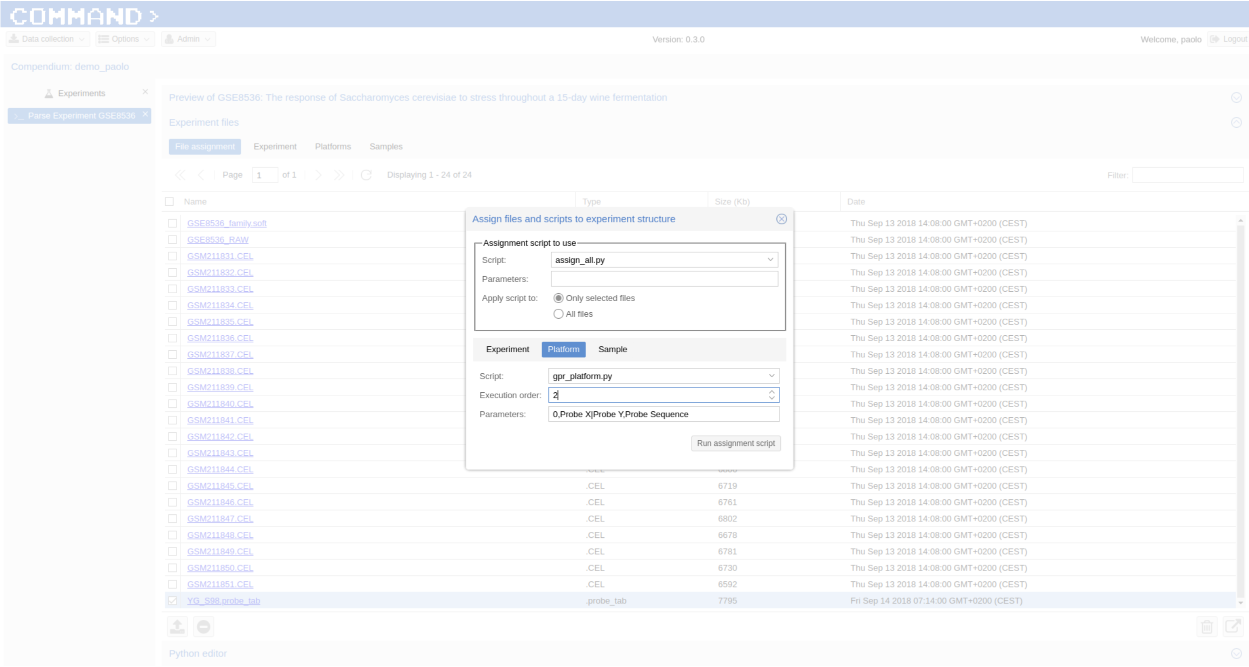
Run assignment script

Now in order to parse the new platform we are going to use the sequences associated to the Affymetrix probe ids we have already downloaded. We import the annotation (`YG_S98 probes`) in the *File assignment* section of *Experiment files* clicking the upload icon on the bottom of the page.

Now we associate the file to the platform:

- In Experiment files Section > File Assignment select the uploaded file (YG_S98.probe_tab) and click *Use assignment script to assign files to experiment entities*. On the Assign files dialog:
- Script: assign_all.py
- Param:
- Only selected files checked (default)
- Platform tab > Script: *gpr_platform.py* , Parameters: 0,Probe X|Probe Y,Probe Sequence , Execution order: 2
- Run assignment script

Note:



Assign files and scripts to experiment structure

Assignment script to use

Script: assign_all.py

Parameters:

Apply script to: ● Only selected files ○ All files

Script: gpr_platform.py

Execution order: 2

Parameters: 0,Probe X|Probe Y,Probe Sequence

- the Parameters assigned to the *gpr_platform.py* script specify to not skip any line, use the combination of Probe X and Probe Y columns to create an unique id for the cel files and indicate the sequences for the probes are in the Probe Sequence column.
- The parsing of the Platform is a once time procedure: from now on we can use this platform for all related experiments.

Now we parse the Affymetrix cel files (sample files):

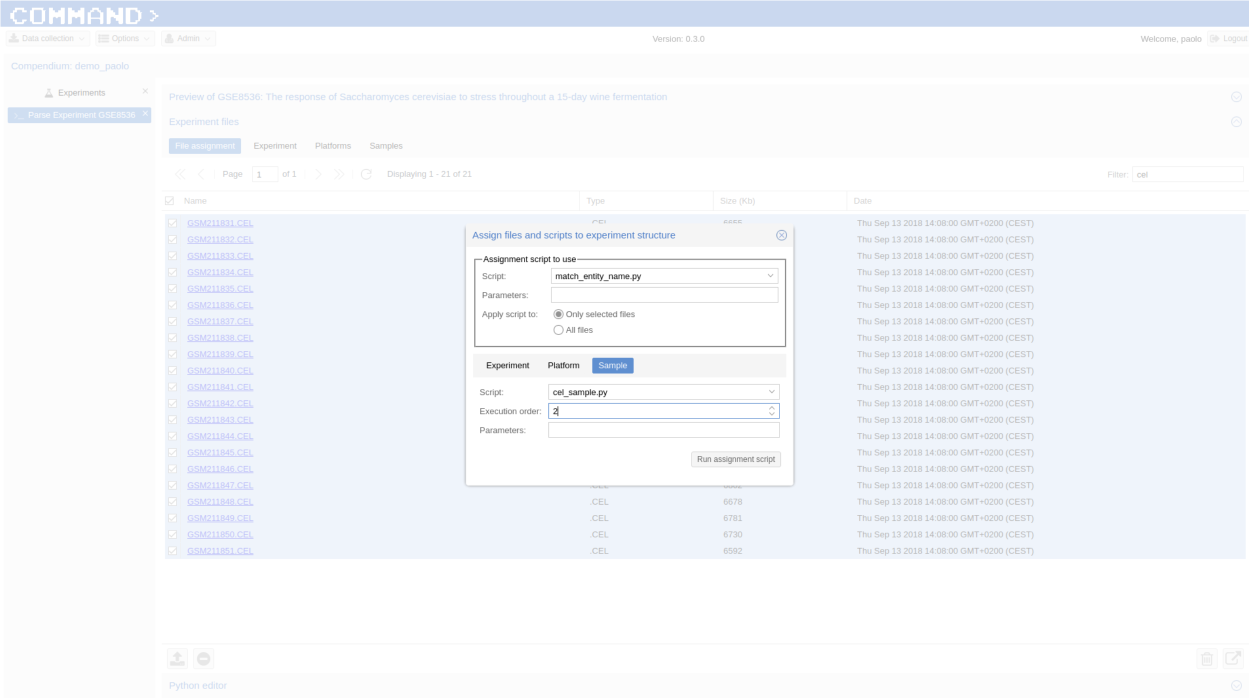
- In Experiment files Section > File Assignment we use CEL as filter and select all files > click the *Use assignment script to assign files to experiment entities* icon on the bottom-right corner and the Assign files and scripts to experiment structure dialog will pop-up:
- Script > match_entity_name.py
- Only selected files (default) checked
- Sample tab > Script: *cell_sample.py*, Execution order: 2
- Run assignment script

Finally, in the Preview Section (*Preview of GSE8536* here) click Run Selected (bottom-right corner). After a while your samples will be parsed.

Now you can Import both the Platform (since is the first time we use this specific one) and the Experiment.

Tip: Check that both the platform and the samples are properly parsed from the Preview interface of the Parse Experiment section clicking on the platform and on each sample.

Click the Import button on the bottom-right corner and select Import whole experiment. After a while the experiment and the platform (in this case) will be imported.



Preview of GSE8536: The response of Saccharomyces cerevisiae to stress throughout a 15-day wine fermentation

Experiment files

File assignment | Experiment | Platforms | Samples

Page 1 of 1 | Displaying 1 - 21 of 21 | Filter: cel

☑ Name	Type	Size (Kb)	Date
☑ GSM211831.CEL	.CEL	6655	Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211832.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211833.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211834.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211835.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211836.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211837.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211838.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211839.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211840.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211841.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211842.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211843.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211844.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211845.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211846.CEL			Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211847.CEL	.CEL		Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211848.CEL	.CEL	6678	Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211849.CEL	.CEL	6781	Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211850.CEL	.CEL	6730	Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)
☑ GSM211851.CEL	.CEL	6592	Thu Sep 13 2018 14:08:00 GMT+0200 (CEST)

Assign files and scripts to experiment structure

Assignment script to use

Script: match_entity_name.py

Parameters:

Apply script to: ● Only selected files ○ All files

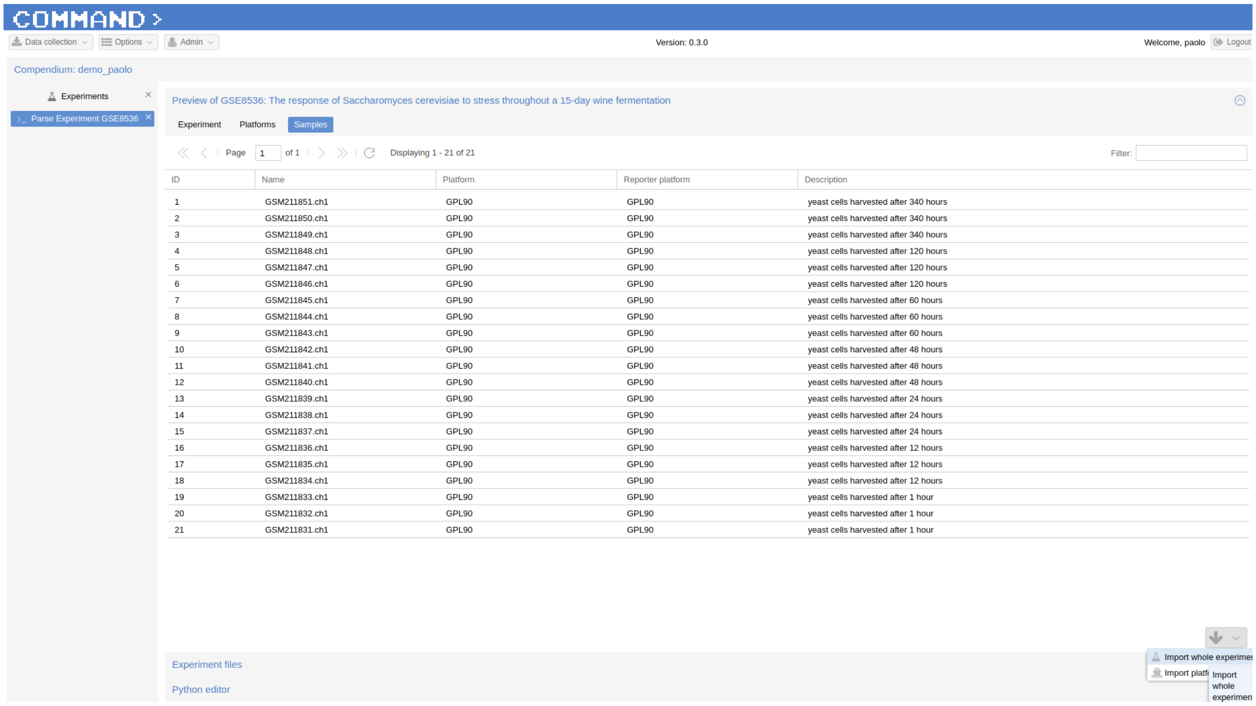
Experiment | Platform | Sample

Script: cel_sample.py

Execution order: 2

Parameters:

Run assignment script



Preview of GSE8536: The response of Saccharomyces cerevisiae to stress throughout a 15-day wine fermentation

Experiment | Platforms | Samples

Page 1 of 1 | Displaying 1 - 21 of 21 | Filter:

ID	Name	Platform	Reporter platform	Description
1	GSM211851.ch1	GPL90	GPL90	yeast cells harvested after 340 hours
2	GSM211850.ch1	GPL90	GPL90	yeast cells harvested after 340 hours
3	GSM211849.ch1	GPL90	GPL90	yeast cells harvested after 340 hours
4	GSM211848.ch1	GPL90	GPL90	yeast cells harvested after 120 hours
5	GSM211847.ch1	GPL90	GPL90	yeast cells harvested after 120 hours
6	GSM211846.ch1	GPL90	GPL90	yeast cells harvested after 120 hours
7	GSM211845.ch1	GPL90	GPL90	yeast cells harvested after 60 hours
8	GSM211844.ch1	GPL90	GPL90	yeast cells harvested after 60 hours
9	GSM211843.ch1	GPL90	GPL90	yeast cells harvested after 60 hours
10	GSM211842.ch1	GPL90	GPL90	yeast cells harvested after 48 hours
11	GSM211841.ch1	GPL90	GPL90	yeast cells harvested after 48 hours
12	GSM211840.ch1	GPL90	GPL90	yeast cells harvested after 48 hours
13	GSM211839.ch1	GPL90	GPL90	yeast cells harvested after 24 hours
14	GSM211838.ch1	GPL90	GPL90	yeast cells harvested after 24 hours
15	GSM211837.ch1	GPL90	GPL90	yeast cells harvested after 24 hours
16	GSM211836.ch1	GPL90	GPL90	yeast cells harvested after 12 hours
17	GSM211835.ch1	GPL90	GPL90	yeast cells harvested after 12 hours
18	GSM211834.ch1	GPL90	GPL90	yeast cells harvested after 12 hours
19	GSM211833.ch1	GPL90	GPL90	yeast cells harvested after 1 hour
20	GSM211832.ch1	GPL90	GPL90	yeast cells harvested after 1 hour
21	GSM211831.ch1	GPL90	GPL90	yeast cells harvested after 1 hour

7.2 Use Case - Nimblegen from ArrayExpress

In COMMAND>_ the preferred way to import experiments from public db is by using GEO which provide the most convenient interface out-of-the-box. In case an experiment is not included in GEO it is possible to import it from ArrayExpress. Start by searching the experiment of interest following the procedure described in *Searching public databases*, select E-GEOD-58806 as Term and ArrayExpress as Database. Go the experiment slide on the left, select the experiment of interest (here E-GEOD-58806) and click >_ Parse/Import experiment. On the main window you can see that the Experiment tab is populated with metadata gathered from the publicDB (ArrayExpress here).

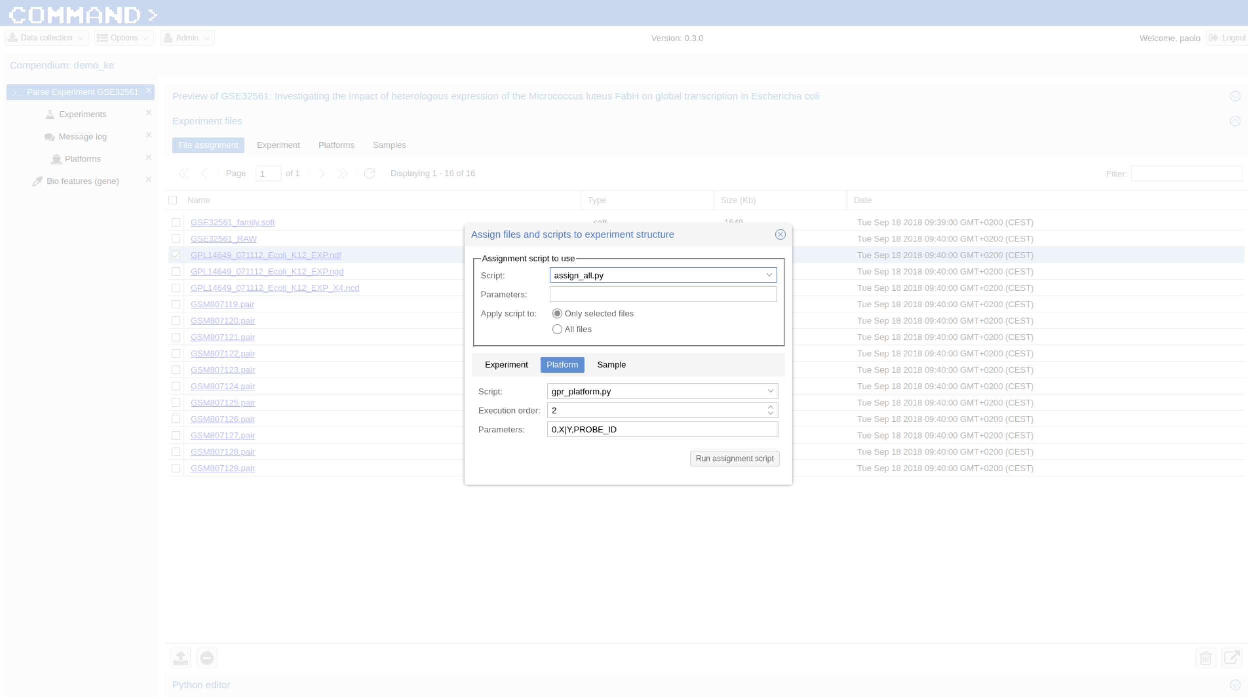
7.2.1 Import Platform from GEO

COMMAND>_can use a previous imported platform from a different public database (either ArrayExpress or GEO) and assign it as Reporter platform (in the preview main section of Parsing) for the current experiment.

In our case we want to parse and import an experiment from ArrayExpress using a previously imported platform from GEO. In order to do so we import ONLY the platform for another experiment (here GSE32561) which uses the same platform of the experiment of interest. After the selection of the new experiment using the Searching from public db procedure we use the Nimblegen ndf files which allows to associate probes to sequences to the platform GPL14649.

Experiment files > File Assignment > Select *GPL14649_071112_Ecoli_K12_EXP.ndf* and in the Assign files dialog:

- Script: match_entity_name.py
- Param: platform
- Only selected files checked (default)
- Platform tab > Script: > gpr_platform.py; Parameters: 0,X|Y,PROBE_ID; Execution order: 2
- Run assignment script



Preview of GSE32561: Investigating the impact of heterologous expression of the Micrococcus luteus FabH on global transcription in Escherichia coli

Displaying 1 - 16 of 16

	Name	Type	Size (Kb)	Date
☐	GSE32561_family.soft	soft	1649	Tue Sep 18 2018 09:39:00 GMT+0200 (CEST)
☐	GSE32561_RAW			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)
☑	GPL14649_071112_Ecoli_K12_EXP.ndf			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)
☐	GPL14649_071112_Ecoli_K12_EXP.ngd			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)
☐	GPL14649_071112_Ecoli_K12_EXP_X4.ncd			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)
☐	GSM807119.pair			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)
☐	GSM807120.pair			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)
☐	GSM807121.pair			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)
☐	GSM807122.pair			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)
☐	GSM807123.pair			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)
☐	GSM807124.pair			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)
☐	GSM807125.pair			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)
☐	GSM807126.pair			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)
☐	GSM807127.pair			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)
☐	GSM807128.pair			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)
☐	GSM807129.pair			Tue Sep 18 2018 09:40:00 GMT+0200 (CEST)

Assign files and scripts to experiment structure

Assignment script to use

Script: assign_all.py

Parameters:

Apply script to: ● Only selected files ○ All files

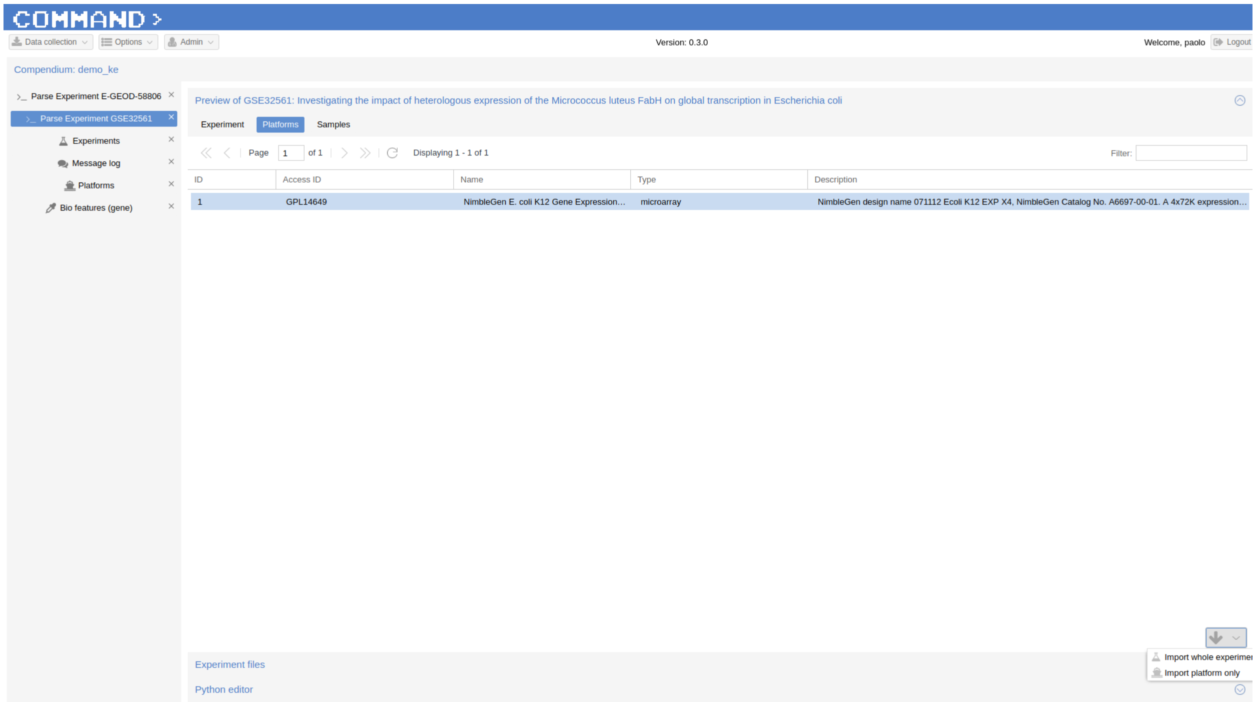
Platform tab:

Script: gpr_platform.py

Execution order: 2

Parameters: 0,X|Y,PROBE_ID

Now we can import this platform from the Platform section of Preview:



Preview of GSE32561: Investigating the impact of heterologous expression of the Micrococcus luteus FabH on global transcription in Escherichia coli

ID	Access ID	Name	Type	Description
1	GPL14649	NimbleGen E. coli K12 Gene Expression…	microarray	NimbleGen design name 071112 Ecoli K12 EXP X4, NimbleGen Catalog No. A6697-00-01. A 4x72K expression…

7.2.2 Parse Experiment, Platform and Samples

Now the Platform is available and can be used to import the experiment retrieved from ArrayExpress. Go to Experiments > Parse Experiment E-GEOD-58806 > Experiment Files > Platform and now click over A-GEOD-14649 in the Reporter Platform field and selected the previously imported GPL14649.

Finally you parse and import the nimblegen .pair files:

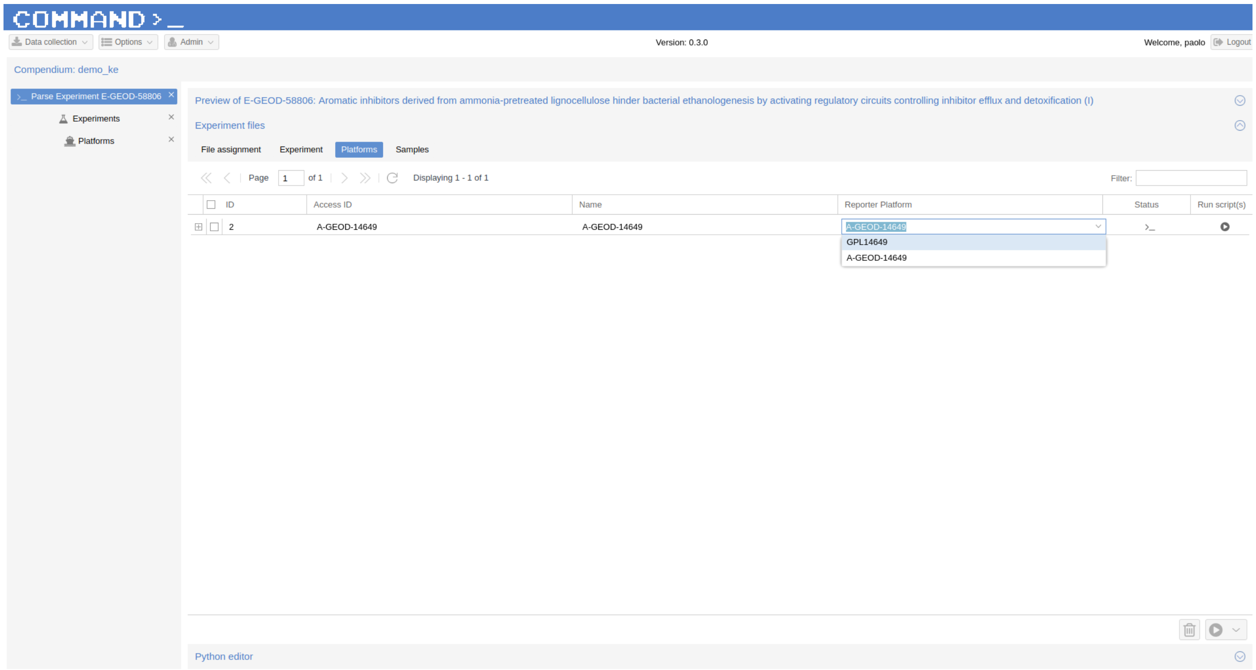
- In Experiment files Section > File Assignment > Filter .pair and select all files
- click the *Use assignment script to assign files to experiment entities* icon on the bottom-right and the Assign files and scripts to experiment structure dialog will pop-up:
- Script: *match_sample_name.py* > Only selected files
- Sample: Script: > *pair_sample.py*, Execution order: 2
- Run assignment script

7.3 Use Case - Multiplatform Experiment

It is standard practice for gene expressione esperiments to make use of multiple platforms for the same organism in the same experiment: usually it comes from multiple single experiments performed in different conditions/time. Here, we select from GEO the GSE13713 experiment regarding Phenotypic and transcriptomic analyses of mildly and severely salt-stressed Bacillus cereus ATCC. It is related to two platforms: GPL7634 and GPL7636.

7.3.1 Import Gene Annotation

Since the platforms related to the selected experiment were never imported before into COMMAND>_, we need the gene sequences in order to properly import our probes at gene level. We got gene/sequence list from ncbi: go here and



Preview of E-GEOD-58806: Aromatic inhibitors derived from ammonia-pretreated lignocellulose hinder bacterial ethanologenesis by activating regulatory circuits controlling inhibitor efflux and detoxification (I)

ID	Access ID	Name	Reporter Platform	Status	Run script(s)
2	A-GEOD-14649	A-GEOD-14649	A-GEOD-14649	>_	

from the top-right button select send to: Coding sequences, Format: FASTA Nucleotide and Choose destination: File. In COMMAND>_ go to > Data Collection (on the top left corner) then > Bio features (genes) > Import biological feature (+ symbol on the bottom left) > Type: FASTA , File name: select the annotation file you downloaded before > Import Biological features.

7.3.2 Parse Platforms and Samples

In order to parse the two platforms, we need both the soft file related to the experiment and the soft_platform.py script.

In Experiment files Section > File Assignement > Select the GSE13713_family.soft file and on the Assign files dialog:

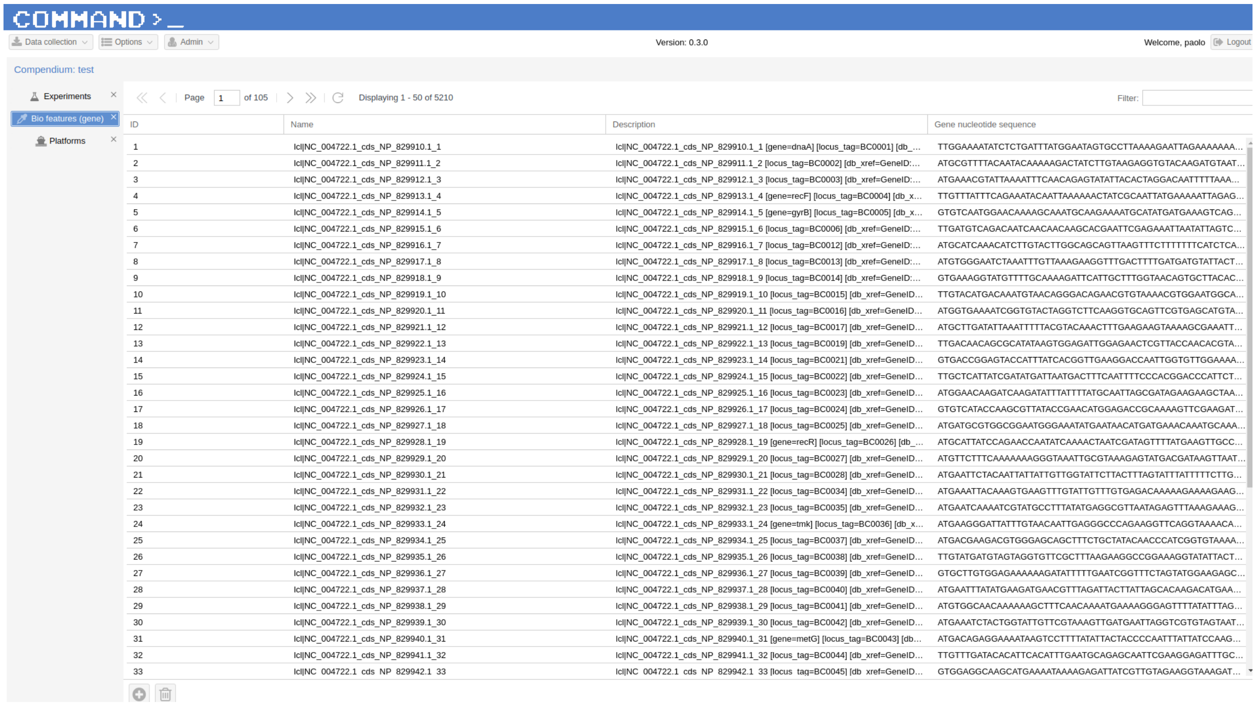
- Script: *match_all.py*
- Param: platform
- Only selected files checked
- Platform tab > Script: > *soft_platform.py*, parameters: True, Execution order: 1

In Experiment files Section > File Assignement > Select the .txt files (all Sultana in the Filter field) and on the Assign files dialog:

- Script: *match_entitye_name.py*
- Parameters: ch1
- Only selected files checked

Platform tab

- Script: *gpr_sample.py*
- parameters: Gene name,Spot Mean Intensity (Cyanine5_060909_1136(1)),0
- Execution order: 2



ID	Name	Description	Gene nucleotide sequence
1	lcl|NC_004722.1_cds_NP_829910.1_1	lcl|NC_004722.1_cds_NP_829910.1_1 [gene=dnaA] [locus_tag=BC0001] [db_...	TTGGAAAATATCTCTGATTTATGGAATAGTGCCTTAAAAGAATTAGAAAAAAA...
2	lcl|NC_004722.1_cds_NP_829911.1_2	lcl|NC_004722.1_cds_NP_829911.1_2 [locus_tag=BC0002] [db_xref=GeneID:...	ATGCGTTTTACAATACAAAAAGACTATCTTGTAAGAGGTGTACAAGATGTAAT...
3	lcl|NC_004722.1_cds_NP_829912.1_3	lcl|NC_004722.1_cds_NP_829912.1_3 [locus_tag=BC0003] [db_xref=GeneID:...	ATGAAACGTATTAAAATTTCAACAGAGTATATTACACTAGGACAATTTTTAAA...
4	lcl|NC_004722.1_cds_NP_829913.1_4	lcl|NC_004722.1_cds_NP_829913.1_4 [gene=recF] [locus_tag=BC0004] [db_x...	TTGTTTATTTCAGAAATACAATTAAAAAACTATCGCAATTATGAAAAATTAGAG...
5	lcl|NC_004722.1_cds_NP_829914.1_5	lcl|NC_004722.1_cds_NP_829914.1_5 [gene=gyrB] [locus_tag=BC0005] [db_x...	GTGTCAATGGAACAAAAGCAAATGCAAGAAAATGCATATGATGAAAGTCAG...
6	lcl|NC_004722.1_cds_NP_829915.1_6	lcl|NC_004722.1_cds_NP_829915.1_6 [locus_tag=BC0006] [db_xref=GeneID:...	TTGATGTCAGACAATCAACAACAAGCACGAATTCGAGAAATTAATATTAGTC...
7	lcl|NC_004722.1_cds_NP_829916.1_7	lcl|NC_004722.1_cds_NP_829916.1_7 [locus_tag=BC0012] [db_xref=GeneID:...	ATGCATCAAACATCTTGTACTTGGCAGCAGTTAAGTTTCTTTTTTTCATCTCA...
8	lcl|NC_004722.1_cds_NP_829917.1_8	lcl|NC_004722.1_cds_NP_829917.1_8 [locus_tag=BC0013] [db_xref=GeneID:...	ATGTGGGAATCTAAATTTGTTAAAGAAGGTTTGACTTTTGATGATGTATTACT...
9	lcl|NC_004722.1_cds_NP_829918.1_9	lcl|NC_004722.1_cds_NP_829918.1_9 [locus_tag=BC0014] [db_xref=GeneID:...	GTGAAAGGTATGTTTTGCAAAAGATTCATTGCTTTGGTAACAGTGCTTACAC...
10	lcl|NC_004722.1_cds_NP_829919.1_10	lcl|NC_004722.1_cds_NP_829919.1_10 [locus_tag=BC0015] [db_xref=GeneID...	TTGTACATGACAAATGTAACAGGGACAGAACGTGTAAAACGTGGAATGGCA...
11	lcl|NC_004722.1_cds_NP_829920.1_11	lcl|NC_004722.1_cds_NP_829920.1_11 [locus_tag=BC0016] [db_xref=GeneID...	ATGGTGAAAATCGGTGTACTAGGTCTTCAAGGTGCAGTTCGTGAGCATGTA...
12	lcl|NC_004722.1_cds_NP_829921.1_12	lcl|NC_004722.1_cds_NP_829921.1_12 [locus_tag=BC0017] [db_xref=GeneID...	ATGCTTGATATTAAATTTTTACGTACAAACTTTGAAGAAGTAAAAGCGAAATT...
13	lcl|NC_004722.1_cds_NP_829922.1_13	lcl|NC_004722.1_cds_NP_829922.1_13 [locus_tag=BC0019] [db_xref=GeneID...	TTGACAACAGCGCATATAAGTGGAGATTGGAGAACTCGTTACCAACACGTA...
14	lcl|NC_004722.1_cds_NP_829923.1_14	lcl|NC_004722.1_cds_NP_829923.1_14 [locus_tag=BC0021] [db_xref=GeneID...	GTGACCGGAGTACCATTTATCACGGTTGAAGGACCAATTGGTGTTGGAAAA...
15	lcl|NC_004722.1_cds_NP_829924.1_15	lcl|NC_004722.1_cds_NP_829924.1_15 [locus_tag=BC0022] [db_xref=GeneID...	TTGCTCATTATCGATATGATTAATGACTTTCAATTTTCCCACGGACCCATTCT...
16	lcl|NC_004722.1_cds_NP_829925.1_16	lcl|NC_004722.1_cds_NP_829925.1_16 [locus_tag=BC0023] [db_xref=GeneID...	ATGGAACAAGATCAAGATATTTATTTTATGCAATTAGCGATAGAAGAAGCTAA...
17	lcl|NC_004722.1_cds_NP_829926.1_17	lcl|NC_004722.1_cds_NP_829926.1_17 [locus_tag=BC0024] [db_xref=GeneID...	GTGTCATACCAAGCGTTATACCGAACATGGAGACCGCAAAAGTTCGAAGAT...
18	lcl|NC_004722.1_cds_NP_829927.1_18	lcl|NC_004722.1_cds_NP_829927.1_18 [locus_tag=BC0025] [db_xref=GeneID...	ATGATGCGTGGCGGAATGGGAAATATGAATAACATGATGAAACAAATGCAAA...
19	lcl|NC_004722.1_cds_NP_829928.1_19	lcl|NC_004722.1_cds_NP_829928.1_19 [gene=recR] [locus_tag=BC0026] [db_...	ATGCATTATCCAGAACCAATATCAAAACTAATCGATAGTTTTATGAAGTTGCC...
20	lcl|NC_004722.1_cds_NP_829929.1_20	lcl|NC_004722.1_cds_NP_829929.1_20 [locus_tag=BC0027] [db_xref=GeneID...	ATGTTCTTTCAAAAAAAGGGTAAATTGCGTAAAGAGTATGACGATAAGTTAAT...
21	lcl|NC_004722.1_cds_NP_829930.1_21	lcl|NC_004722.1_cds_NP_829930.1_21 [locus_tag=BC0028] [db_xref=GeneID...	ATGAATTCTACAATTATTATTGTTGGTATTCTTACTTTAGTATTTATTTTTCTTG...
22	lcl|NC_004722.1_cds_NP_829931.1_22	lcl|NC_004722.1_cds_NP_829931.1_22 [locus_tag=BC0034] [db_xref=GeneID...	ATGAAATTACAAAGTGAAGTTTGTATTGTTTGTGAGACAAAAAGAAAAGAAG...
23	lcl|NC_004722.1_cds_NP_829932.1_23	lcl|NC_004722.1_cds_NP_829932.1_23 [locus_tag=BC0035] [db_xref=GeneID...	ATGAATCAAAATCGTATGCCTTTATATGAGGCGTTAATAGAGTTTAAAGAAAG...
24	lcl|NC_004722.1_cds_NP_829933.1_24	lcl|NC_004722.1_cds_NP_829933.1_24 [gene=tmk] [locus_tag=BC0036] [db_x...	ATGAAGGGATTATTTGTAACAATTGAGGGCCCAGAAGGTTCAGGTAAAACA...
25	lcl|NC_004722.1_cds_NP_829934.1_25	lcl|NC_004722.1_cds_NP_829934.1_25 [locus_tag=BC0037] [db_xref=GeneID...	ATGACGAAGACGTGGGAGCAGCTTTCTGCTATACAACCCATCGGTGTAAAA...
26	lcl|NC_004722.1_cds_NP_829935.1_26	lcl|NC_004722.1_cds_NP_829935.1_26 [locus_tag=BC0038] [db_xref=GeneID...	TTGTATGATGTAGTAGGTGTTCGCTTTAAGAAGGCCGGAAAGGTATATTACT...
27	lcl|NC_004722.1_cds_NP_829936.1_27	lcl|NC_004722.1_cds_NP_829936.1_27 [locus_tag=BC0039] [db_xref=GeneID...	GTGCTTGTGGAGAAAAAAGATATTTTTGAATCGGTTTCTAGTATGGAAGAGC...
28	lcl|NC_004722.1_cds_NP_829937.1_28	lcl|NC_004722.1_cds_NP_829937.1_28 [locus_tag=BC0040] [db_xref=GeneID...	ATGAATTTATATGAAGATGAACGTTTAGATTACTTATTAGCACAAGACATGAA...
29	lcl|NC_004722.1_cds_NP_829938.1_29	lcl|NC_004722.1_cds_NP_829938.1_29 [locus_tag=BC0041] [db_xref=GeneID...	ATGTGGCAACAAAAAAGCTTTCAACAAAATGAAAAGGGAGTTTTATATTTAG...
30	lcl|NC_004722.1_cds_NP_829939.1_30	lcl|NC_004722.1_cds_NP_829939.1_30 [locus_tag=BC0042] [db_xref=GeneID...	ATGAAATCTACTGGTATTGTTCGTAAAGTTGATGAATTAGGTCGTGTAGTAAT...
31	lcl|NC_004722.1_cds_NP_829940.1_31	lcl|NC_004722.1_cds_NP_829940.1_31 [gene=metG] [locus_tag=BC0043] [db...	ATGACAGAGGAAAATAAGTCCTTTTATATTACTACCCCAATTTATTATCCAAG...
32	lcl|NC_004722.1_cds_NP_829941.1_32	lcl|NC_004722.1_cds_NP_829941.1_32 [locus_tag=BC0044] [db_xref=GeneID...	TTGTTTGATACACATTCACATTTGAATGCAGAGCAATTCGAAGGAGATTTGC...
33	lcl|NC 004722.1 cds NP 829942.1 33	lcl|NC 004722.1 cds NP 829942.1 33 [locus tag=BC0045] [db xref=GeneID...	GTGGAGGCAAGCATGAAAATAAAAGAGATTATCGTTGTAGAAGGTAAAGAT...

Do the same again for the ch2 but use as Parameters for Platform:

Platform tab

- Script: *gpr_sample.py*
- parameters: Gene name,Spot Mean Intensity (Cyanine3_060909_1136(1)),0
- Execution order: 2

for Platform GPL10439:

- In Experiment files Section > File Assignement > Select the .ndf file and on the Assign files dialog":
  - Script: *match_entity_type_param.py*
  - Param: platform
  - Only selected files checked
  - Platform tab > Script: > *soft_platform.py*, Execution order: 2
- In Experiment files Section > File Assignement > Select the .txt files (all pair files) and on the Assign files dialog:
  - Script: *match_entity_name.py*
- Parameters: ch1
  - Only selected files checked
  - Platform tab > Script: > *gpr_sample.py*; Execution; order: 2
  - Parameters: ID_REF,Spot Mean Intensity (Alexa555_101810_0935(1)),0
- Parameters: ch2
  - Only selected files checked

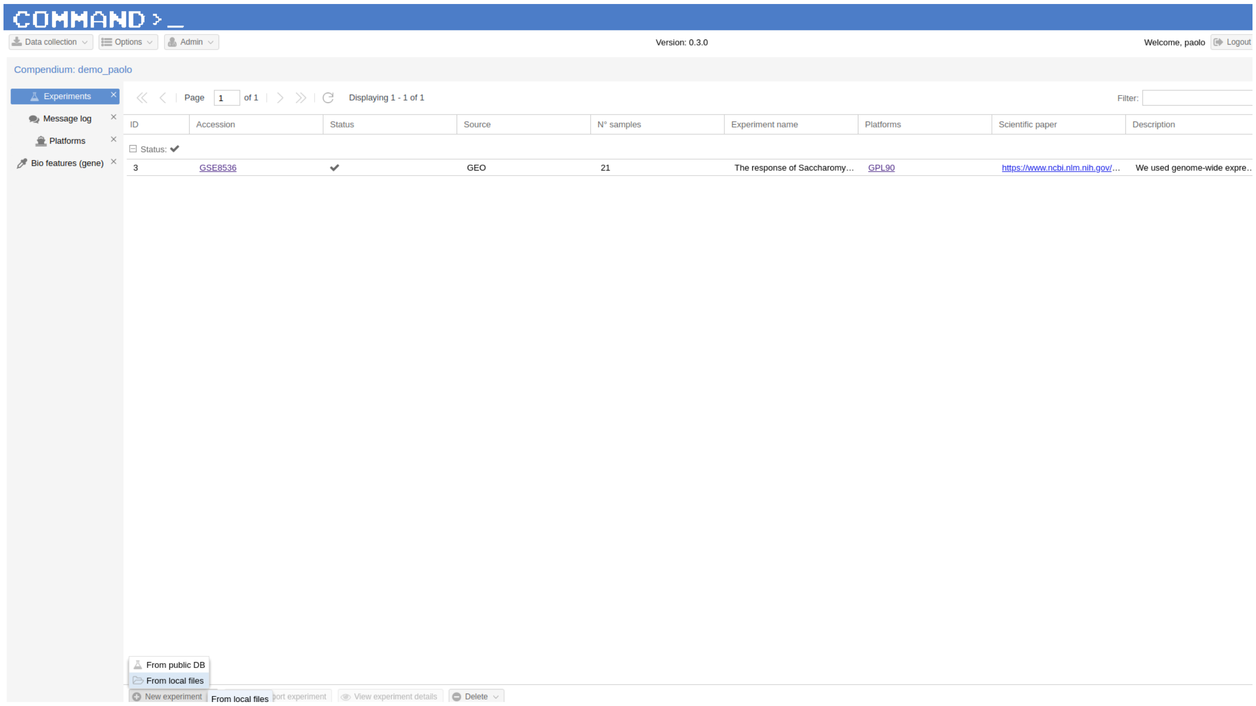
  - Platform tab > Script: > *gpr_sample.py*; Execution; order: 2
  - Parameters: ID_REF,Spot Mean Intensity (Alexa647_111510_1227(1))

7.4 Use Case - Import experiment from local file

In order to import an experiment which is not available from public repositories the user needs to provide:

- a yaml file (see an example: `here`) containing the description of the experiment to be imported: The first row contains the Experiment id, the other rows start with the Platform id followed by the Samples ids.
- a single compressed file (either zip or tar.gz) containing the raw data.

Go to Experiments > New Experiment (bottom-left) > From local file

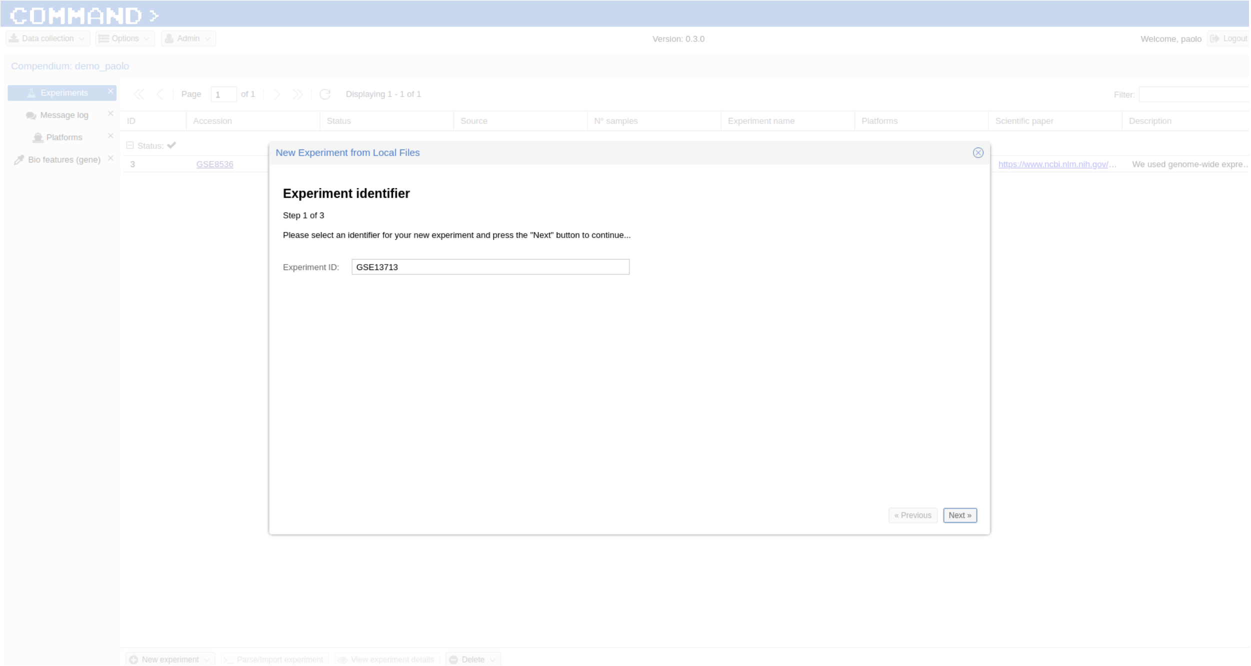


ID	Accession	Status	Source	N° samples	Experiment name	Platforms	Scientific paper	Description
3	GSE8536	✓	GEO	21	The response of Saccharomy…	GPL90	https://www.ncbi.nlm.nih.gov/…	We used genome-wide expre…

Fill the form which popped up starting with Experiment ID (the same contained in the yaml file, GSE13713 for the embedded example) then upload the yaml file (the system will take care to check if the format is ok), finally upload the compressed data. In a while your experiment is going to be imported.

7.5 Use Case - RNA-Seq

Similarly to the microarray cases, RNA-Seq experiments can be retrieved from public database, specifically the Sequence Read Archive (SRA) , from the New Experiment/From public DB interface (bottom-left border icon). Here we select a small RNA-Seq experiment from SRA (PRJNA471071) where the authors employed a computational model of underground metabolism and laboratory evolution experiments to examine the role of enzyme promiscuity in the acquisition and optimization of growth on predicted non-native substrates in E. coli K-12 MG1655.

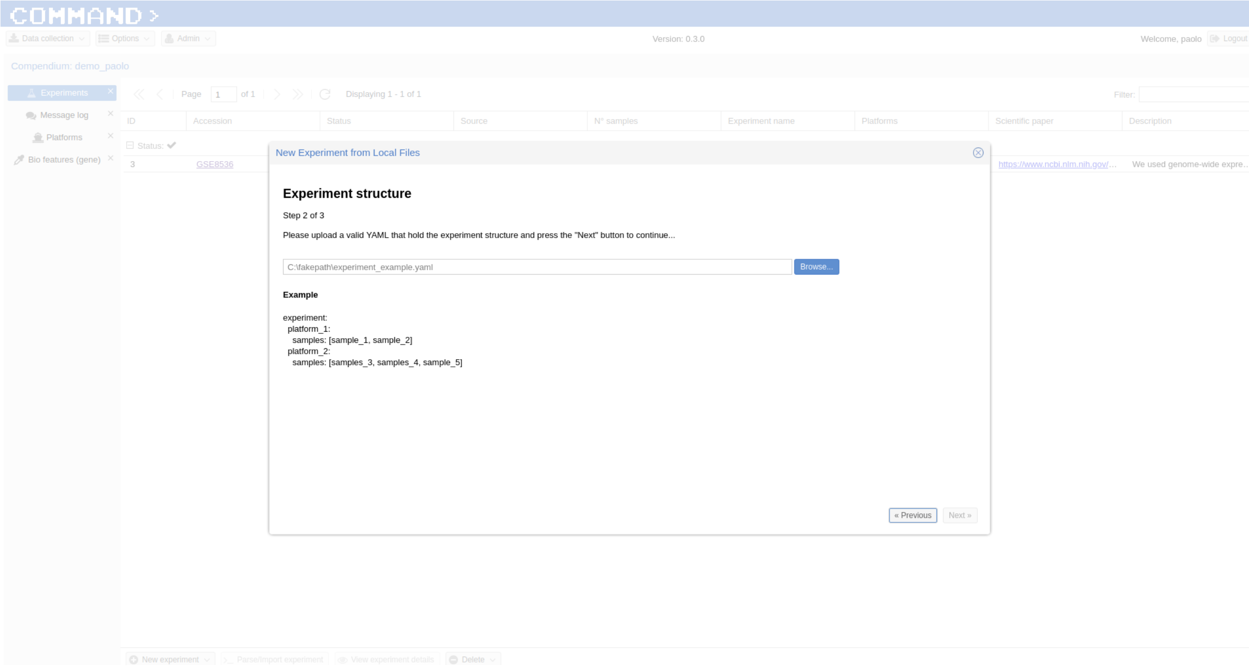


Experiment identifier

Step 1 of 3

Please select an identifier for your new experiment and press the "Next" button to continue...

Experiment ID: GSE13713



Experiment structure

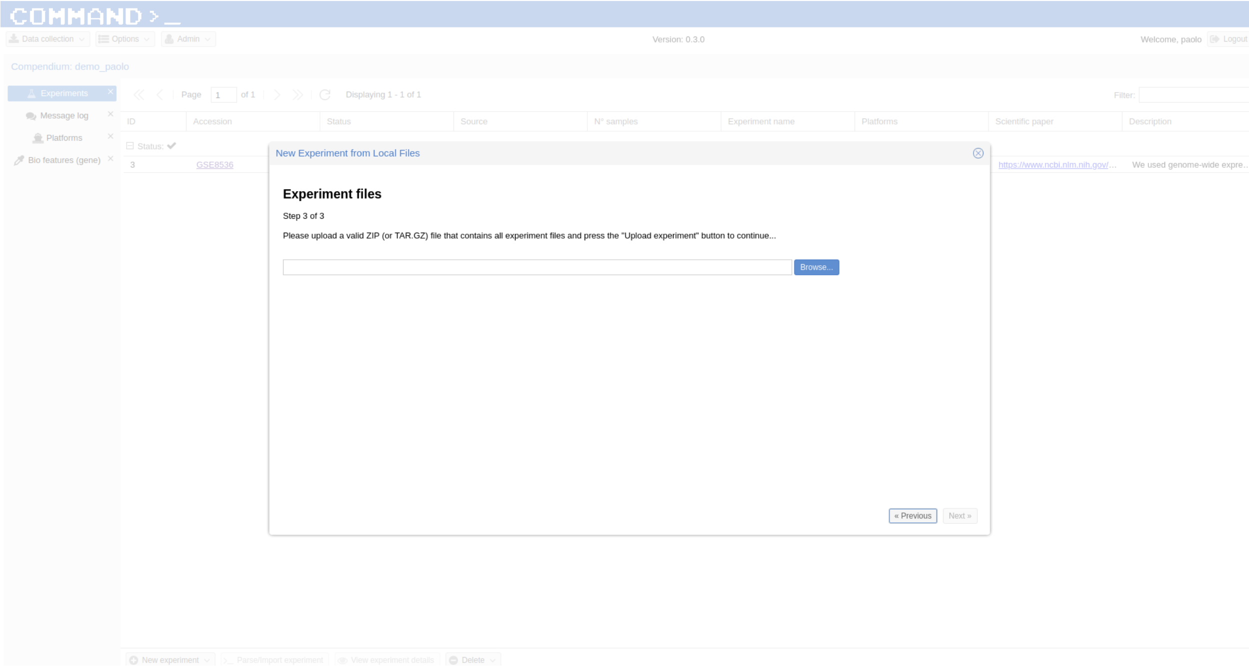
Step 2 of 3

Please upload a valid YAML that hold the experiment structure and press the "Next" button to continue...

C:\fakepath\experiment_example.yaml

Example

```
experiment:
  platform_1:
    samples: [sample_1, sample_2]
  platform_2:
    samples: [samples_3, samples_4, sample_5]
```

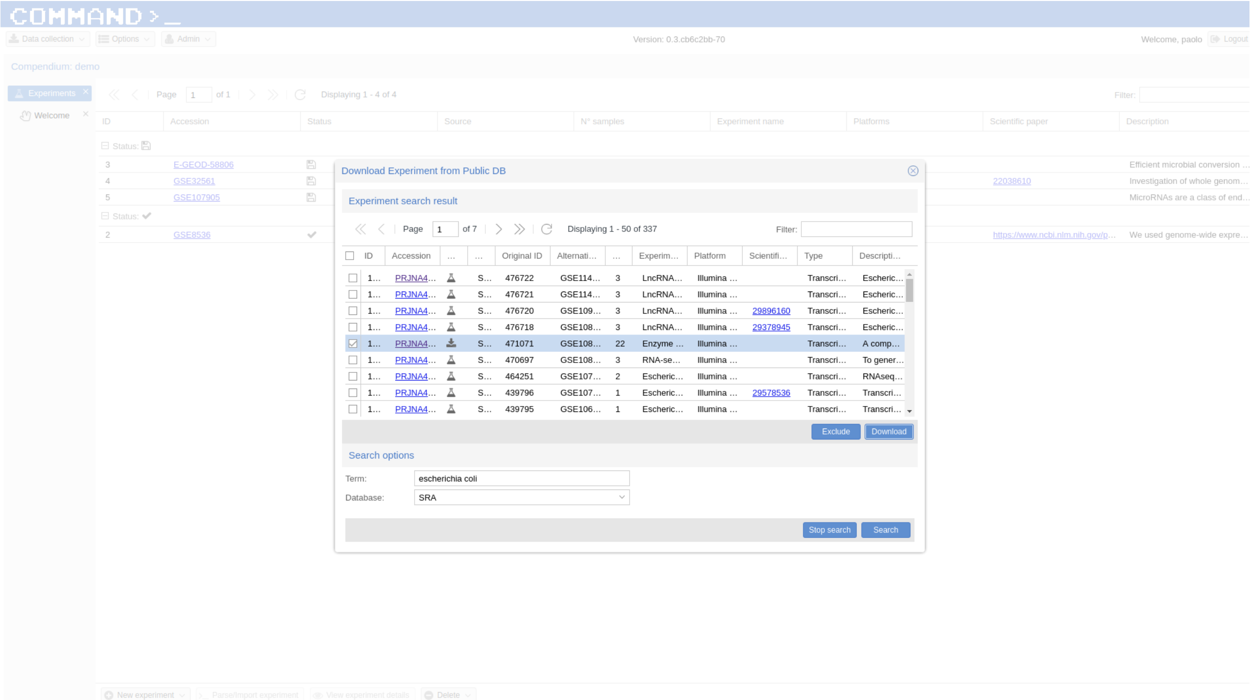


New Experiment from Local Files

Experiment files

Step 3 of 3

Please upload a valid ZIP (or TAR.GZ) file that contains all experiment files and press the "Upload experiment" button to continue...



Download Experiment from Public DB

Experiment search result

Page 1 of 7 Displaying 1 - 50 of 337

☐	ID	Accession	...	...	Original ID	Alternati...	...	Experim...	Platform	Scientifi...	Type	Descripti...
☐	1...	PRJNA4...		S...	476722	GSE114...	3	LncRNA...	Illumina ...		Transcri...	Escheric...
☐	1...	PRJNA4...		S...	476721	GSE114...	3	LncRNA...	Illumina ...		Transcri...	Escheric...
☐	1...	PRJNA4...		S...	476720	GSE109...	3	LncRNA...	Illumina ...	29896160	Transcri...	Escheric...
☐	1...	PRJNA4...		S...	476718	GSE108...	3	LncRNA...	Illumina ...	29378945	Transcri...	Escheric...
☑	1...	PRJNA4...		S...	471071	GSE108...	22	Enzyme ...	Illumina ...		Transcri...	A comp...
☐	1...	PRJNA4...		S...	470697	GSE108...	3	RNA-se...	Illumina ...		Transcri...	To gener...
☐	1...	PRJNA4...		S...	464251	GSE107...	2	Escheric...	Illumina ...		Transcri...	RNAseq...
☐	1...	PRJNA4...		S...	439796	GSE107...	1	Escheric...	Illumina ...	29578536	Transcri...	Transcri...
☐	1...	PRJNA4...		S...	439795	GSE106...	1	Escheric...	Illumina ...		Transcri...	Transcri...

Search options

Term: escherichia coli

Database: SRA

7.5.1 Indexing

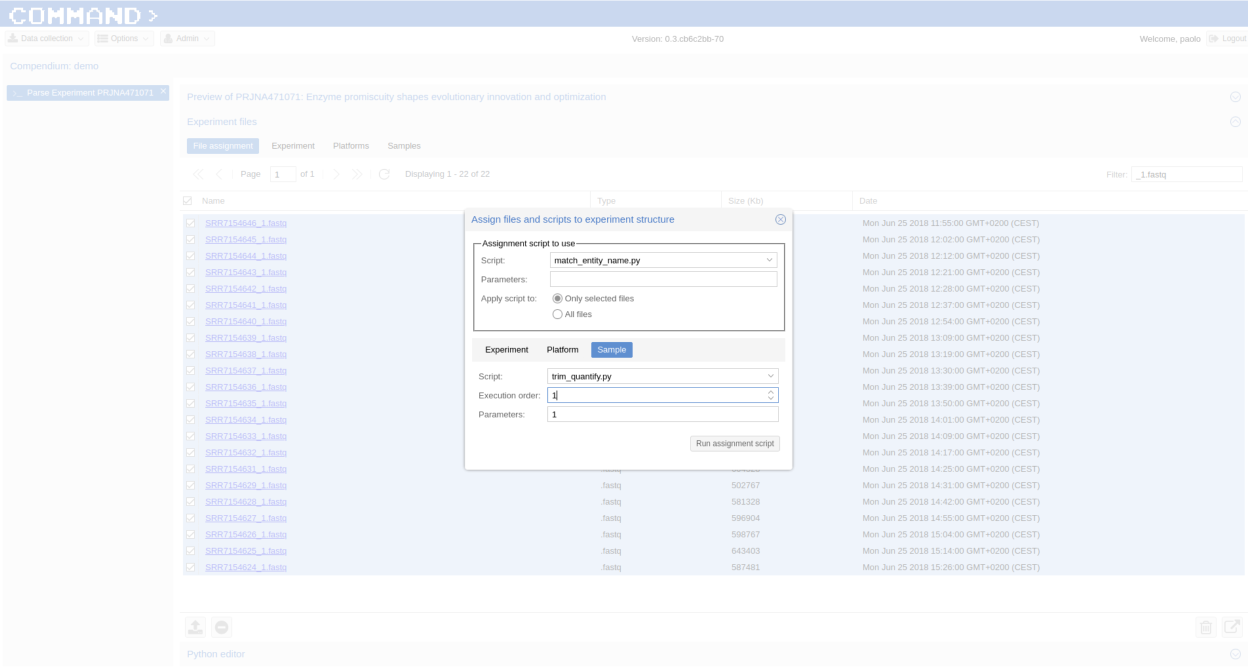
The first step is to build the index for the quasi-alignment mapper (kallisto here[1]): select *demo.fasta*, It contains the sequences for the genes of the Escherichia coli genome and it is automatically build by COMMAND>_ when you begin parsing the data.

Use Assignment Script (bottom-right corner icon) > from the dialog:*match_entity_name.py* > Only selected files Experiment tab > Script: > *kallisto_index.py*, Execution order: 1 > Run assignment script

7.5.2 RNA-Seq pre-processing and summarization

Since the experiment is paired-end, the default script for preprocessing and summarization requires to indicate only one of the two paired files. You can do it using the filter and selecting **1.fastq*, the script will take care of the rest.

Use Assignment Script (bottom-right corner icon) > from the dialog:*match_entity_name.py* > Only selected files Experiment tab > Script: > *trim_quantify.py*, Execution order: 1, Parameters: 1 (being a paired end)



Assign files and scripts to experiment structure

Assignment script to use

Script: match_entity_name.py

Parameters:

Apply script to: ● Only selected files ○ All files

Experiment | Platform | Sample

Script: trim_quantify.py

Execution order: 1

Parameters: 1

Run assignment script

7.5.3 Run assignment script

After a while all the sample will be preprocessed and summarized and the experiment can be imported from the Preview section: bottom-right corner > Import whole experiment.

---

[1] Nicolas L Bray, Harold Pimentel, Páll Melsted and Lior Pachter, Near-optimal probabilistic RNA-seq quantification, Nature Biotechnology 34, 525–527 (2016), doi:10.1038/nbt.3519

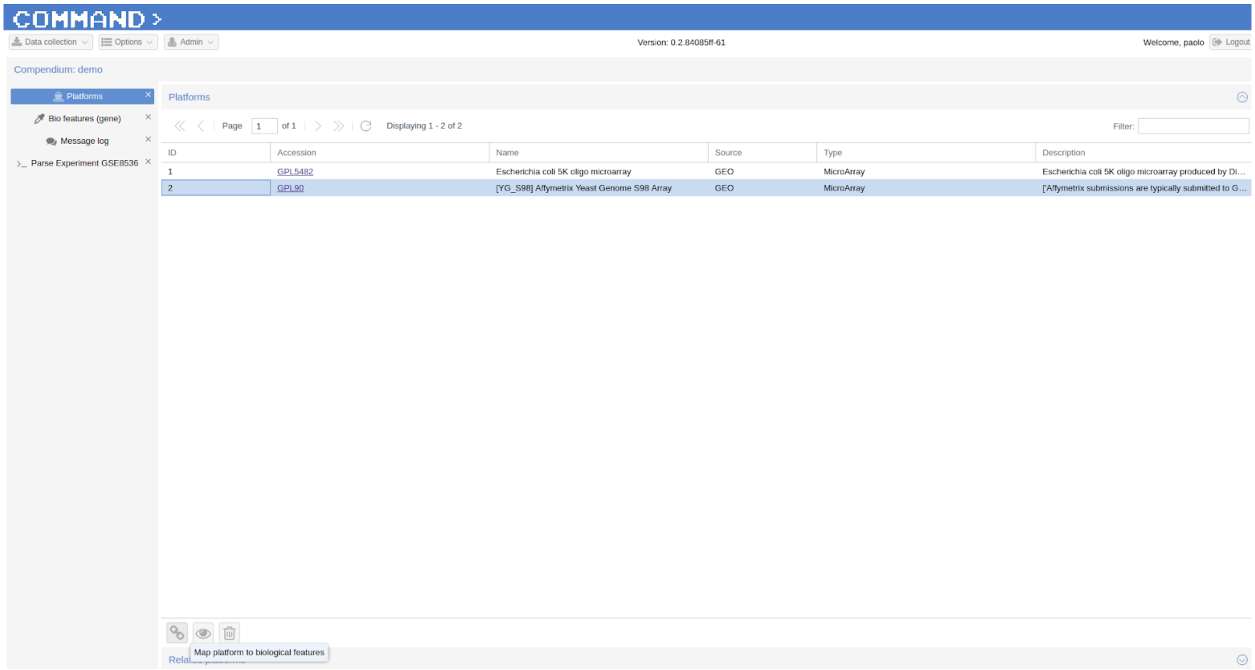
CHAPTER 8

Mapping probes and export the gene expression matrix

If you are done with importing experiments you can now map the probes to genes using BLAST[2] and a double filtering GUI of COMMAND>_. Go to Platform, select the platform to be mapped (e.g. GPL90 from the Affymetrix Use Case) and click the chain icon (map platform to biological features) on the bottom left corner.

Now you can use the dialog to run BLAST and filter the data (here we use the default settings).

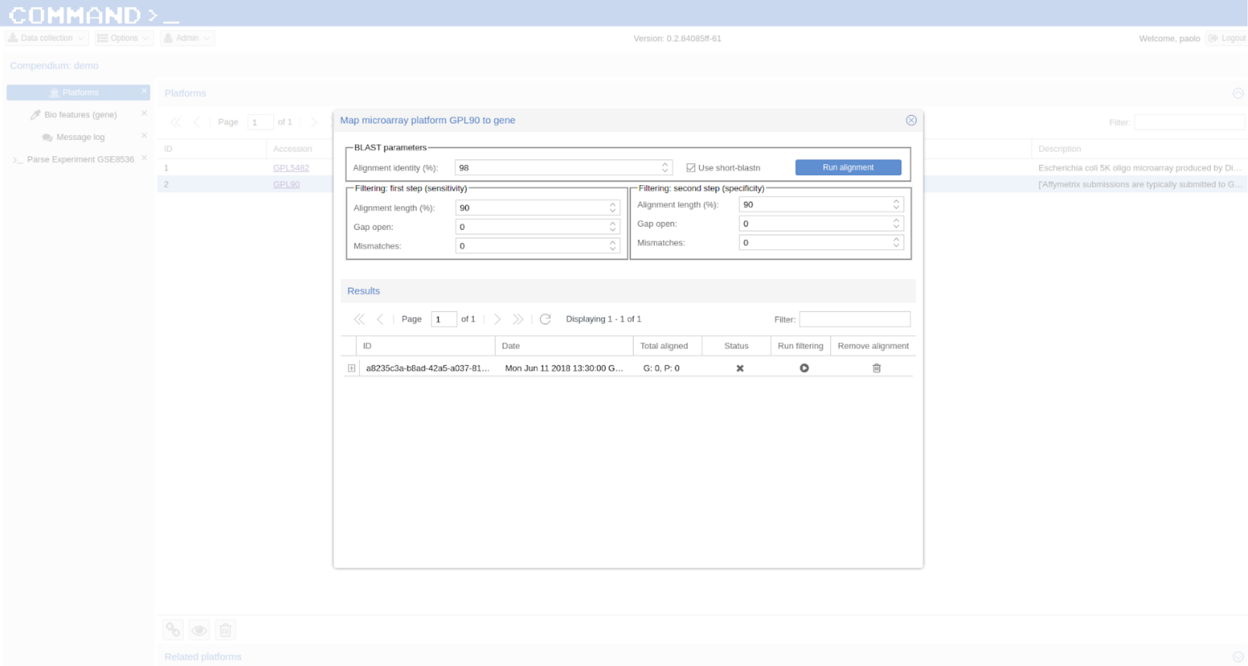
When your are fine with filtering you can use one of the selected filtered objects and download the expression matrix going to Options > Export.



ID	Accession	Name	Source	Type	Description
1	GPL5482	Escherichia coli 5K oligo microarray	GEO	MicroArray	Escherichia coli 5K oligo microarray produced by Di…
2	GPL90	[YG_S98] Affymetrix Yeast Genome S98 Array	GEO	MicroArray	['Affymetrix submissions are typically submitted to G…

---

[2] Altschul, S.F., Gish, W., Miller, W., Myers, E.W. & Lipman, D.J. (1990) “Basic local alignment search tool.” J. Mol. Biol. 215:403-410.



Map microarray platform GPL90 to gene

BLAST parameters

Alignment identity (%): 98

☑ Use short-blastn

Run alignment

Filtering: first step (sensitivity)

Alignment length (%): 90

Gap open: 0

Mismatches: 0

Filtering: second step (specificity)

Alignment length (%): 90

Gap open: 0

Mismatches: 0

Results

Page 1 of 1 Displaying 1 - 1 of 1 Filter:

ID	Date	Total aligned	Status	Run filtering	Remove alignment
a8235c3a-b8ad-42a5-a037-81…	Mon Jun 11 2018 13:30:00 G…	G: 0, P: 0	✖	▶	🗑

Tip: You can filter the data with different parameters, each set of parameters is saved in a specific slot.

References

CHAPTER 9

Python parsing scripts

The Experiment Object, Platform Object and Sample Object are Python objects used as proxy to import a new experiment in the database

The file name of the experiment, platform or sample is stored in the variable named **INPUT_FILE** The name of the entity (experiment, platform or sample name) is stored in the variable named **ENTITY_NAME** To access parameters passed to each script use the list **PARAMETERS** Within each entity (experiment, platform or sample) you can choose the execution order of the script using the Order column.

To access Experiment Object use the **EXPERIMENT_OBJECT** variable in the Python script used with experiment files.

9.1 EXPERIMENT_OBJECT variables

EXPERIMENT_OBJECT.experiment_access_id: (string) the experiment access id **EXPERIMENT_OBJECT**.experiment_name: (string) the experiment name **EXPERIMENT_OBJECT**.scientific_paper_ref: (string) pubblication associated to the experiment **EXPERIMENT_OBJECT**.description: (string) the experiment description

To access Platform Object use the **PLATFORM_OBJECT** variable in the Python script used with platform files.

9.2 PLATFORM_OBJECT variables

PLATFORM_OBJECT.platform_access_id (string) the platform access id **PLATFORM_OBJECT**.platform_name (string) the platform name **PLATFORM_OBJECT**.platform_type (string) ‘microarray or rna-seq’ **PLATFORM_OBJECT**.description (string) the platform description **PLATFORM_OBJECT**.add_bio_feature_reporter_data(name, description, **kwargs): add a reporter to the platform

**kwargs are platform_type dependent. i.e. for ‘microarray’ they are probe_access_id, probe_set_name, probe_type and sequence

To access Sample Object use the **SAMPLE_OBJECT** variable in the Python script used with sample files.

9.3 SAMPLE_OBJECT variables and methods

SAMPLE_OBJECT.sample_name (string) the sample name **SAMPLE_OBJECT**.description (string) the sample description **SAMPLE_OBJECT**.add_raw_data(bio_feature_reporter_name, value): add raw data of this sample

9.4 parsing_scripts package

9.4.1 Subpackages

parsing_scripts.experiment package

Submodules

parsing_scripts.experiment.kallisto_index module

`parsing_scripts.experiment.kallisto_index.`**`main`**`()`

Create an index file for the KALLISTO software using the current BIOLOGICAL FEATURES

Biologial features for this compendium are putted into a FASTA file that is then indexed to be used for RNA-seq quantification using KALLISTO

PARAMETERS: None

parsing_scripts.experiment.soft_experiment module

`parsing_scripts.experiment.soft_experiment.`**`main`**`()`

Parse a SOFT file and extract EXPERIMENT information

Looks for **accession number**, **experiment name**, **scientific paper**, **experiment description**

PARAMETERS: None

Module contents

parsing_scripts.file_assignment package

Submodules

parsing_scripts.file_assignment.assign_all module

`parsing_scripts.file_assignment.assign_all.`**`assign`**(*input_files*, *entity*, *entity_type*, *parameters*)

Assign the selected input files (or all the files if checked) to every selected ENTITY

For each ENTITY (experiment, platforms or samples) for which a parsing script is selected, all the (selected) input files will be assigned regardless.

PARAMETERS: None

parsing_scripts.file_assignment.match_entity_name module

`parsing_scripts.file_assignment.match_entity_name.`**`assign`**`(`*input_files*, *entity*, *entity_type*, *parameters*`)`

Assign the selected input files (or all the files if checked) to every ENTITY with matching NAME

For each ENTITY (experiment, platforms or samples) for which a parsing script is selected, only the (selected) input files with a name that match the one of the entity will be assinged (for example a file name GSE123.soft would match the experiment entity GSE123).

PARAMETERS: None

Module contents

parsing_scripts.platform package

Submodules

parsing_scripts.platform.adf_platform module

`parsing_scripts.platform.adf_platform.`**`main`**`()`

Parse an ADF file and extract PLATFORM information

Looks for **accession number**, **platform name**, **platform type** and **platform description**

PARAMETERS: *param1* (string): The original probe id field. If it is composed by more than one field, put all of them separated with a |. For example X|Y

param2 (bool): If True (or 1 or a non-empty string) the probe information (sequence) will be added

parsing_scripts.platform.cdf_platform module

`parsing_scripts.platform.cdf_platform.`**`main`**`()`

Parse a CDF file (Affymetrix) and extract PLATFORM information

Looks for **probe set name** and **probe id**. Please note that CDF does not contain probe sequence, for that information refer to cdf_platform_fasta.py

PARAMETERS: None

parsing_scripts.platform.cdf_platform_fasta module

`parsing_scripts.platform.cdf_platform_fasta.`**`main`**`()`

Parse a FASTA file containing probe sequences

This script is usually used before cdf_platform.py in order to get the probe sequence information that a CDF file doesn't provide.

PARAMETERS: None

parsing_scripts.platform.csv_platform module

`parsing_scripts.platform.csv_platform.main()`

Parse a CSV file containing probe sequences

A CSV file containing probe information is parsed and probes get added to the platform. This script is usually used together with other PLATFORM scripts

PARAMETERS: *param1* (string): The probe id field

param2 (string): The probe sequence

parsing_scripts.platform.gpr_platform module

`parsing_scripts.platform.gpr_platform.main()`

Parse a GPR file containing PLATFORM information and probe sequences

A GPR file is a TAB-delimited file with headers and complete platform information (descriptions and probe sequences)

PARAMETERS: *param1* (int): Number of lines to skip

param2 (string): The column header to parse out the original probe id field. If it is composed by more than one field, put all of them separated with a |. For example X|Y (actual probe ids will be concatenated with dots . in that case)

param3 (string): The column header of the probe **sequence** you want to parse out

param4 (string): DEPRECATED - The column header to parse out the DB 'gene_map_content' field; if multiple seperate with a pipe | (actual probe ids will be concatenated with dots . in that case)

param5 (string): The column header to parse out probe name field. If it is composed by more than one field, put all of them separated with a |. For example X|Y (actual probe ids will be concatenated with dots . in that case)

param6 (string): The column header to parse out probe set name field

param7 (bool): Ensure that orgiginal probe id in SAMPLE_OBJECT will be unique (defaults to False)

parsing_scripts.platform.ndf_platform module

`parsing_scripts.platform.ndf_platform.main()`

Parse a NDF file containing probe sequences

A NDF file is an ArrayExpress file that contains probe sequences. They have a header file with X and Y position for the probe, the SEQUENCE field and a PROBE_ID field. The combination of X.Y is used to store the probe id and ensure that is a unique name

PARAMETERS: *param1* (int): Number of lines to skip

parsing_scripts.platform.soft_platform module

`parsing_scripts.platform.soft_platform.main()`

Parse a SOFT file and extract PLATFORM information

Looks for **accession number**, **platform name**, **platform type** and **platform description**. If True is passed as parameter it will look for probe sequence information in the data table part of the file

PARAMETERS: *param1* (bool): Read the data table information (default False)

Module contents

parsing_scripts.sample package

Submodules

parsing_scripts.sample.cel_sample module

`parsing_scripts.sample.cel_sample.`**`main`**`()`

Parse a CEL (Affymetrix) file and extract SAMPLE raw data

The probe original id is given by X.Y

PARAMETERS: None

parsing_scripts.sample.gpr_sample module

`parsing_scripts.sample.gpr_sample.`**`main`**`()`

Parse a GPR file and extract SAMPLE raw data

A GPR file is a TAB-delimited file with headers and complete sample raw data information

PARAMETERS: *param1* (string): The column header of the original probe id to parse out. If it is composed by more than one field, put all of them separated with a |. For example X|Y (actual probe ids will be concatenated with dots . in that case)

param2 (string): The column header of the data value you want to parse out

param3 (int): Number of lines to skip

param4 (int): The sample channel (optional)

parsing_scripts.sample.pair_sample module

`parsing_scripts.sample.pair_sample.`**`main`**`()`

Parse a PAIR file and extract SAMPLE raw data

A PAIR file is a TAB-delimited file with headers and complete sample raw data information The probe id is given by X.Y to ensure uniqueness and the raw data value is taken from the PM column

PARAMETERS: *param1* (int): Number of lines to skip

parsing_scripts.sample.soft_sample module

`parsing_scripts.sample.soft_sample.`**`main`**`()`

Parse a SOFT file and extract SAMPLE description and optionally raw data

PARAMETERS: *param1* (string): The raw data value field, if empty it will be assigned automatically using the sample_column_identifier function

parsing_scripts.sample.trim_quantify module

`parsing_scripts.sample.trim_quantify.`**`main`**`()`

Trim a FASTQ file using Trimmomatic and quantify using KALLISTO

The result counts will be added to the SAMPLE OBJECT

PARAMETERS: *param1* (bool): True if this FASTQ file has a PAIRED file (forward or reverse), default False

Module contents

parsing_scripts.utils package

Submodules

parsing_scripts.utils.column_identifier module

`parsing_scripts.utils.column_identifier.`**`sample_column_identifier`**`(`*query*, *header*`)`

Tries to automatically identify the header column that contains the raw data given some query information (like the dye color)

Multi-channel array might have different dye color on different samples (dye-swap) and thus it would be tedious to manually define it for each single sample. This function tries to do it for you and is tipically invoked for the SOFT sample files.

PARAMETERS: *query* (string): The query string is usually something that contains information about the color i.e. cy3, red, green etc.

header (list): The header is a list of string from which to chose one that will match the query

parsing_scripts.utils.rnaseq module

`parsing_scripts.utils.rnaseq.`**`create_fasta`**`(`*file*, *compendium*`)`

Create a FASTA file using the BIOLOGICAL FEATURE of the current Organism

PARAMETERS: *file* (string): The output FASTA file name

compendium (string): The organism (nick) name

Module contents

9.4.2 Module contents

CHAPTER 10

COMMAND>_ for developers

In order to add new features to COMMAND>_ you'll need to stick with the whole framework. As a demonstration we will create a basic page to retrieve some data from the database and show them in a grid within COMMAND>_. So we will take care of:

- create the ExtJS interface;
- create the Python view;
- create the permission to access the view;
- make an AJAX call passing parameters;
- perform a job on celery to run in background;
- handle websocket to show the results on a grid;

We will also see how to extend COMMAND>_ functionalities such as how to add a new public database users can use to perform search on, how to add a new platform type and so on.

Note: For anything else related to the interface design please refer to the ExtJS documentation. While to properly add new models and extend the Data Model, please refer to the Django documentation

10.1 Add brand new feature in COMMAND>_

10.1.1 Create the ExtJS interface

COMMAND>_ is a single-page application, so everything you see runs within one HTML file and the Javascript code needed to display the interface is loaded and managed by the ExtJS framework. All ExtJS interface files (views) live within the directory `command/static/command/js/ext-js/app/view`. So let's create a `test` directory in here and, within that directory let's create 2 files: `Test.js` and `TestController.js`.

Let's fill these two files with some basic code like the following:

```
// Test.js

Ext.define('command.view.test.Test', {
  extend: 'Ext.Component',

  xtype: 'test',

  title: 'Test',

  requires: [
      'Ext.panel.Panel',
      'command.view.test.TestController'
  ],

  controller: 'test',

  store: null,

  alias: 'widget.test',

  itemId: 'test',

  reference: 'test',

  viewModel: {},

  html: 'TEST',

  listeners: {
      //
  },

  initComponent: function() {
      this.callParent();
  },

  destroy: function() {
      this.callParent();
  }
});
```

```
// TestController.js

Ext.define('command.view.test.TestController', {
  extend: 'Ext.app.ViewController',

  alias: 'controller.test'
});
```

Now you will need to run the command `sencha app build` from within the `command/static/command/js/ext-js` directory.

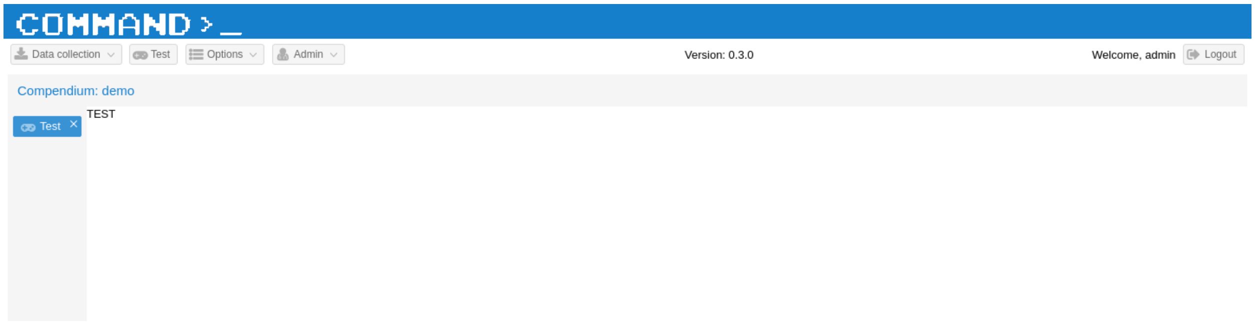
Note: To use the `sencha app build` command you will need to download and install Sencha CMD

Now you should be able to point your browser to http://localhost/#view/test and see that the `Test` panel has been correctly loaded as a tab within the main application panel. To make it reachable with a button and to add a small icon next to the tab name we should edit two files, `Main.js` (here) and `Application.js` (here).

```
// Main.js
// Add the ``Test`` menu button

},{
    text: 'Test',
    itemId: 'test_menu_item',
    iconCls: null,
    glyph: 'xf11b',
    listeners: {
        click: {
            fn: 'onAction',
            hash: 'view/test',
            glyph: 'xf11b',
            panel: 'test'
        }
    }
},{
    text: 'Options',
    ...
```

```
// Application.js
// Add the ``test`` glyph

version: null,

panel_glyph: {
    'test': 'xf11b',
    ...
```

You should see something like the following:



10.1.2 Create the Python View code

Now let's create a grid, a basic double-click event and a link to a Python view. First of all we need to create the `test.py` file within the `views` directory (here). The basic view file should look something like that:

```
// test.py

import json
from django.http import HttpResponse
from django.views import View
from command.lib.utils.decorators import forward_exception_to_http
```

(continues on next page)

(continued from previous page)

```


class TestView(View):

   def get(self, request, operation, *args, **kwargs):
      method = getattr(self, operation)
      return method(request, *args, **kwargs)

   def post(self, request, operation, *args, **kwargs):
      method = getattr(self, operation)
      return method(request, *args, **kwargs)

   @staticmethod
   @forward_exception_to_http
   def test(request, *args, **kwargs):

     return HttpResponse(json.dumps({'success': True}),
                         content_type="application/json")
```

The `test` function does nothing at the moment and is meant to respond to an Ajax call. We'll see that within the same `TestView` class we will put both code to manage Ajax and WebSocket requests. Before we add any business logic code we need to tell COMMAND>_ that the ExtJS view `test` will make requests to the Python view `TestView` and that users need no specific privileges to do that (for the moment). So let's add one line in the `consumer.py` script (here):

```
# consumer.py

class Dispatcher:
   dispatcher = {
      ...
      ExportDataView: ['export_data'],
      TestView: ['test']
   }
```

10.1.3 Add a grid to the ExtJS interface

So far, so good. Let's remove the HTML code from the `Test.js` file and let's add a grid to show all the experiments for the selected compendium. The file will now look like this:

```
// Test.js

Ext.define('command.view.test.Test', {
   extend: 'command.Grid',

   xtype: 'test',

   title: 'Test',

   requires: [
     'Ext.panel.Panel',
     'command.view.test.TestController'
   ],

   controller: 'test',
```

(continues on next page)

(continued from previous page)

```

   store: null,

   alias: 'widget.test',

   itemId: 'test',

   reference: 'test',

   viewModel: {},

   mixins: {
      getRequestObject: 'RequestMixin'
   },

   command_view: 'test',

   command_read_operation: 'test_read',

   listeners: {
     //
   },

   columns: [{
     text: 'Accession',
     flex: 2,
     sortable: true,
     dataIndex: 'experiment_access_id',
   }, {
     text: 'Experiment name',
     flex: 2,
     sortable: true,
     tdCls: 'command_tooltip',
     dataIndex: 'experiment_name'
   }, {
     text: 'Scientific paper',
     flex: 2,
     sortable: true,
     dataIndex: 'scientific_paper_ref'
   }, {
     text: 'Description',
     flex: 2,
     sortable: true,
     tdCls: 'command_tooltip',
     dataIndex: 'description'
   }],

   initComponent: function() {
     this.store = Ext.create('command.store.Experiments');
     this.callParent();
   },

   destroy: function() {
     this.callParent();
   }
});
```

Please note that:

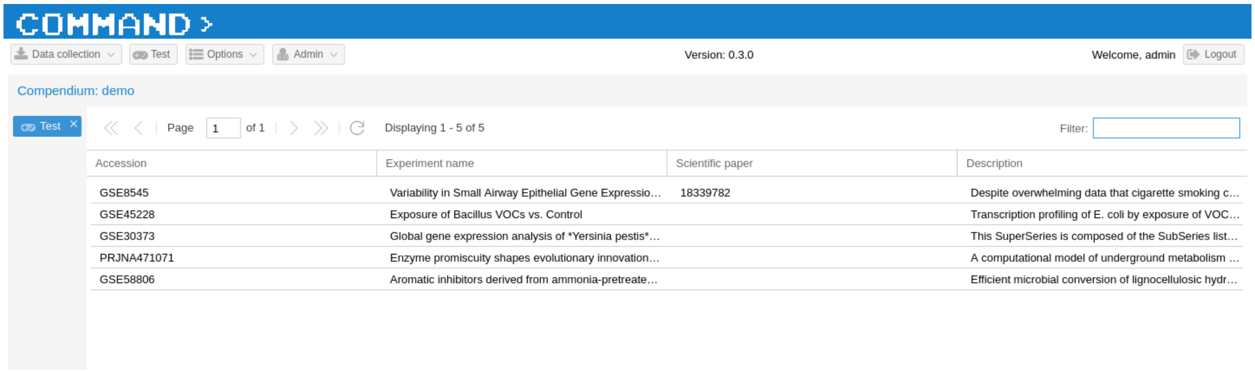
- at line `4` we extend `command.Grid`;
- at line `31` we are saying to COMMAND>_ the view to be used;
- at line `33` we are declaring the default read operation (i.e. the default Python function to be called);
- at line `64` we are declaring the ExtJS store to use.

10.1.4 Link the ExtJS grid to the Python code via WebSocket

The `test.py` Python view file will have a `test_read` function that will look like the following:

```python
# test.py

@staticmethod
@forward_exception_to_channel
def test_read(channel_name, view, request, user):
   channel = Channel(channel_name)

   start = 0
   end = None
   compendium = CompendiumDatabase.objects.get(id=request['compendium_id'])
   if request['page_size']:
      start = (request['page'] - 1) * request['page_size']
      end = start + request['page_size']
   order = ''
   if request['ordering'] == 'DESC':
      order = '-'

   query_response = Experiment.objects.using(compendium.compendium_nick_name). \
      filter(Q(experiment_access_id__icontains=request['filter']) |
             Q(scientific_paper_ref__icontains=request['filter']) |
             Q(description__icontains=request['filter']) |
             Q(experiment_name__icontains=request['filter']))
   try:
      query_response = query_response.order_by(order + request['ordering_value'])
   except Exception as e:
      pass

   total = query_response.count()
   query_response = query_response[start:end]

   channel.send({
      'text': json.dumps({
          'stream': view,
          'payload': {
              'request': request,
              'data': {
                  'experiments': [exp.to_dict() for exp in query_response],
                  'total': total
              }
          }
      })
   })
```

If you refresh your browser, you should now see something like the following:



Accession	Experiment name	Scientific paper	Description
GSE8545	Variability in Small Airway Epithelial Gene Expressio…	18339782	Despite overwhelming data that cigarette smoking c…
GSE45228	Exposure of Bacillus VOCs vs. Control		Transcription profiling of E. coli by exposure of VOC…
GSE30373	Global gene expression analysis of *Yersinia pestis*…		This SuperSeries is composed of the SubSeries list…
PRJNA471071	Enzyme promiscuity shapes evolutionary innovation…		A computational model of underground metabolism …
GSE58806	Aromatic inhibitors derived from ammonia-pretreate…		Efficient microbial conversion of lignocellulosic hydr…

As final step in this brief tutorial, let's add a double-click event on the grid to call the `test` function defined in the `TestView` Python view to run an empty job on the Celery task manager. When the job is done we'll have a callback function to show a message back on the interface. First thing is to add the event listener.

10.1.5 Create the Ajax call on double-click event

```
// Test.js

listeners: {
   itemdblclick: 'onTestDoubleClick'
},
```

Then we'll need to implement the `onTestDoubleClick` in the `TestController.js`

```
// TestController.js

onTestDoubleClick: function(dv, record, item, index, e) {
     var grid = dv.up('grid');
     var gridSelection = grid.getSelection();
     var request = grid.getRequestObject('test');
     request.values = JSON.stringify(gridSelection[0].data);
     Ext.Ajax.request({
         url: request.view + '/' + request.operation,
         params: request,
         success: function (response) {
             command.current.checkHttpResponse(response);
         },
         failure: function (response) {
             console.log('Server error', reponse);
         }
     });
 }
```

10.1.6 Manage asynchronous code using Celery and WebSocket

The `request` object is configured to automatically retrieve the view name (`request.view`) and setted to call the `test` function in the Python `TestView`.

```
# test.py

```

(continues on next page)

(continued from previous page)

```
@staticmethod
@forward_exception_to_http
def test(request, *args, **kwargs):
    values = json.loads(request.POST['values'])

    comp_id = request.POST['compendium_id']
    channel_name = request.session['channel_name']
    view = request.POST['view']
    operation = request.POST['operation']

    test.test_task.apply_async(
        (request.user.id, comp_id, values['id'], channel_name, view, operation)
    )

    return HttpResponse(json.dumps({'success': True}),
                        content_type="application/json")
```

With the `test.test_task.apply_async` we are calling the `test_task` function from the Celery task file `test.py` (not to be confused with the Python view file that have the same name). We need to create this file and implement the functionality. So let's create a file name `test.py` in the `command/command/lib/task` directory (here). The file will look like that:

```
# test.py

from __future__ import absolute_import, unicode_literals
from time import sleep
import celery
from channels import Channel
from command.lib.utils.message import Message


class TestCallbackTask(celery.Task):
    def on_success(self, retval, task_id, args, kwargs):
        user_id, compendium_id, path, channel_name, view, operation = args
        channel = Channel(channel_name)
        message = Message(type='info', title='Hello world!',
                          message='Hi there!'
                          )
        message.send_to(channel)

    def on_failure(self, exc, task_id, args, kwargs, einfo):
        pass


@celery.task(base=TestCallbackTask, bind=True)
def test_task(self, user_id, compendium_id, exp_id, channel_name, view, operation):
    sleep(1)
```

The `test_task` function simply wait for one seconds. When it's done the `on_success` callback function gets called and it retrieve the WebSocket channel name to send back a simple message. That message will be captured on the client side and a pop-up will appear. Before trying it out we need to inform Celery that there's an extra file to search for when calling a task. This is done in the Django setting file, here.

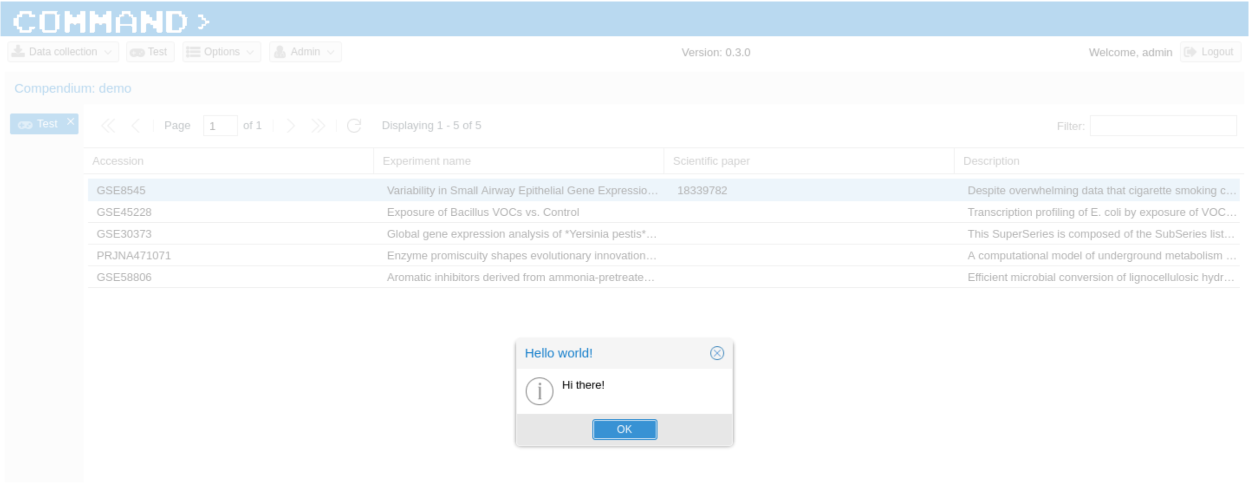
```
# settings.py

CELERY_IMPORTS = (
```

(continues on next page)

(continued from previous page)

```
    'command.lib.tasks.experiment_public',
    'command.lib.tasks.experiment_local',
    'command.lib.tasks.uncompress_file',
    'command.lib.tasks.run_file_assignment_script',
    'command.lib.tasks.run_parsing_script',
    'command.lib.tasks.parse_bio_feature_file',
    'command.lib.tasks.run_platform_mapper',
    'command.lib.tasks.import_experiment',
    'command.lib.tasks.import_platform_mapping',
    'command.lib.tasks.export_data',
    'command.lib.tasks.test',
)
```

You should now be able to double-click on a grid value and see something like this.



Accession	Experiment name	Scientific paper	Description
GSE8545	Variability in Small Airway Epithelial Gene Expressio…	18339782	Despite overwhelming data that cigarette smoking c…
GSE45228	Exposure of Bacillus VOCs vs. Control		Transcription profiling of E. coli by exposure of VOC…
GSE30373	Global gene expression analysis of *Yersinia pestis*…		This SuperSeries is composed of the SubSeries list…
PRJNA471071	Enzyme promiscuity shapes evolutionary innovation…		A computational model of underground metabolism …
GSE58806	Aromatic inhibitors derived from ammonia-pretreate…		Efficient microbial conversion of lignocellulosic hydr…

Hello world!

Hi there!

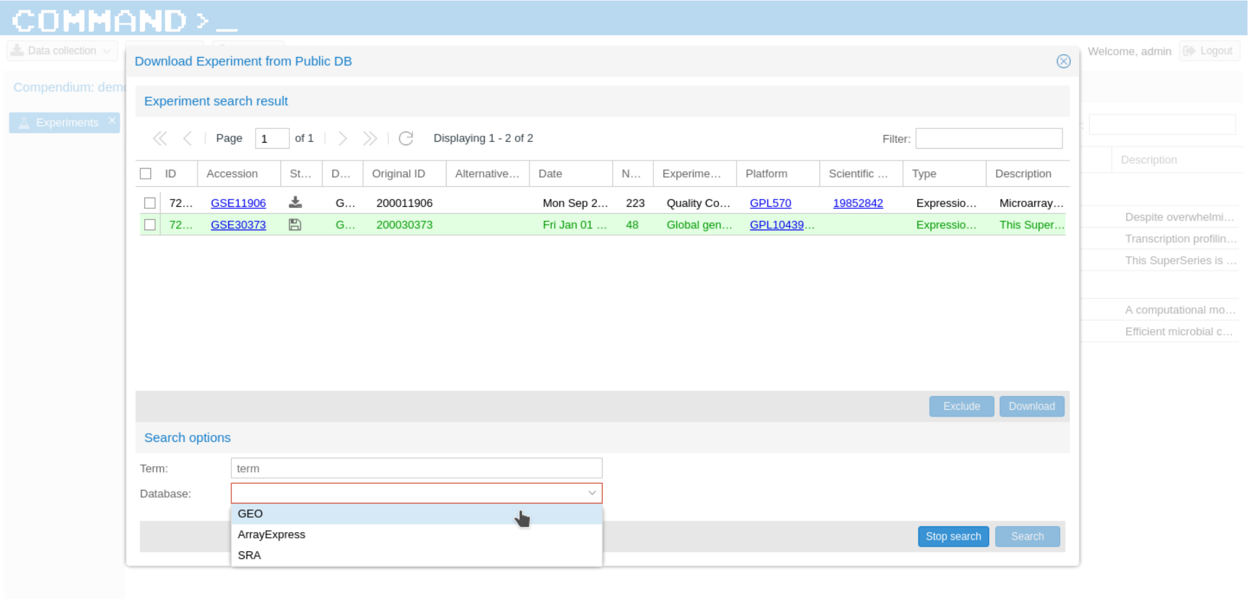
10.2 Add new public database manager

At the moment of writing, COMMAND>_ is able to search on GEO, ArrayExpress and SRA.

To add a new database on this list, you will need to add a line in a database table and extend one class. In the `command_datasource` database table you should add the source name and the class to handle it.

The class should be defined extending the class `PublicDatabase` that is defined here. This is an abstract class and to extend it you will need to implement three methods:

- `search`: it perform the actual search on the public database (through a REST call or FTP for example) and create one `ExperimentSearchResult` for each retrieved entry to be stored in the database;
- `download_experiment_files`: it is responsible to get all the data files related to one single `ExperimentSearchResult` and save them in the output directory;
- `create_experiment_structure`: starting from the information of the downloaded files, this method should create the *experiment*, *platform*, *sample* structures and save it using `Experiment`, `Platform` and `Sample` Django models.



Download Experiment from Public DB

Experiment search result

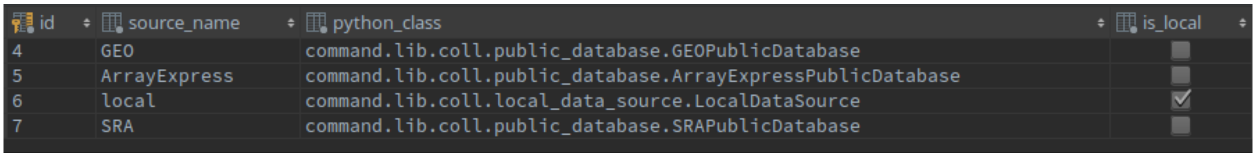
Page 1 of 1 Displaying 1 - 2 of 2 Filter:

☐	ID	Accession	St…	D…	Original ID	Alternative…	Date	N…	Experime…	Platform	Scientific …	Type	Description
☐	72…	GSE11906		G…	200011906		Mon Sep 2…	223	Quality Co…	GPL570	19852842	Expressio…	Microarray…
☐	72…	GSE30373		G…	200030373		Fri Jan 01 …	48	Global gen…	GPL10439…		Expressio…	This Super…

Search options

Term: term

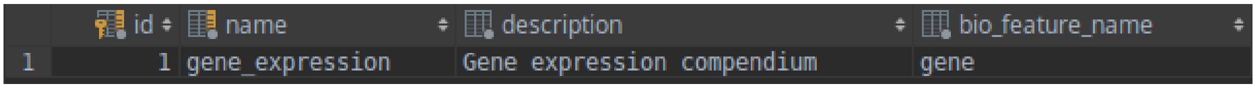
Database:



id	source_name	python_class	is_local
4	GEO	command.lib.coll.public_database.GEOPublicDatabase	☐
5	ArrayExpress	command.lib.coll.public_database.ArrayExpressPublicDatabase	☐
6	local	command.lib.coll.local_data_source.LocalDataSource	☑
7	SRA	command.lib.coll.public_database.SRAPublicDatabase	☐

10.3 Add new compendium type

This is by far the easiest thing to do since it's just a matter of adding one tuple on the `command` DB. The table to be modified is the `command_compendiumtype` table. At the moment the only compendium type defined is the gene expression one. The fields are *name*, *description* and the *biological feature name*, so respectively *gene_expression*, *Gene expression compendium* and *gene*.



	id	name	description	bio_feature_name
1	1	gene_expression	Gene expression compendium	gene

10.4 Add new biological feature file importer

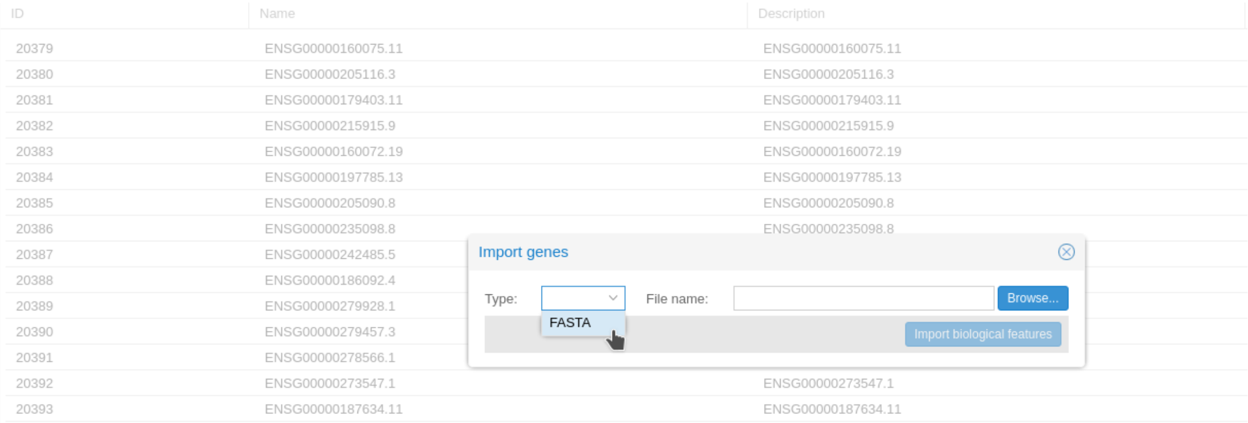
All the classes releated to importing *biological features* are located here. First thing to do is to inform the dispatcher in the `importers.py` file which are the classes responsible to manage different file types. For example, genes will be imported using FASTA files. The second step is to actually implement the class extending the `BaseImporter` class. The newly defined class will need to implement the `parse` method and redefine the `FILE_TYPE_NAME` variable.

```
# fasta_file_importer.py

class FastaFileImporter(BaseImporter):
    FILE_TYPE_NAME = 'FASTA'

    def parse(self, filename):
        sequence_field = BioFeatureFields.objects.using(self.compendium).get(name=
↪'sequence')
```

(continues on next page)

(continued from previous page)

```
        with transaction.atomic(using=self.compendium):
            with open(filename, 'rU') as handle:
                for record in SeqIO.parse(handle, 'fasta'):
                    gene = BioFeature()
                    gene.name = record.id
                    gene.description = record.description
                    gene.save(using=self.compendium)
                    bf_value = BioFeatureValues()
                    bf_value.bio_feature = gene
                    bf_value.bio_feature_field = sequence_field
                    bf_value.value = str(record.seq)
                    bf_value.save(using=self.compendium)
```

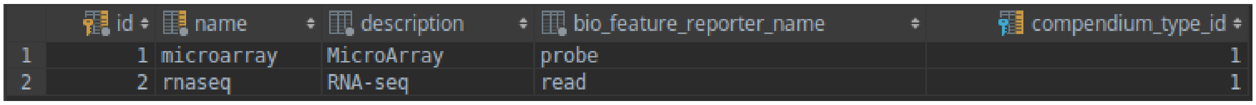


ID	Name	Description
20379	ENSG00000160075.11	ENSG00000160075.11
20380	ENSG00000205116.3	ENSG00000205116.3
20381	ENSG00000179403.11	ENSG00000179403.11
20382	ENSG00000215915.9	ENSG00000215915.9
20383	ENSG00000160072.19	ENSG00000160072.19
20384	ENSG00000197785.13	ENSG00000197785.13
20385	ENSG00000205090.8	ENSG00000205090.8
20386	ENSG00000235098.8	ENSG00000235098.8
20387	ENSG00000242485.5	
20388	ENSG00000186092.4	
20389	ENSG00000279928.1	
20390	ENSG00000279457.3	
20391	ENSG00000278566.1	
20392	ENSG00000273547.1	ENSG00000273547.1
20393	ENSG00000187634.11	ENSG00000187634.11

10.5 Add new platform type

To add a new platform type there are several step to do and mostly depends on the kind of platform is going to be added.

Database entry To add a new platform type for a single compendium (organism) you will need to add a tuple with name, description, bio feature reporter name and the compendium type ID, for example: *microarray*, *MicroArray*, *probe* and *1* to the `command_platformtype` table. If you want *every* new compendium you are going to create to have such new platform you will need to add the same tuple to the `command_platformtypeadmin` table in the `command` DB.



	id	name	description	bio_feature_reporter_name	compendium_type_id
1	1	microarray	MicroArray	probe	1
2	2	rnaseq	RNA-seq	read	1

Reporters ExtJS GUI Next step will be to inform the GUI how to behave when the user wants to see the *biological feature reporters* associated with the new platform. For example in case of Microarray the *biological feature reporters* are the probes. The file to modify is `PlatformController.js` (defined here). `onViewBioFeatureReporter` is the method to modify adding a new case for the new platform. For example in case of RNA-seq we simply display a message to say there's no associated *biological feature reporters* since the gene expression measurement in this case is directly given by read counts. For Microarray instead we have probes and thus we will open a new window to show the probes associated with this platform, the `window_bio_feature_reporter` window.

```
// PlatformsController.js

onViewBioFeatureReporter: function (me) {
    var selection = me.up('grid').getSelectionModel().getSelection()[0].data;
    var comp = JSON.parse(localStorage.getItem("current_compendium"));
    if (selection.platform_type) {
        switch (selection.platform_type.name) {
            case 'rnaseq':
                Ext.MessageBox.show({
                    title: 'RNA-seq platform',
                    msg: 'For RNA-seq platform ' + selection.platform_access_id +  ',
↪' +  comp.compendium_type.bio_feature_name + ' is/are directly measured',
                    buttons: Ext.MessageBox.OK,
                    icon: Ext.MessageBox.INFO,
                    fn: function (a) {
                    }
                });
                break
            case 'microarray':
                var win = Ext.create({
                    xtype: 'window_bio_feature_reporter',
                    title: 'Microarray platform ' + selection.platform_access_id + ':
↪' +
                        comp.compendium_type.bio_feature_name + ' feature reporters (
↪' + selection.platform_type.bio_feature_reporter_name + ')',
                    platform: selection
                });
                break
        }
    }
```

10.6 Add new platform mapper

When a platform has *biological feature reporters* associated, these must be mapped to the *biological features*. In case of *gene expression* compendium the *biological features* are genes. So to give a concrete example we will need to associate Microarray probes to genes. This step is very platform-dependant and so a lot of freedom is left to the developer to design the GUI. There are just few things to keep in mind in order to have everything working correctly within the COMMAND>_ framework.

Mapper ExtJS GUI First thing will be to inform the GUI how to behave when the user wants to map this platform reporters to the *biological features*. The file to modify is again the `PlatformController.js` (defined here), but this time we are going to modify the `onMapPlatformToBioFeature` method, adding a new case for the new platform. For Microarray we defined a new window `window_map_microarray_platform` here. Again, in this case the developer is left completely free to design it as he wants.

```
// PlatformsController.js

onMapPlatformToBioFeature: function (me) {
    var selection = me.up('grid').getSelectionModel().getSelection()[0].data;
    var comp = JSON.parse(localStorage.getItem("current_compendium"));
    if (selection.platform_type) {
        switch (selection.platform_type.name) {
            case 'rnaseq':
                Ext.MessageBox.show({
```

(continues on next page)

(continued from previous page)

```
                    title: 'RNA-seq platform',
                    msg: 'RNA-seq platform ' + selection.platform_access_id +  ' is␣
↪automatically mapped to ' + comp.compendium_type.bio_feature_name,
                    buttons: Ext.MessageBox.OK,
                    icon: Ext.MessageBox.INFO,
                    fn: function (a) {
                    }
                });
                break
            case 'microarray':
                command.current.createWin({
                    xtype: 'window_map_microarray_platform',
                    title: 'Map microarray platform ' + selection.platform_access_id␣
↪+ ' to ' + comp.compendium_type.bio_feature_name,
                    platform: selection
                });
                break
        }
    }
```

Mapper Django View The associated Django View is defined in `platform.py` view file here and for Microarray this is the `MicroarrayPlatformView` class. This is pretty standard view as described previously.

Mapper code The actual code is stored in a class that will extend the `BaseMapper` (placeholder) class. For Microarray this class is `MicroarrayMapper` and is located here. Last step is to inform the mapper dispatcher on which class to invoke, and this is done in the `mappers.py` file located here.

```
// mappers.py

from command.lib.coll.platform.microarray_mapper import MicroarrayMapper

platform_mapper = {
    'microarray': MicroarrayMapper
}
```

CHAPTER 11

Contribute & Support

Use the GitHub Push Request and/or Issue Tracker.

CHAPTER 12

Author

To send me an e-mail about anything else related to COMMAND>_ write to marco.moretto@fmach.it

CHAPTER 13

License

The project is licensed under the GPLv3 license.

CHAPTER 14

How to cite

If you find COMMAND>_ useful for your work please cite

Moretto, M., Sonego, P., Villaseñor-Altamirano, A. B., & Engelen, K. (2019). **First step toward gene expression data integration: transcriptomic data acquisition with COMMAND>_.** *BMC bioinformatics*, 20(1), 54. ISO 690

https://doi.org/10.1186/s12859-019-2643-6

Python Module Index

Index