GenomeDepot: microbial genomic data management system

GenomeDepot documentation

GenomeDepot is an open-source web-based platform for annotation, management, and comparative analysis of microbial genomic data. With GenomeDepot, you can create web-sites for your own genome collections. These web-sites have tools for interactive genome browsing, BLAST search, annotation search, comparative genomic neighborhood visualization, and sequence download.

Contents

Introduction

The data management system for comparative genomics (GenomeDepot) is an open-source web-based platform for annotation, management and comparative analysis of microbial genomic sequences and associated data including ortholog families, protein domains, operons, regulatory interactions, metagenomic samples, strains taxonomy and metadata.

GenomeDepot is a tool developed to create web portals for microbial genome collections each containing hundreds and thousands of genomes. The web portals are built on the Django framework and backed by a MySQL database that aggregates gene annotations generated by various bioinformatic tools. The genome annotation tools are installed in separate Conda environments and run by the GenomeDepot annotation pipeline. GenomeDepot employs Django Q, a multiprocessing task queue, for scheduling and executing pipeline jobs. In addition to the pipeline-generated data, administrators of a GenomeDepot-based portal can import gene annotations from text files or enter them manually in the site administration panel.

Demo GenomeDepot-based genome collection portal

Image Credits

Background image: *B. burgdorferi* bacteria. Photo by Jamice Haney Carr, Claudia Molins, USCDCP on Pixnio

Main page image: *Streptococcus pneumoniae* bacterial colonies that were grown on primary isolation medium. Photo by Dr. Richard Facklam, USCDCP on Pixnio

Earth spinning rotating animation. Amirabbaszakavi, CC BY-SA 4.0, via Wikimedia Commons

Software credits

GenomeDepot employs a wide variety of genome data transformation and analysis tools:

eggNOG-mapper https://github.com/eggnogdb/eggnog-mapper

License: GNU GPL v3.0

References: [1] eggNOG-mapper v2: functional annotation, orthology assignments, and domain prediction at the metagenomic scale. Carlos P. Cantalapiedra, Ana Hernandez-Plaza, Ivica Letunic, Peer Bork, Jaime Huerta-Cepas. 2021. Molecular Biology and Evolution, msab293, https://doi.org/10.1093/molbev/msab293

[2] eggNOG 5.0: a hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses. Jaime Huerta-Cepas, Damian Szklarczyk, Davide Heller, Ana Hernández-Plaza, Sofia K Forslund, Helen Cook, Daniel R Mende, Ivica Letunic, Thomas Rattei, Lars J Jensen, Christian von Mering, Peer Bork Nucleic Acids Res. 2019 Jan 8; 47(Database issue): D309–D314. doi: 10.1093/nar/gky1085

AMRFinderPlus https://github.com/ncbi/amr

License: Public Domain

Reference: Feldgarden M, Brover V, Gonzalez-Escalona N, Frye JG, Haendiges J, Haft DH, Hoffmann M, Pettengill JB, Prasad AB, Tillman GE, Tyson GH, Klimke W. AMRFinderPlus and the Reference Gene Catalog facilitate examination of the genomic links among antimicrobial resistance, stress response, and virulence. Sci Rep. 2021 Jun 16;11(1):12728. doi: 10.1038/s41598-021-91456-0. PMID: 34135355; PMCID: PMC8208984. https://pubmed.gov/31427293

antiSMASH https://github.com/antismash/antismash

License: GNU GPL v3.0

Reference: antiSMASH 6.0: improving cluster detection and comparison capabilities Kai Blin, Simon Shaw, Alexander M Kloosterman, Zach Charlop-Powers, Gilles P van Weezel, Marnix H Medema, & Tilmann Weber Nucleic Acids Research (2021) doi: 10.1093/nar/gkab335.

PhiSpy https://github.com/linsalrob/PhiSpy

License: MIT License

Reference: Sajia Akhter, Ramy K. Aziz, Robert A. Edwards; PhiSpy: a novel algorithm for finding prophages in bacterial genomes that combines similarity- and composition-based strategies. Nucl Acids Res 2012; 40 (16): e126. doi: 10.1093/nar/gks406

eCIS-screen https://github.com/ipb-jianyang/eCIS-screen

License: GPL-3.0 license

Reference: Chen L, Song N, Liu B, Zhang N, Alikhan NF, Zhou Z, Zhou Y, Zhou S, Zheng D, Chen M, Hapeshi A, Healey J, Waterfield NR, Yang J, Yang G. Genome-wide Identification and Characterization of a Superfamily of Bacterial Extracellular Contractile Injection Systems. Cell Rep. 2019 Oct 8;29(2):511-521.e2. doi: 10.1016/j.celrep.2019.08.096. PMID: 31597107; PMCID: PMC6899500.

Fama https://github.com/aekazakov/Fama

License: LBNL BSD 3-clause

Reference: Kazakov A, Novichkov P. Fama: a computational tool for comparative analysis of shotgun metagenomic data. Great Lakes Bioinformatics conference (poster presentation). 2019. , https://iseq.lbl.gov/mydocs/fama_glbio2019_poster.pdf

GapMind https://github.com/morgannprice/PaperBLAST

License: GPL-3.0 license

Reference: Price MN, Deutschbauer AM, Arkin AP. GapMind: Automated Annotation of Amino Acid Biosynthesis. mSystems. 2020 Jun 23;5(3):e00291-20. doi: 10.1128/mSystems.00291-20. PMID: 32576650; PMCID: PMC7311316.

DefenseFinder https://github.com/mdmparis/defense-finder

License: GPL-3.0 license

Reference: “Systematic and quantitative view of the antiviral arsenal of prokaryotes” Nature Communication, 2022, Tesson F., Hervé A. , Mordret E., Touchon M., d’Humières C., Cury J., Bernheim A.

MacSyFinder https://github.com/gem-pasteur/macsyfinder

License: GPL-3.0 license

Reference: “MacSyFinder: A Program to Mine Genomes for Molecular Systems with an Application to CRISPR-Cas Systems.” PloS one 2014 Abby S., Néron B.,Ménager H., Touchon M. Rocha EPC.

geNomad https://github.com/apcamargo/genomad

License: ACADEMIC, INTERNAL, RESEARCH & DEVELOPMENT, NON-COMMERCIAL USE ONLY, LICENSE

Reference: Camargo, A.P., Roux, S., Schulz, F. et al. Identification of mobile genetic elements with geNomad. Nat Biotechnol 42, 1303–1312 (2024). https://doi.org/10.1038/s41587-023-01953-y

HMMER https://github.com/EddyRivasLab/hmmer

License: BSD 3-clause

Reference: S. R. Eddy. Accelerated profile HMM searches. PLOS Comp. Biol., 7:e1002195, 2011

Muscle https://github.com/rcedgar/muscle

License: GPL-3.0 license

Reference: Edgar RC., Muscle5: High-accuracy alignment ensembles enable unbiased assessments of sequence homology and phylogeny. Nature Communications 13.1 (2022): 6968. https://www.nature.com/articles/s41467-022-34630-w.pdf

NCBI BLAST+ https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST

License: Public Domain

Reference: Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, Madden TL. BLAST+: architecture and applications. BMC Bioinformatics. 2009 Dec 15;10:421. doi: 10.1186/1471-2105-10-421. PMID: 20003500; PMCID: PMC2803857.

Django https://github.com/django/django

License: BSD 3-Clause

Reference: Django Software Foundation, 2019. Django, Available at: https://djangoproject.com.

parasail https://github.com/jeffdaily/parasail

License: Battelle BSD-style

Reference: Daily, Jeff. (2016). Parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments. BMC Bioinformatics, 17(1), 1-11. doi:10.1186/s12859-016-0930-z

Biopython https://doi.org/10.1093/bioinformatics/btp163

License: BSD 3-Clause

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

Jbrowse v1 https://jbrowse.org/jbrowse1.html

License: GNU LGPL

Reference: Buels R et al. JBrowse: a dynamic web platform for genome visualization and analysis.Genome Biology (2016).

samtools https://github.com/samtools/samtools

License: The MIT/Expat License

Reference: Twelve years of SAMtools and BCFtools Petr Danecek, James K Bonfield, Jennifer Liddle, John Marshall, Valeriu Ohan, Martin O Pollard, Andrew Whitwham, Thomas Keane, Shane A McCarthy, Robert M Davies, Heng Li GigaScience, Volume 10, Issue 2, February 2021, giab008, https://doi.org/10.1093/gigascience/giab008

tabix https://doi.org/10.1093/gigascience/giab007

License: The MIT/Expat License

Reference: HTSlib: C library for reading/writing high-throughput sequencing data James K Bonfield, John Marshall, Petr Danecek, Heng Li, Valeriu Ohan, Andrew Whitwham, Thomas Keane, Robert M Davies GigaScience, Volume 10, Issue 2, February 2021, giab007,

POEM https://github.com/Rinoahu/POEM_py3k

License: GPL-3.0 license

Reference: Identifying Core Operons in Metagenomic Data Xiao Hu, Iddo Friedberg bioRxiv 2019.12.20.885269; doi: https://doi.org/10.1101/2019.12.20.885269

Xiao R. (2019). POEM py3k: GitHub repository. Available online at: https://github.com/Rinoahu/POEM_py3k (accessed December 16, 2020).

Scribl https://github.com/chmille4/Scribl

License: MIT License

Reference: Miller CA, Anthony J, Meyer MM, Marth G. Scribl: an HTML5 Canvas-based graphics library for visualizing genomic data over the web. Bioinformatics. 2013 Feb 1;29(3):381-3. doi: 10.1093/bioinformatics/bts677. Epub 2012 Nov 19. PMID: 23172864; PMCID: PMC3562066.

Installation

This document uses the /opt/genomedepot directory as an example, but GenomeDepot can be installed in another directory as well. **Installation in the /opt/genomedepot directory requires sudo rights.**

Installing GenomeDepot in the user's home directory **is not recommended** because of possible issues with file access permissions and web app execution by the Apache web server.

In this document, https://example.com/mygenomes is used as a base URL for the GenomeDepot-based web portal. For your installation, use your own domain name instead of example.com.

Prerequisites

- Linux-based OS (GenomeDepot was developed and tested in 64-bit Ubuntu Linux system)
- Python 3.8+
- conda (miniconda is recommended)
- MySQL server
- Apache2 web-server with mod_wsgi and ssl extensions

- Muscle
- HMMER
- NCBI-BLAST (blastp and megablast required)
- Perl
- zlib1g-dev package
- curl
- git
- pkg-config

In Ubuntu-based distibutions, you can install most prerequisites using the APT package manager:

```
sudo apt install apache2 mysql-server muscle hmmer ncbi-blast+-legacy build-
essential zlib1g-dev libexpat1-dev python3-dev libmysqlclient-dev curl git pkg-
config libapache2-mod-wsgi-py3 python3.10-venv
```

If Conda is not installed in the system, install Miniconda as described in the Conda user guide.

Install dependencies

Create a directory where GenomeDepot and external tools will be installed.

```
cd /opt
sudo mkdir genomedepot
```

If you created the genomedepot directory with sudo, change the directory ownership to a non-root user (replace USERNAME with a real user name):

```
sudo chown USERNAME:www-data genomedepot
```

Create apps subdirectory. In this directory, GenomeDepot-based portals will be installed.

```
cd genomedepot
mkdir apps
cd apps
```

Create a directory for a new GenomeDepot portal in genomedepot/apps (for example, mygenomes) and clone the repository into it.

```
mkdir mygenomes
cd mygenomes
git clone https://github.com/aekazakov/genome-depot
```

Run the install.sh script, which installs external tools and creates a Python virtual environment for GenomeDepot. Running the script may take quite some time for the deployment of the first GenomeDepot-based portal because it will create all the conda environments and download reference data for the tools.

```
cd genome-depot
bash install.sh
```

The install.sh script installs all required Python libraries including Django framework, genome annotation tools and other dependencies. If it fails, check the error message, fix the problem and start install.sh again. The most common problem is a name conflict with existing conda environments. In this case, the installation script does not overwrite the existing environment but stops the installation. User can rename or delete the existing environment before restarting the installation.

Another common problem is an incomplete installation of the operon prediction tool POEM. If POEM fails to predict operons, and running poem.sh script throws an error "AttributeError: module 'tensorflow' has no attribute 'get_default_graph'", it means the versions of keras and tensorflow are not compatible. Activate conda genomedepot-poem environment and run `pip install tensorflow==1.13.1`.

Initial GenomeDepot configuration

Create MySQL user (for example, gduser), if needed, or use an existing mysql account.

Create MySQL database (for example, gdgenomes): log into mysql as root:

```
mysql -u root -p
```

If MySQL returns error "Access denied for user 'root'@'localhost'", use sudo before the command:

```
mysql -u root -p
```

Enter commands that create the database and grant the user access to the database:

```
CREATE DATABASE gdgenomes CHARACTER SET utf8;
GRANT ALL PRIVILEGES ON gdgenomes.* TO 'gduser'@'localhost';
quit
```

Copy the genomedepot/apps/mygenomes/genome-depot/genomebrowser/.env.template file to genomedepot/apps/mygenomes/genome-depot/genomebrowser/.env. Open genomedepot/apps/mygenomes/genome-depot/genomebrowser/.env in a text editor and enter the settings:

- SECRET_KEY: Django secret key
- ALLOWED_HOSTS: comma-separated list of host names and IP addresses for the web server (for example, example.com,127.0.0.1,testserver)
- INTERNAL_IPS: comma-separated list of IP addresses for this site (usually, 127.0.0.1,)
- DB_USER: mysql user name (created at step 1)
- DB_PASSWORD: mysql password (created at step 1)
- DB_NAME: mysql database name (created at step 2)
- STATIC_URL: URL for static files directory (for example, https://example.com/gdstatic/mygenomes/)
- TITLE: web site title
- BASE_URL: URL of the web site. It can be a domain name (https://example.com/) or a subdirectory (https://example.com/mygenomes/)
- ADMIN: admin name and email separated by comma (John Doe,johndoe@example.com)
- EMAIL_HOST_USER: an email address for GenomeDepot to send messages from (mygenomes@example.com)
- EMAIL_HOST_PASSWORD: a password for the EMAIL_HOST_USER email
- EMAIL_HOST: external e-mail server address
- STATIC_ROOT: full path of a directory for static files from which the web server serves up STATIC_URL. For example, /opt/genomedepot/static/mygenomes

- STATICFILES_DIR: a directory with static files in the GenomeDepot installation. For example, /opt/genomedepot/apps/mygenomes/genome-depot/genomebrowser/static
- LOGVIEWER_LOGS: full path to the django.log file in the genomebrowser directory, followed by comma. For example, /opt/genomedepot/apps/mygenomes/genome-depot/genomebrowser/django.log,

Open /opt/genomedepot/apps/mygenomes/genome-depot/genomebrowser/configs.txt in a text editor. This file is generated by the GenomeDepot installation script and contains GenomeDepot configuration parameters to be saved in the database. Check the parameters for correctness.

Activate virtual environment genomedepot-venv, change directory to /opt/genomedepot/app/mygenomes/genome-depot/genomebrowser and run Django configuration commands

```
source /opt/genomedepot/genomedepot-venv/bin/activate
cd /opt/genomedepot/app/mygenomes/genome-depot/genomebrowser
python manage.py collectstatic
python manage.py makemigrations
python manage.py migrate
python manage.py createsuperuser
# Enter your desired username, email and password
python manage.py import_config -i configs.txt
python manage.py createcachetable
python manage.py update_taxonomy
```

Now your new GenomeDepot portal is ready for testing. Run

```
python manage.py runserver 127.0.0.1:8000
```

and open in the browser http://127.0.0.1:8000/admin. You should be able to log in with the superuser username and password you just created.

If test server cannot find static files, check if www-data user can read from the /opt/genomedepot/static/mygenomes directory (giving rw permissions for www-data group would work).

How to configure the Apache web server for GenomeDepot

You need sudo privileges for making changes in Apache configuration files. Open apache2 site configuration file (it may be /etc/apache2/sites_available/default_ssl.conf) in a text editor and add the following (change file paths, if needed):

```
WSGIDaemonProcess genomedepotpy python-home=/opt/genomedepot/genomedepot-venv
python-path=/opt/genomedepot/app/mygenomes/genome-depot/genomebrowser
WSGIScriptAlias /mygenomes /opt/genomedepot/app/mygenomes/genome-
depot/genomebrowser/genomebrowser/wsgi.py process-group=genomedepotpy application-
group=%{GLOBAL}
<Directory /opt/genomedepot/app/mygenomes/genome-
depot/genomebrowser/genomebrowser/>
<Files wsgi.py>

    Require all granted

</Files>
</Directory>
```

```
<Directory /opt/genomedepot/static/>
Options -Indexes +FollowSymLinks

<IfModule mod_headers.c>

  Header set Access-Control-Allow-Origin http://127.0.0.1:8000

</IfModule>

Require all granted
Alias /gdstatic /opt/genomedepot/static/
</Directory>
```

You may have to add “Header always set X-Frame-Options “SAMEORIGIN”” to web server configuration if the embedded genome viewer is not properly displayed.

Change group ownership to www-data for the genome-depot/genomebrowser/django.log file. For example:

```
sudo chown :www-data /opt/genomedepot/apps/mygenomes/genome-
depot/genomebrowser/django.log
```

Restart apache2:

```
sudo systemctl restart apache2
```

Now you would be able to open https://your.domain.name/mygenomes in a web browser.

Start Django Q cluster from the command line

A Django Q cluster must be started for the execution of GenomeDepot pipeline jobs from the task queue. To start the cluster, start a new screen session with “screen” command and enter:

```
conda deactivate
source /<GenomeDepot venv path>/bin/activate
cd /<GenomeDepot path>/genomebrowser
python manage.py qcluster
```

Optionally, the screen session can be named. Press Ctrl-A, then type “:sessionname “ and type a name.

After that, detach the screen session (press Ctrl-A, then d).

If the cluster is up and running, the number of clusters in the administration panel changes from 0 to 1. There is no need to run more than one cluster for a GenomeDepot-based portal, since only one task can be processed at any moment. The GenomeDepot pipeline does not let parallel processing of tasks to prevent concurrent modification of the data.

To stop the Django Q cluster, attach the screen session with “screen -r” command, and press Ctrl-C.

Start Django Q cluster from crontab

The Django Q cluster can be started on server restart as a cron job. First, create a bash script that starts the cluster:

```
#!/usr/bin/bash
conda deactivate
source /<GenomeDepot venv path>/bin/activate
cd /<GenomeDepot path>/genomebrowser
python manage.py qcluster
```

Then, open cron file (crontab -e) and add a line to the end:

```
@reboot(. ~/.profile /usr/bin/screen -dmS gdcluster <path to the shell script>)
```

Save the file and exit the editor. To stop a cluster, run “screen -r gdcluster” command and press Ctrl-C. Warning: if the cluster is running a task, it may not stop immediately, and some data may be lost or corrupted. If you have more than one GenomeDepot-based portal, change “gdcluster” to a unique name for each screen session.

How to change the background image of your web-site

There are two page background images in the repository, one for the dark mode (genomebrowser/static/images/background.jpg) and the other for the light mode (genomebrowser/static/images/background_light.jpg). The background image on the start page is genomebrowser/static/images/slide02.jpg. You can replace any of these files with your own images, then run `python manage.py collectstatic` command and restart the Apache web server.

To keep git from overwriting image files during future updates, run:

```
git update-index --skip-worktree genomebrowser/static/images/slide02.jpg
git update-index --skip-worktree genomebrowser/static/images/background.jpg
git update-index --skip-worktree genomebrowser/static/images/background_light.jpg
```

How to change links at the bottom of web pages

Edit the genomebrowser/browser/templates/footer_links.html file. To replace icons at the links, you can choose from Font Awesome 4.0.3 Icons set.

To keep git from overwriting the file during future updates, run:

```
git update-index --skip-worktree genomebrowser/browser/templates/footer_links.html
```

Tweak MySQL configuration

As the genome database grows, MySQL performance may decrease because of excessive I/O usage. To increase MySQL performance, change innodb_buffer_pool_size parameter in MySQL configuration file. Run the following query in the mysql window to find optimal innodb_buffer_pool_size:

```
SELECT CEILING(Total_InnoDB_Bytes*1.6/POWER(1024,3)) RIBPS FROM (SELECT
SUM(data_length+index_length) Total_InnoDB_Bytes FROM information_schema.tables
WHERE engine='InnoDB') A;
```

It will show RIBPS value in gigabytes. If your server has enough memory, enter that number into the mysqld section of mysqld.conf (/etc/mysql/mysql.conf.d/mysqld.conf) and add “G”, for example:

```
[mysqld]
innodb_buffer_pool_size=8G
```

After making the changes, restart MySQL:

```
sudo systemctl restart mysql
```

User guide

Concepts, general description

Genome is a complete or partial assembly of genomic sequences of a particular organism. Genome can include one or several contiguous nucleotide sequences, or contigs. Genomes can represent cultured microbial isolates or uncultured organisms from metagenomic samples.

Strain in GenomeDepot is an organism cultivated under controlled laboratory conditions, typically in the absence of other species. There can be more than one genome of a strain, if several versions of a genome assembly are available.

Sample is a metagenomic data set representing multiple members of a microbial community. Metagenome-assembled genomes (MAGs) are linked to samples.

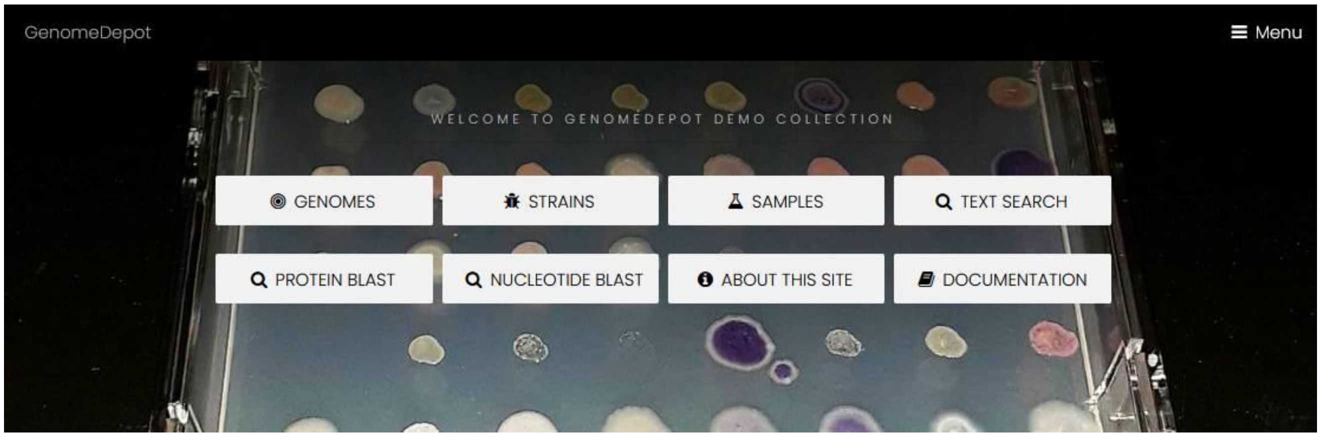
Gene in GenomeDepot is a continuous DNA segment, typically with assigned function. Genes can be coding or non-coding. Coding genes are linked to proteins. Non-coding genes are usually either rRNA and tRNA genes or pseudogenes.

Regulon is a group of genes (or operons) that are directly regulated by a DNA-binding transcription factor, which recognizes binding sites upstream of those genes.

Start page

The start page is a main entry point for a GenomeDepot-based web site. The start page contains eight links:

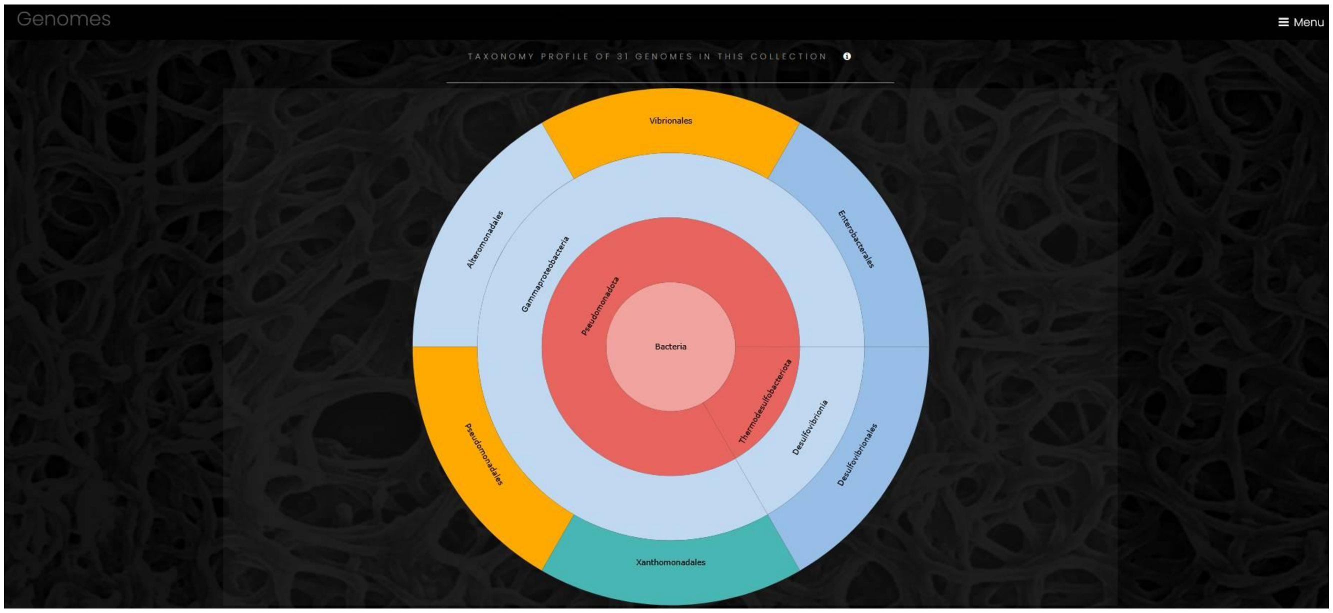
- List of genomes
- List of strains
- List of samples
- Text search
- Nucleotide sequence search
- Protein sequence search
- About page with site statistics
- Link to GenomeDepot documentation site



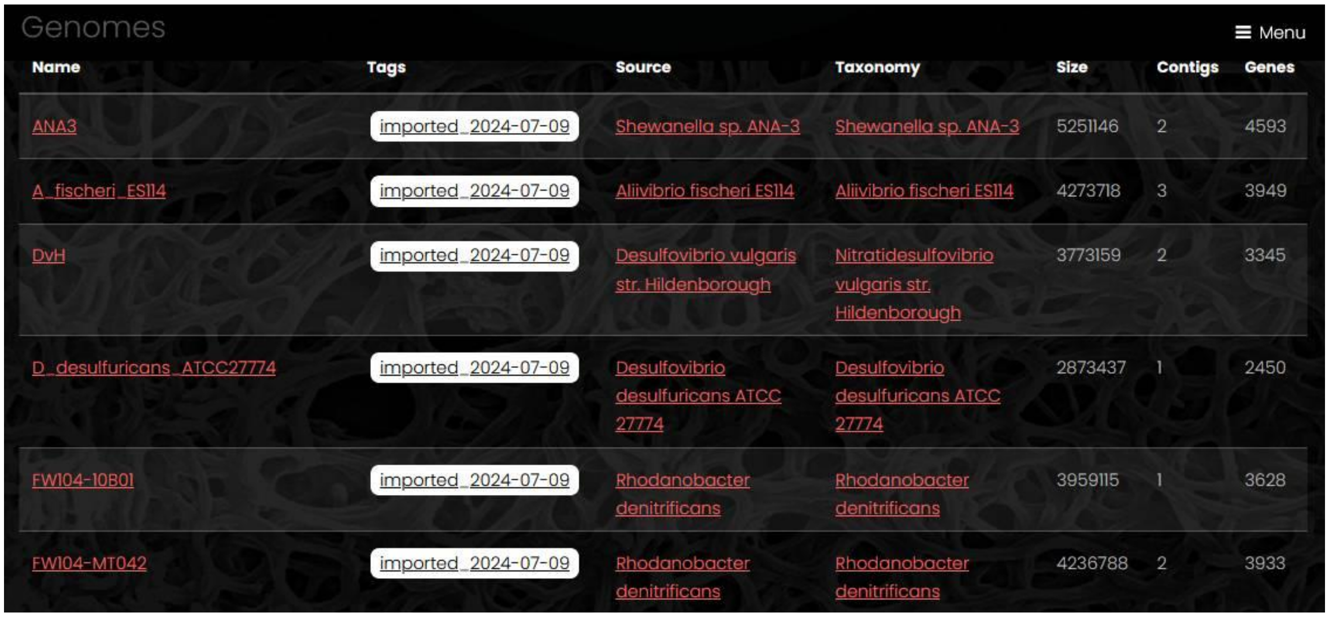
The menu button in the top right corner of the start page opens a menu with the same eight links, and also a link switching between dark and bright modes. You can find the menu button on every page of the site.

Genome list

The page with a list of genomes has a search field, a sunburst chart of genomes taxonomy, and a table of genomes. The search field helps site visitors to find a genome by name or by metadata. The search is case-insensitive.



The sunburst chart visualizes taxonomy data spanning outwards radially from the root node. It shows four levels of taxonomy, and a click on a sector outside of the taxon label puts this taxon to the center of the chart and opens additional children taxa, if any. A click on the central circle makes one step up in taxonomical hierarchy. A click on a taxon’s label opens a page of the taxon. Outermost sectors display genomes, and a click on the label of a genome sector opens a page of the genome in a new browser tab.

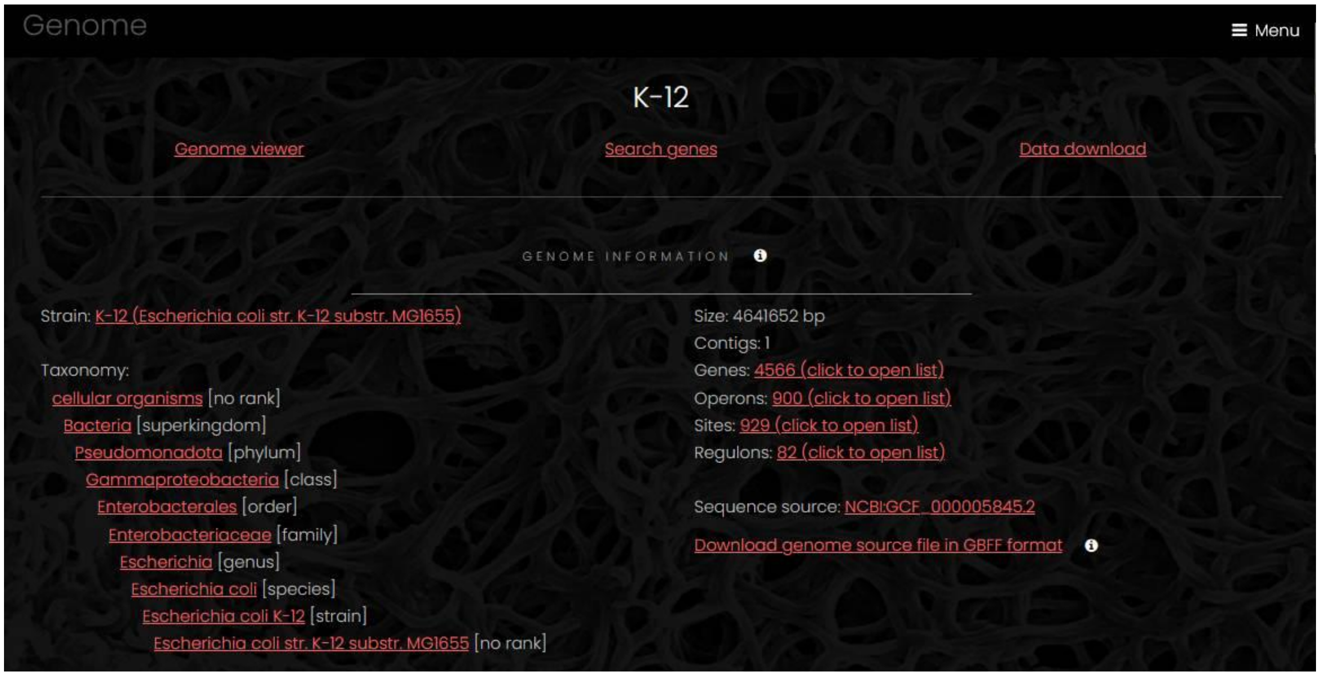

Genomes

Name	Tags	Source	Taxonomy	Size	Contigs	Genes
ANA3	imported_2024-07-09	Shewanella sp. ANA-3	Shewanella sp. ANA-3	5251146	2	4593
A_fischeri_ES114	imported_2024-07-09	Aliivibrio fischeri ES114	Aliivibrio fischeri ES114	4273718	3	3949
DvH	imported_2024-07-09	Desulfovibrio vulgaris str. Hildenborough	Nitratidesulfovibrio vulgaris str. Hildenborough	3773159	2	3345
D_desulfuricans_ATCC27774	imported_2024-07-09	Desulfovibrio desulfuricans ATCC 27774	Desulfovibrio desulfuricans ATCC 27774	2873437	1	2450
FW104-10B01	imported_2024-07-09	Rhodanobacter denitrificans	Rhodanobacter denitrificans	3959115	1	3628
FW104-MT042	imported_2024-07-09	Rhodanobacter denitrificans	Rhodanobacter denitrificans	4236788	2	3933

The list of genomes displays genome name, genome tags, strain name, taxon name, genome size, number of contigs and number of genes. A link under the table exports full genome list with additional fileds as a tab-separated file.

Genome

Genome pages have four parts: genome information, interactive genome viewer, search fields and data download. The information section has links to the strain page, list of genes, list of operons, hierarchical list of taxa with links to each taxon page and a link to the source sequence file.



K-12

Genome viewer | Search genes | Data download

GENOME INFORMATION

Strain: K-12 (Escherichia coli str. K-12 substr. MG1655)

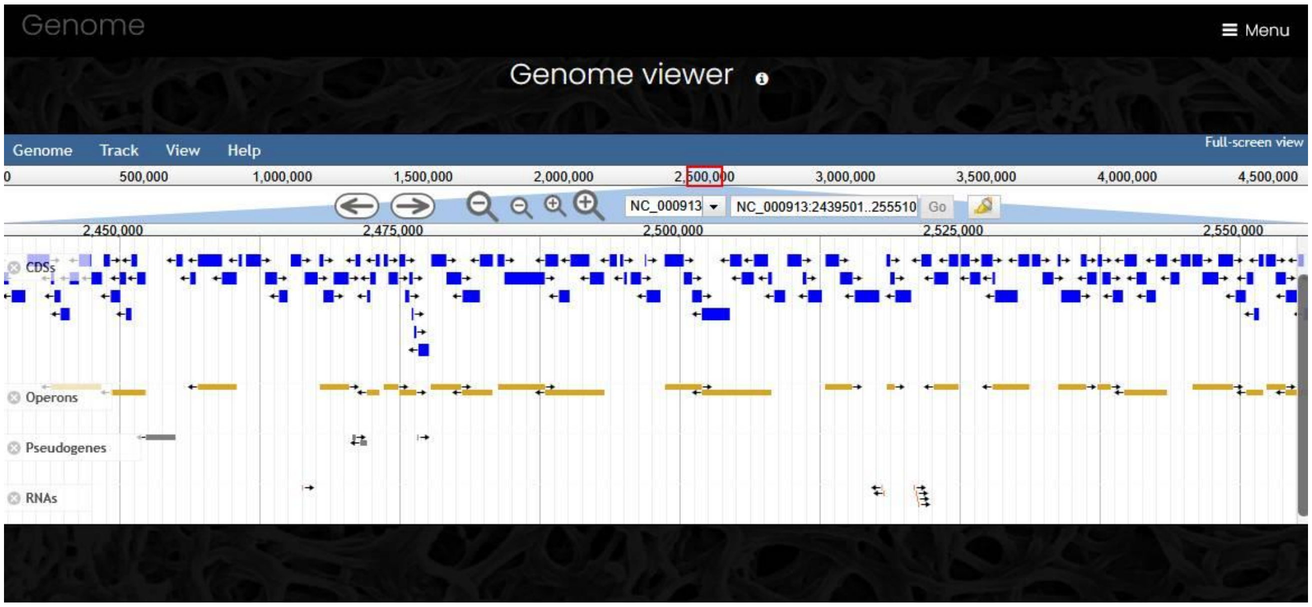
Taxonomy:
- cellular organisms [no rank]
  - Bacteria [superkingdom]
    - Pseudomonadota [phylum]
      - Gammaproteobacteria [class]
        - Enterobacterales [order]
          - Enterobacteriaceae [family]
            - Escherichia [genus]
              - Escherichia coli [species]
                - Escherichia coli K-12 [strain]
                  - Escherichia coli str. K-12 substr. MG1655 [no rank]

Size: 4641652 bp
Contigs: 1
Genes: 4566 (click to open list)
Operons: 900 (click to open list)
Sites: 929 (click to open list)
Regulons: 82 (click to open list)

Sequence source: NCBI:GCF_000005845.2

Download genome source file in GBFF format

The genome viewer section shows embedded Jbrowse genome browser with reference sequence track and four feature tracks (CDSs, operons, pseudogenes and RNA genes). A click on a feature opens feature details widget that has a link to the feature page in the Name field. In the genome viewer, visitors can scroll a contig side to side using your mouse wheel or via click and drag. The zoom buttons and the slider bar found in the header of the linear genome view can be used to zoom in and out on the view. Visitors can switch between contigs using dropdown menu.



In the search section, site visitors can look for genes in the genome by name, locus tag, functional annotation, or use one of functional classifications to filter the gene list.

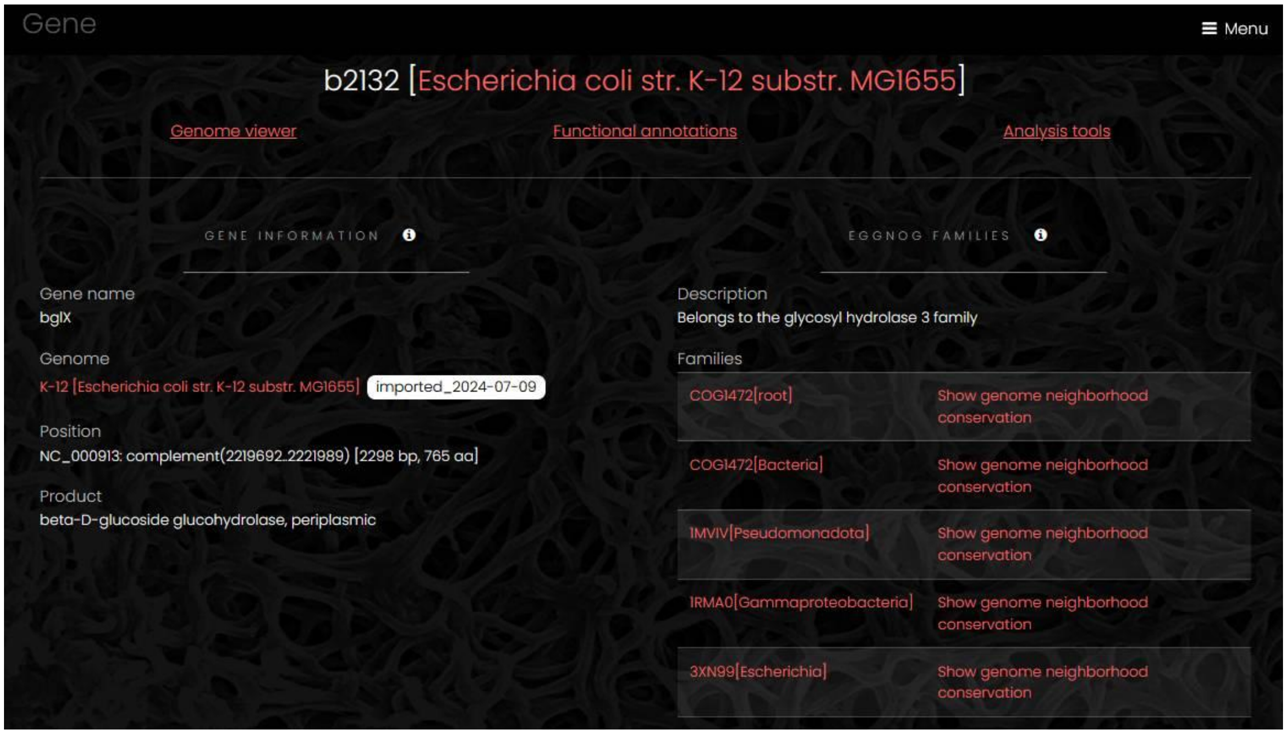
Three buttons in the download section generate a GenBank format file with all annotations, export a table of genes as tab-separated file, or export all protein sequences in FASTA format.

Gene

Gene pages have four sections: gene information, genome viewer, functional annotations and analysis tools. The gene information section contains links to eggNOG families at different taxonomic levels, links to conserved genome neighborhood visualizations and general information about the gene.

The genome viewer section is similar to the viewer section of the genome page but with the current gene highlighted.

The functional annotations section shows annotations generated by eggNOG-mapper in the left column and annotations generated by other tools in the right column. The former includes KEGG pathways, reactions, and orthologs, Gene Ontology labels, EC numbers, TCDB families, CAZy terms and COG functional classes. The latter includes Pfam protein domains, TIGRFAM families, antimicrobial resistance families, secondary metabolism biosynthetic genes, phage defense systems, microbial secretion system genes etc.


Gene

≡ Menu

b2132 [Escherichia coli str. K-12 substr. MG1655]

Genome viewer | Functional annotations | Analysis tools

GENE INFORMATION

Gene name
bglX

Genome
K-12 [Escherichia coli str. K-12 substr. MG1655] imported_2024-07-09

Position
NC_000913: complement(2219692..2221989) [2298 bp, 765 aa]

Product
beta-D-glucoside glucohydrolase, periplasmic

EGGNOG FAMILIES

Description
Belongs to the glycosyl hydrolase 3 family

Families

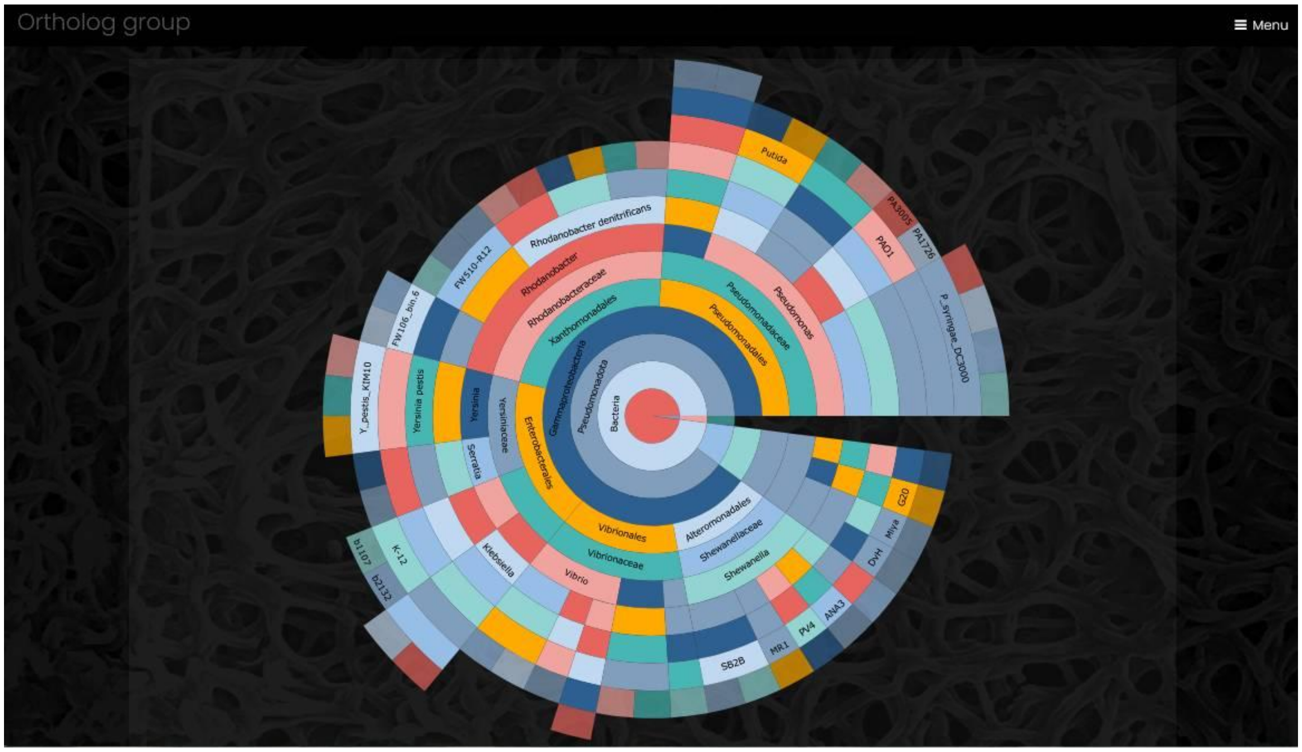
COG1472[root]	Show genome neighborhood conservation
COG1472[Bacteria]	Show genome neighborhood conservation
1MVIV[Pseudomonadota]	Show genome neighborhood conservation
1RMA0[Gammaproteobacteria]	Show genome neighborhood conservation
3XN99[Escherichia]	Show genome neighborhood conservation

The analysis tools section contains links to external sequence analysis tools for prediction of signal peptides and transmembrane segments, protein domain mapping and similarity search in sequence databases.

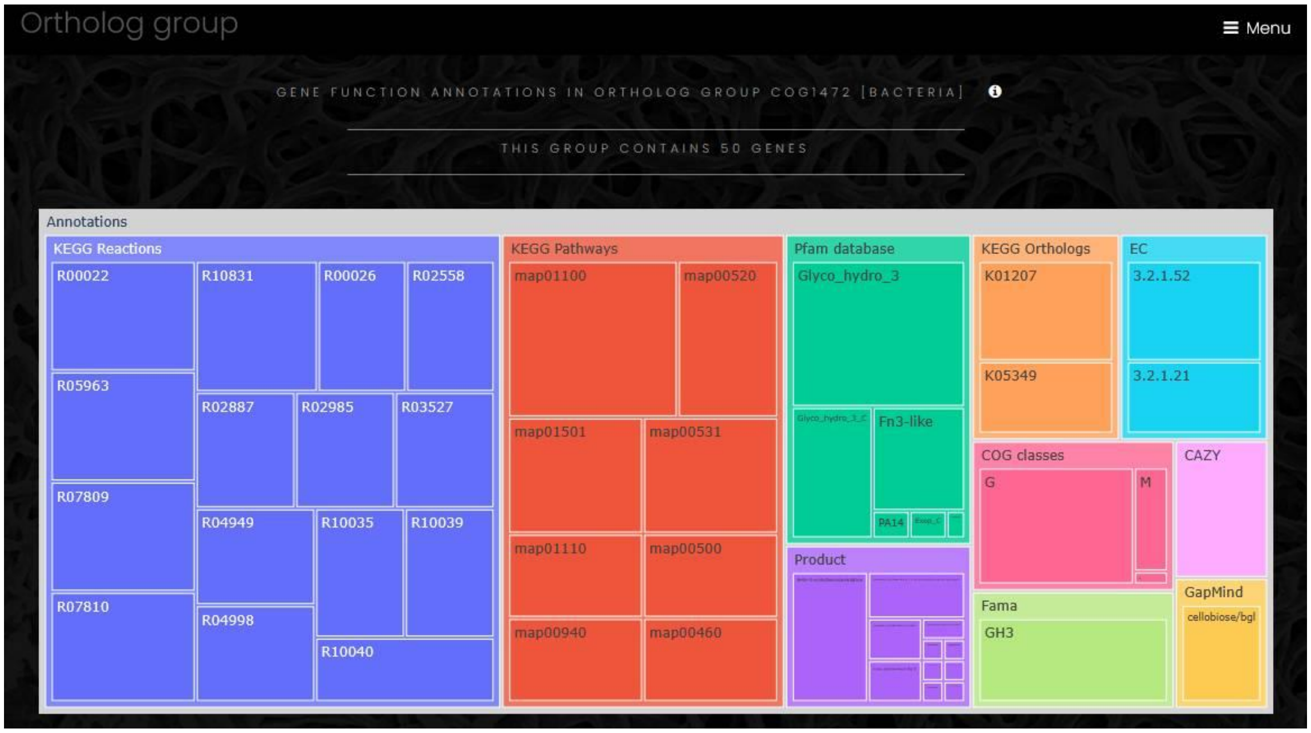
Ortholog family

Proteins are assigned to ortholog families by eggNOG-mapper, and all ortholog families keep eggNOG database identifiers.

An eggNOG family page has a link to a list of all genes in the family, a taxonomy profile of the family members and functional profile of the family.



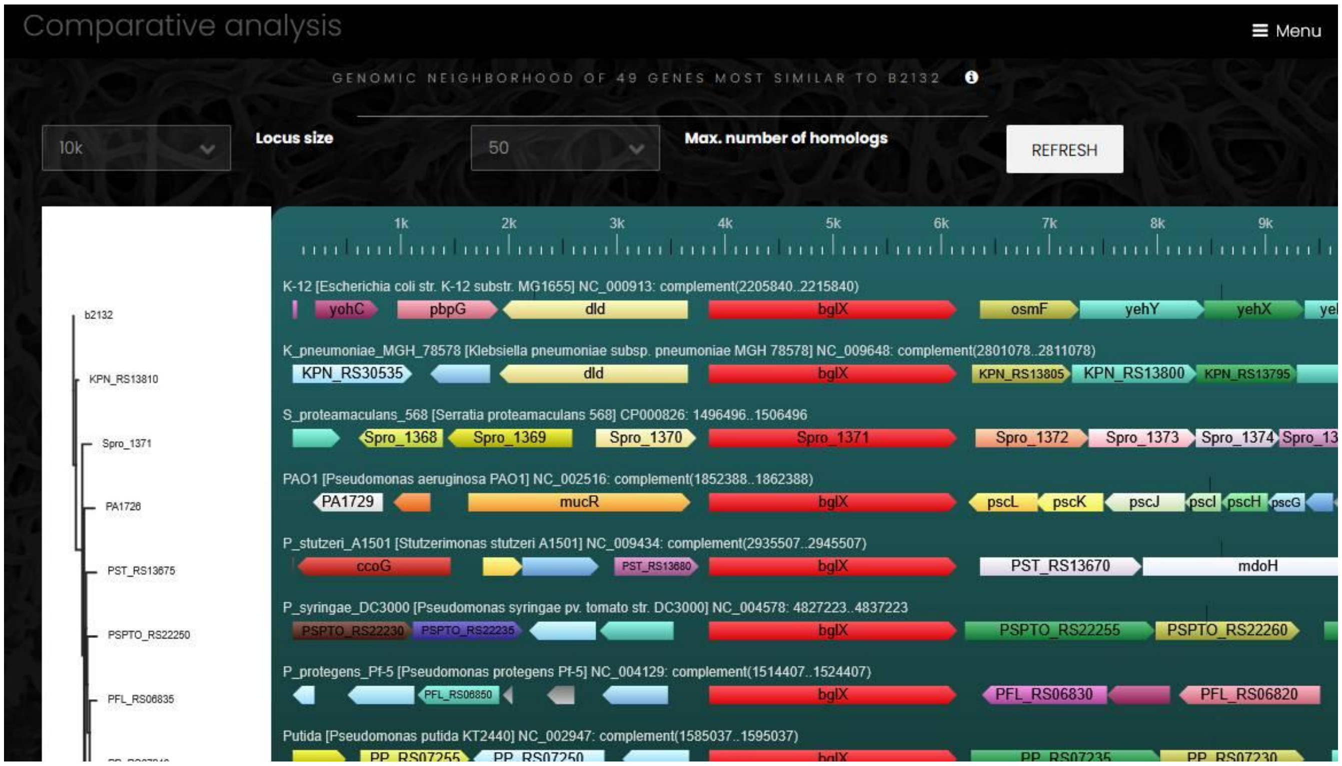
The taxonomy profile is a sunburst chart visualizing taxonomy data spanning outwards radially from the root node. It is similar to the genome taxonomy profile, but the outermost sectors of the chart contain genes, and a click on gene name opens the gene page.



The functional profile of ortholog family is a treemap chart composed of nested rectangles. The highest level of the functional profile shows functional classifications and functional annotation sources, and lower level of the profile shows functional categories. The size of a category rectangle corresponds to the number of genes in that category. Functional profiles can be exported as tab-separated text file.

Gene conserved neighborhood

A page for comparative analysis of genome neighborhood displays a region around a selected gene and several orthologs from a chosen eggNOG family. User can change the number of orthologs (up to 200 genes) and the size of the genome region (up to 100 kbp). There are several steps in the conserved gene neighborhood analysis. At the first step, the selected protein is compared to all members of the selected eggNOG family using Smith-Waterman sequence alignment algorithm, which selects 10-200 most similar proteins. At the next step, Muscle calculates multiple sequence alignment of the selected proteins, and a neighbor-joining phylogenetic tree is calculated from the multiple sequence alignment. At the final step, the selected genomes are arranged according to the order of sequences in the phylogenetic tree, a database search identifies neighbor genes of each protein-coding gene on the tree, and a color is assigned to each gene according to eggNOG family at the highest taxonomic level available. The starting gene and its orthologs always have red color, and genes that have no eggNOG ortholog mappings are always marked grey. A protein tree to the left from the genome neighborhood chart shows phylogenetic relationships of the orthologs. Users can save the phylogenetic tree and the protein multiple alignment by clicking on the buttons under the plot. This page also contains a treemap chart with functional profile of all displayed genes in the genome neighborhood in all the genomes.


Comparative analysis

GENOMIC NEIGHBORHOOD OF 49 GENES MOST SIMILAR TO B2132

Locus size: 10k

Max. number of homologs: 50

K-12 [Escherichia coli str. K-12 substr. MG1655] NC_000913: complement(2205840..2215840)

K_pneumoniae_MGH_78578 [Klebsiella pneumoniae subsp. pneumoniae MGH 78578] NC_009648: complement(2801078..2811078)

S_proteamaculans_568 [Serratia proteamaculans 568] CP000826: 1496496..1506496

PAO1 [Pseudomonas aeruginosa PAO1] NC_002516: complement(1852388..1862388)

P_stutzeri_A1501 [Stutzerimonas stutzeri A1501] NC_009434: complement(2935507..2945507)

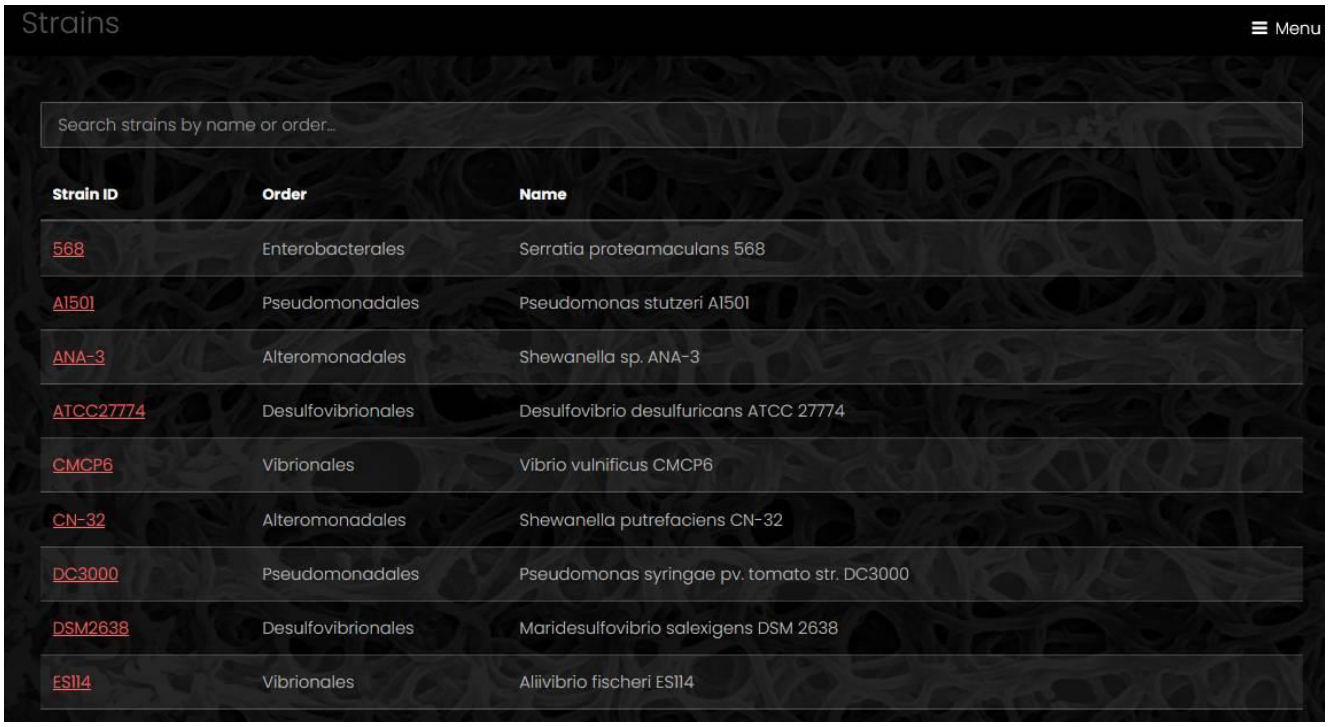
P_syringae_DC3000 [Pseudomonas syringae pv. tomato str. DC3000] NC_004578: 4827223..4837223

P_protegens_Pf-5 [Pseudomonas protegens Pf-5] NC_004129: complement(1514407..1524407)

Putida [Pseudomonas putida KT2440] NC_002947: complement(1585037..1595037)

Strain and Sample

A strain page contains list of genomes associated with the strain, taxonomy information and strain metadata entries (if any). The metadata entries a grouped by source. Similarly, a sample page contains list of genomes associated with the sample, description of the sample and sample metadata entries.



Strains

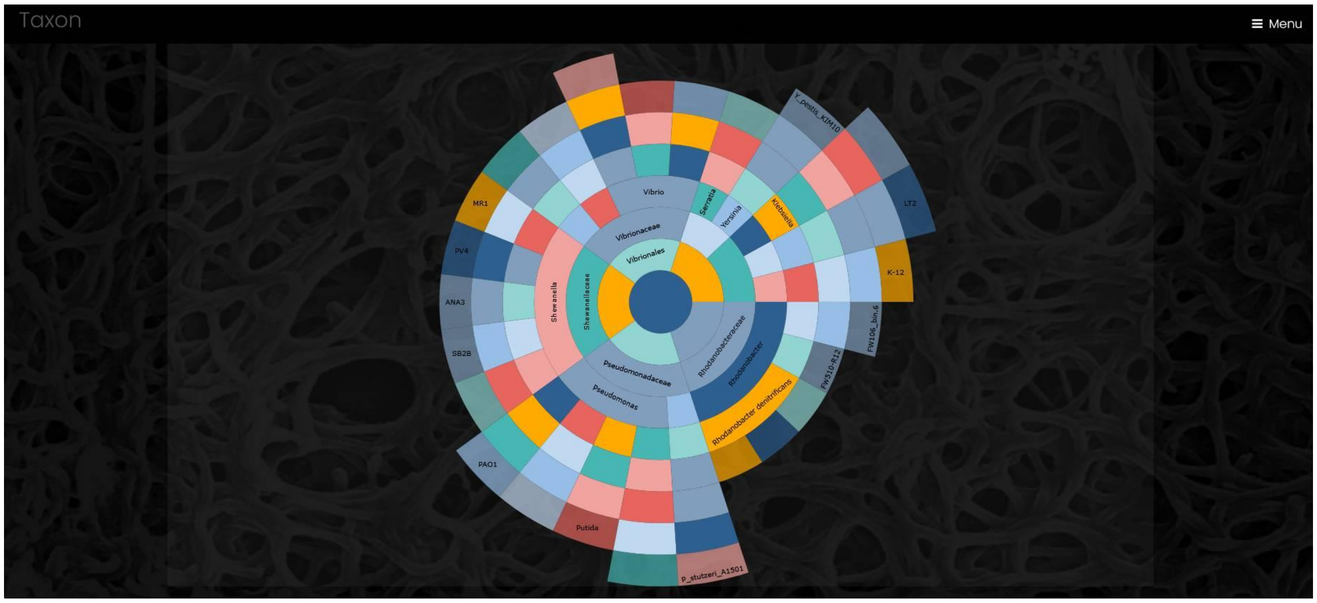
☰ Menu

Search strains by name or order...

Strain ID	Order	Name
568	Enterobacterales	Serratia proteamaculans 568
A1501	Pseudomonadales	Pseudomonas stutzeri A1501
ANA-3	Alteromonadales	Shewanella sp. ANA-3
ATCC27774	Desulfovibrionales	Desulfovibrio desulfuricans ATCC 27774
CMCP6	Vibrionales	Vibrio vulnificus CMCP6
CN-32	Alteromonadales	Shewanella putrefaciens CN-32
DC3000	Pseudomonadales	Pseudomonas syringae pv. tomato str. DC3000
DSM2638	Desulfovibrionales	Maridesulfovibrio salexigens DSM 2638
ES114	Vibrionales	Aliivibrio fischeri ES114

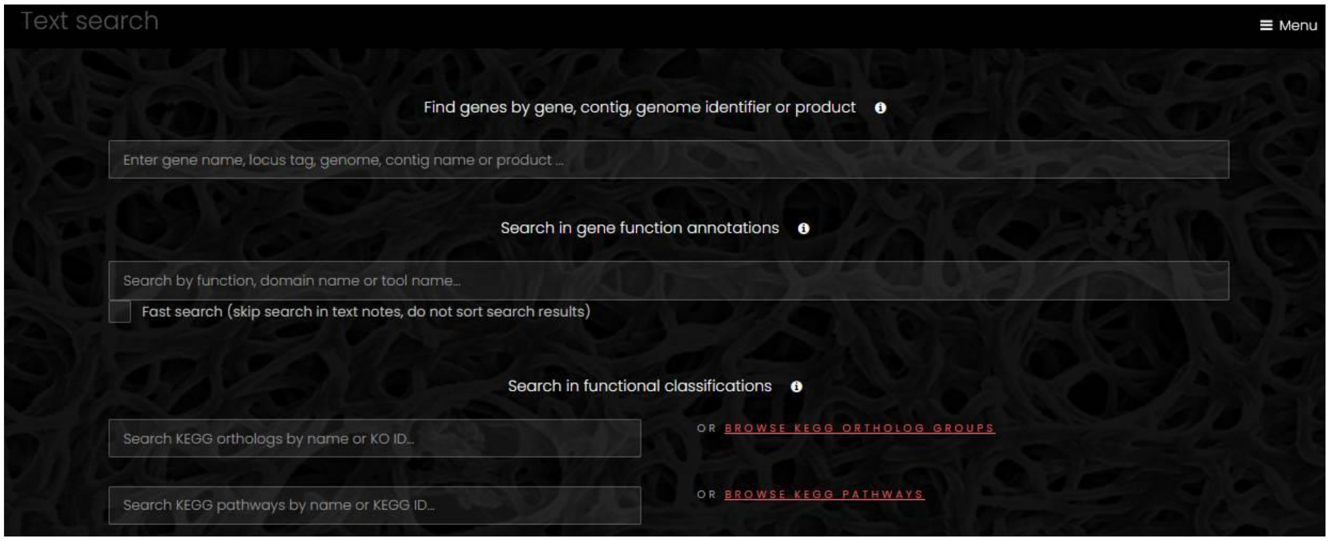
Taxon

A taxon page contains a sunburst chart for taxonomic profile of all genomes associated with the taxon, list of the genomes and list of strains from the taxon.



Text search

The text search page has text fields for search of genes by name or function, search in functional classifications and search in taxonomy. The search in genes by name first tries to find a unique exact match to a locus tag, and if there is no such match, it proceeds with a search in gene names, genome names and contig names. Next to each search field for functional classifications, there is a link that displays a full list of the functional classes.


Text search
≡ Menu
Find genes by gene, contig, genome identifier or product
Enter gene name, locus tag, genome, contig name or product ...
Search in gene function annotations
Search by function, domain name or tool name...
Fast search (skip search in text notes, do not sort search results)
Search in functional classifications
Search KEGG orthologs by name or KO ID...
OR BROWSE KEGG ORTHOLOG GROUPS
Search KEGG pathways by name or KEGG ID...
OR BROWSE KEGG PATHWAYS

Functional classifications

The genome import pipeline of GenomeDepot maps protein-coding genes to eight functional classifications using eggnog-mapper:

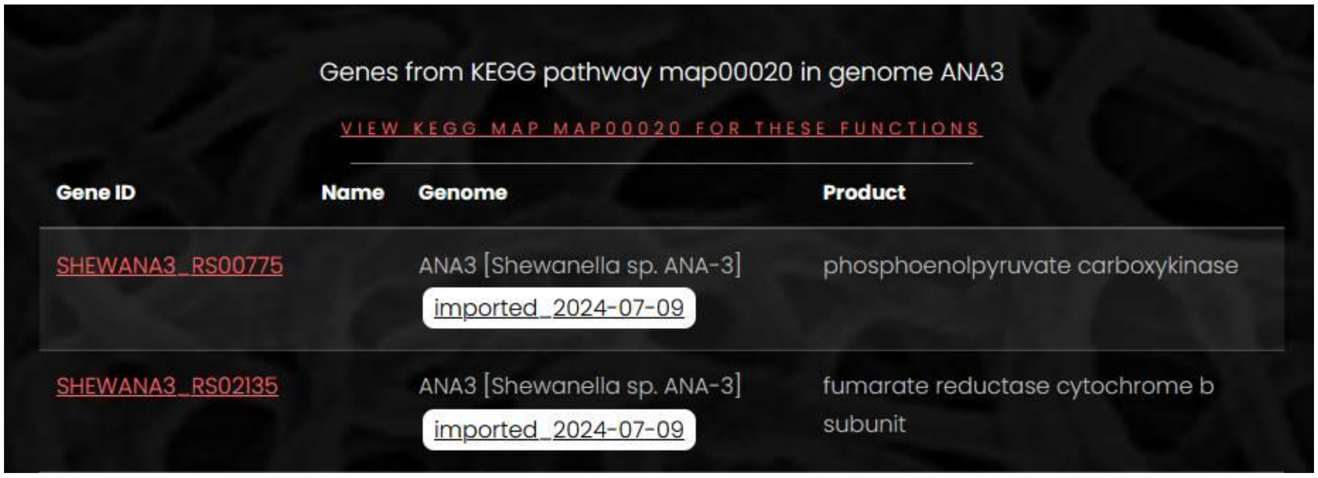
- KEGG Ortholog groups
- KEGG pathways
- KEGG reactions
- Enzyme families (EC)
- Transporter families (TCDB)
- CAZy families
- COG classes
- GO terms

A search in functional classifications returns matches in names and descriptions of the functional classes. In the search results page, a link from the name of a functional class opens a page with a list of genes mapped to that class.

A search from a genome page only returns genes mapped to functional classes in that genome. Links to lists of functional classes on a genome page return only the functional classes mapped to the genes of that genome.

KEGG pathway maps

A search for KEGG pathways from a genome page returns a list of pathways identified in that genome. To show genes of a KEGG pathway on the pathway’s map, click on KEGG ID of a pathway of interest. Next page will contain a list of gnes mapped to the pathway, with a link for KEGG map at the top of the page

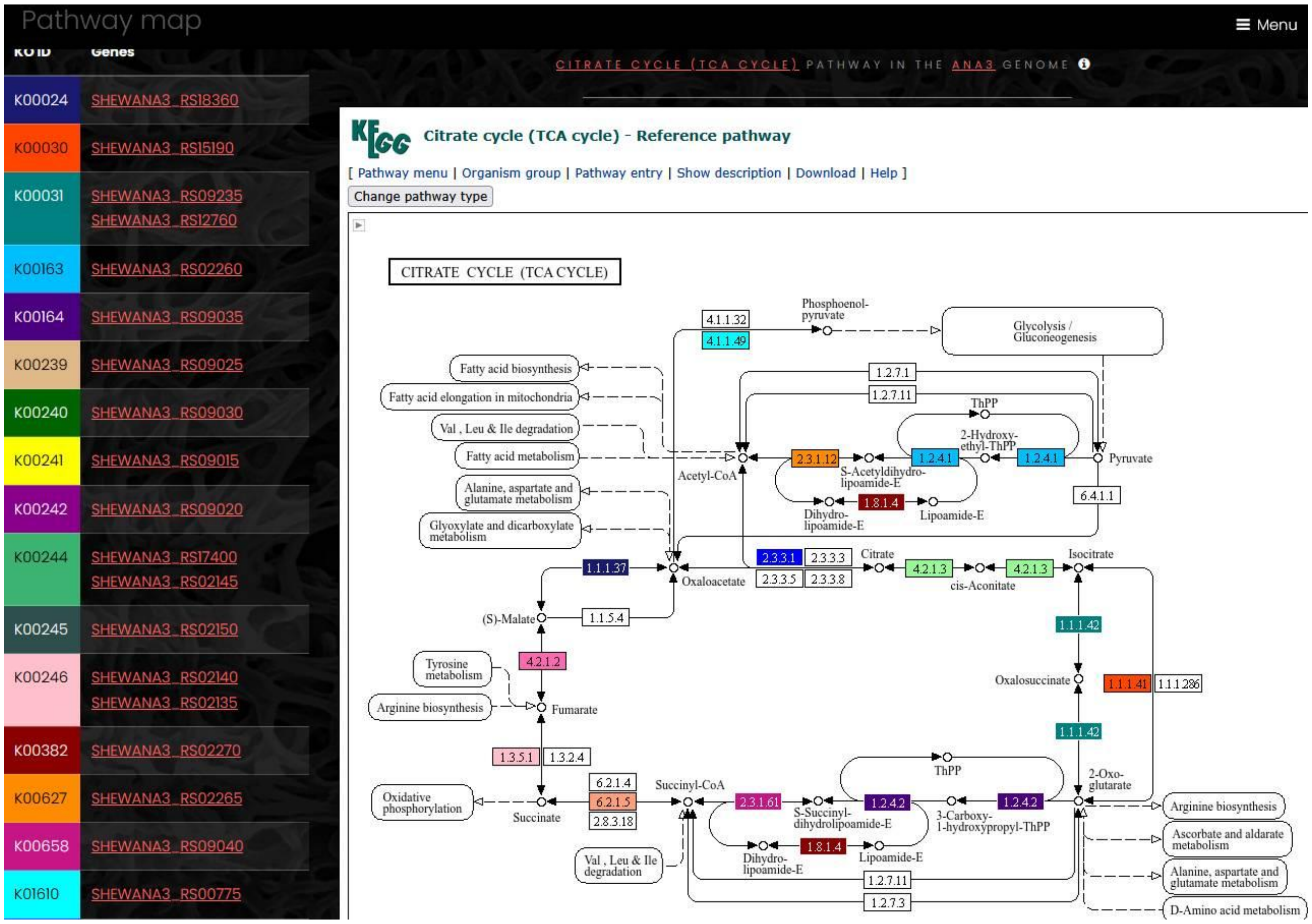


Genes from KEGG pathway map00020 in genome ANA3

VIEW KEGG MAP MAP00020 FOR THESE FUNCTIONS

Gene ID	Name	Genome	Product
SHEWANA3_RS00775		ANA3 [Shewanella sp. ANA-3] imported_2024-07-09	phosphoenolpyruvate carboxykinase
SHEWANA3_RS02135		ANA3 [Shewanella sp. ANA-3] imported_2024-07-09	fumarate reductase cytochrome b subunit

A click on the map link opens a page with a list of KEGG ortholog groups and genes on the left side. On the right side, a map of the pathway is loaded from the KEGG web-site. On the pathway map, colors of EC numbers correspond to colors of KEGG ortholog groups in the list. If an EC number on the map corresponds to more than one KEGG ortholog group, only one color is shown. Hover mouse pointer over the EC number on the map to display full list of KEGG ortholog groups mapped to this enzyme activity.



Pathway map

CITRATE CYCLE (TCA CYCLE) PATHWAY IN THE ANA3 GENOME

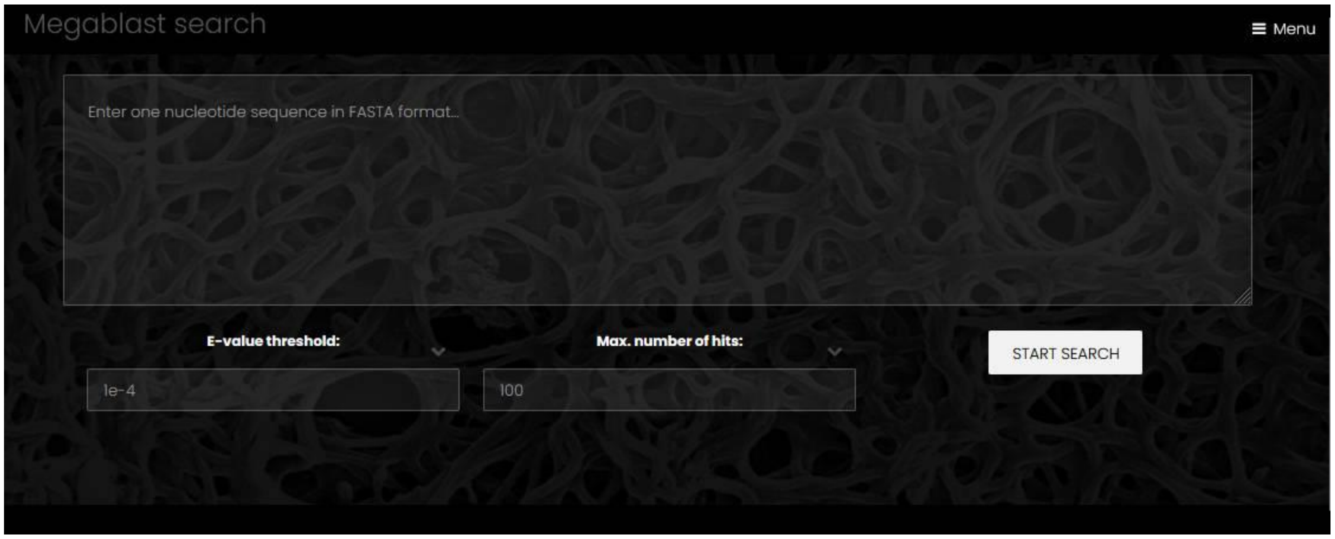
KO ID	Genes
K00024	SHEWANA3_RS18360
K00030	SHEWANA3_RS15190
K00031	SHEWANA3_RS09235 SHEWANA3_RS12760
K00163	SHEWANA3_RS02260
K00164	SHEWANA3_RS09035
K00239	SHEWANA3_RS09025
K00240	SHEWANA3_RS09030
K00241	SHEWANA3_RS09015
K00242	SHEWANA3_RS09020
K00244	SHEWANA3_RS17400 SHEWANA3_RS02145
K00245	SHEWANA3_RS02150
K00246	SHEWANA3_RS02140 SHEWANA3_RS02135
K00382	SHEWANA3_RS02270
K00627	SHEWANA3_RS02265
K00658	SHEWANA3_RS09040
K01610	SHEWANA3_RS00775

≡ Menu
Citrate cycle (TCA cycle) - Reference pathway
[Pathway menu | Organism group | Pathway entry | Show description | Download | Help]
Change pathway type
CITRATE CYCLE (TCA CYCLE)
Phosphoenol-pyruvate
Glycolysis / Gluconeogenesis
Fatty acid biosynthesis
Fatty acid elongation in mitochondria
Val , Leu & Ile degradation
Fatty acid metabolism
Alanine, aspartate and glutamate metabolism
Glyoxylate and dicarboxylate metabolism
Acetyl-CoA
Pyruvate
S-Acetyldihydro-lipoamide-E
Dihydro-lipoamide-E
Lipoamide-E
2-Hydroxy-ethyl-ThPP
ThPP
Oxaloacetate
Citrate
cis-Aconitate
Isocitrate
(S)-Malate
Oxalosuccinate
Tyrosine metabolism
Arginine biosynthesis
Fumarate
2-Oxo-glutarate
Succinyl-CoA
Succinate
Oxidative phosphorylation
S-Succinyl-dihydrolipoamide-E
3-Carboxy-1-hydroxypropyl-ThPP
Ascorbate and aldarate metabolism
D-Amino acid metabolism

Since KEGG maps are loaded from the external web-site, this page requires web-access to the KEGG web-site to work properly. The loading speed of KEGG map web-page may vary.

Sequence search

Pages for search by nucleotide or protein sequence have the text area for query sequence and dropdown lists for E-value threshold and maximum number of hits to display. The query sequence must be in the FASTA format, without leading spaces and empty lines. The protein sequence similarity search uses BLASTP, and the nucleotide sequence similarity search uses megablast. The page of protein search results displays link to the hit gene, genome name, %identity, alignment length, query coverage %, E-value and bit-score. The page of nucleotide search results displays link to the hit sequence, genome name, %identity, alignment length, query coverage %, E-value and bit-score. A click on the nucleotide search hit opens a genome page with the hit region highlighted.

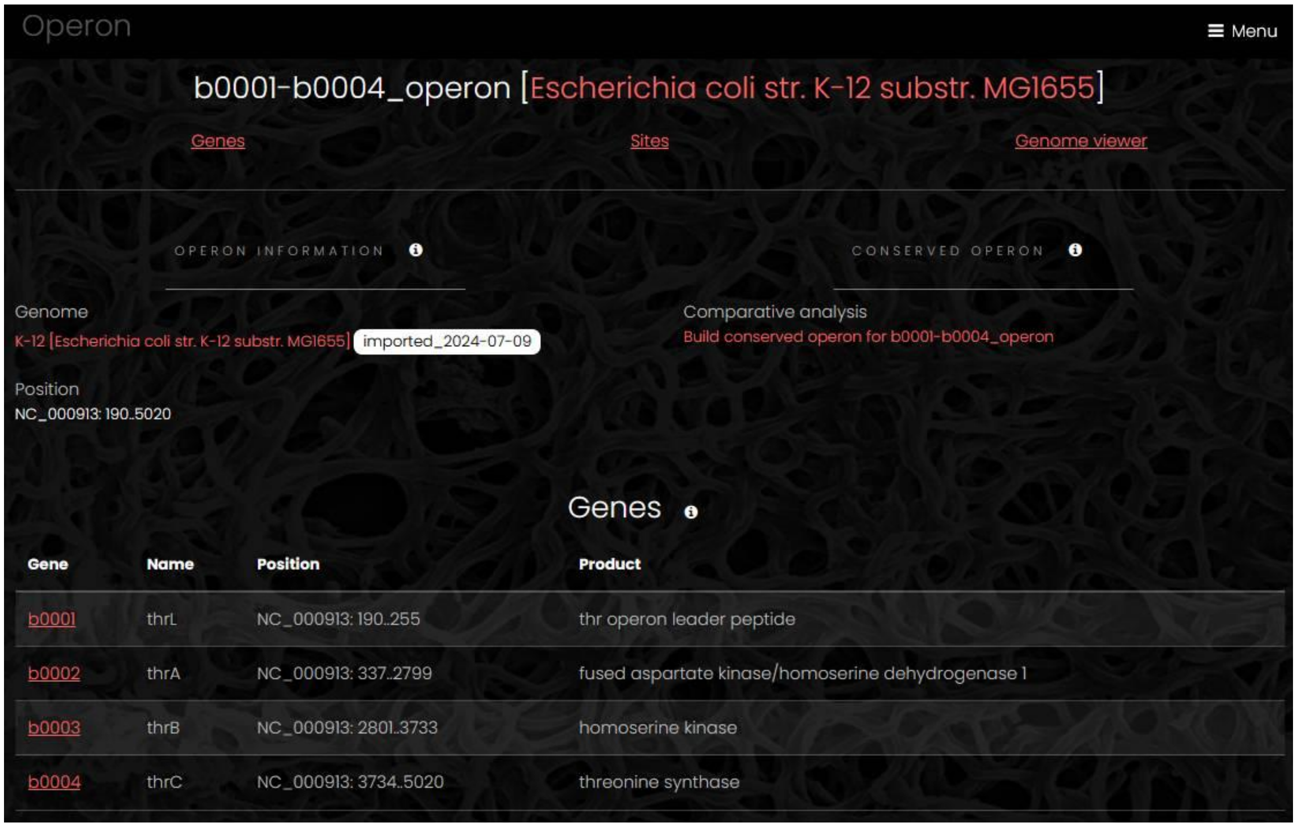


E-value threshold: 1e-4

Max. number of hits: 100

Operon

An operon page has three or four sections: operon information section, list of genes, list of sites (if any) and interactive genome viewer. The information section contains link to the genome page, sequence name with operon start and end positions and a link to conserved operon page. The list of genes contains name, position and product of all operon members. The genome viewer displays the genome with the operon region highlighted.



Operon

≡ Menu

b0001-b0004_operon [Escherichia coli str. K-12 substr. MG1655]

Genes Sites Genome viewer

OPERON INFORMATION

Genome
K-12 [Escherichia coli str. K-12 substr. MG1655] imported_2024-07-09

Position
NC_000913: 190..5020

CONSERVED OPERON

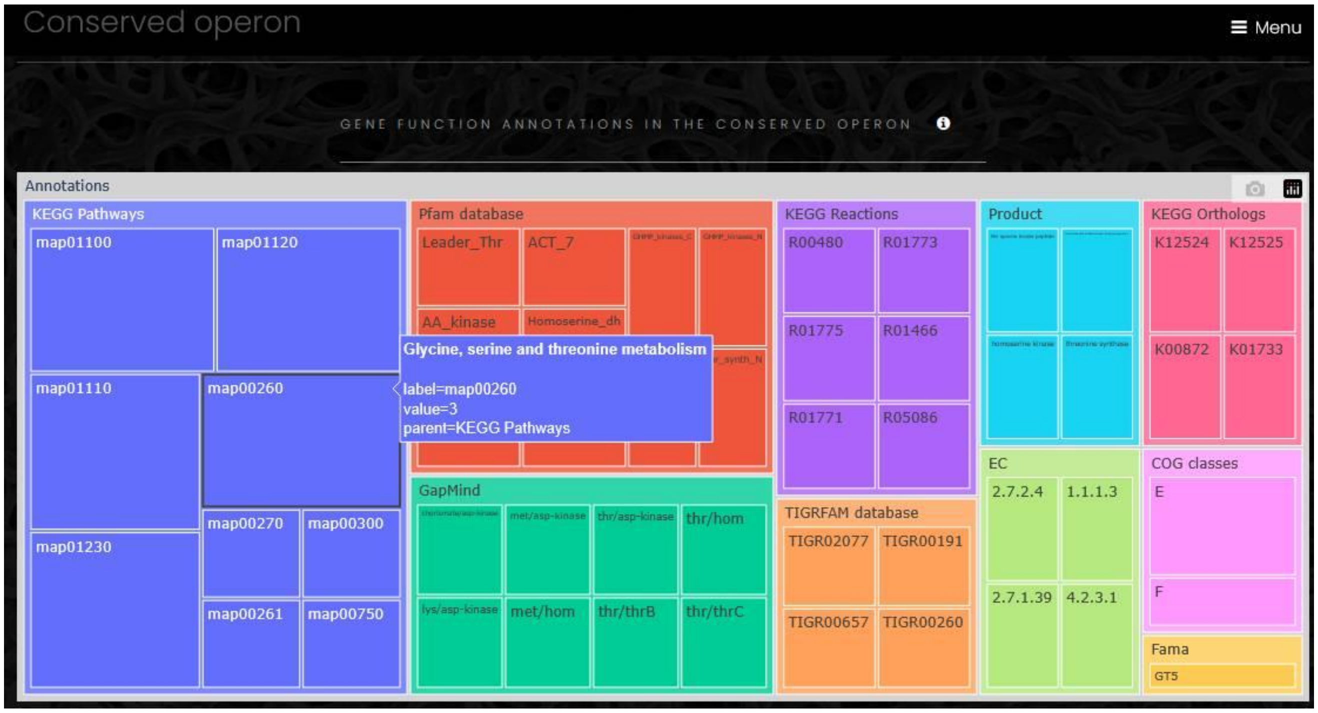
Comparative analysis
Build conserved operon for b0001-b0004_operon

Genes

Gene	Name	Position	Product
b0001	thrL	NC_000913: 190..255	thr operon leader peptide
b0002	thrA	NC_000913: 337..2799	fused aspartate kinase/homoserine dehydrogenase 1
b0003	thrB	NC_000913: 2801..3733	homoserine kinase
b0004	thrC	NC_000913: 3734..5020	threonine synthase

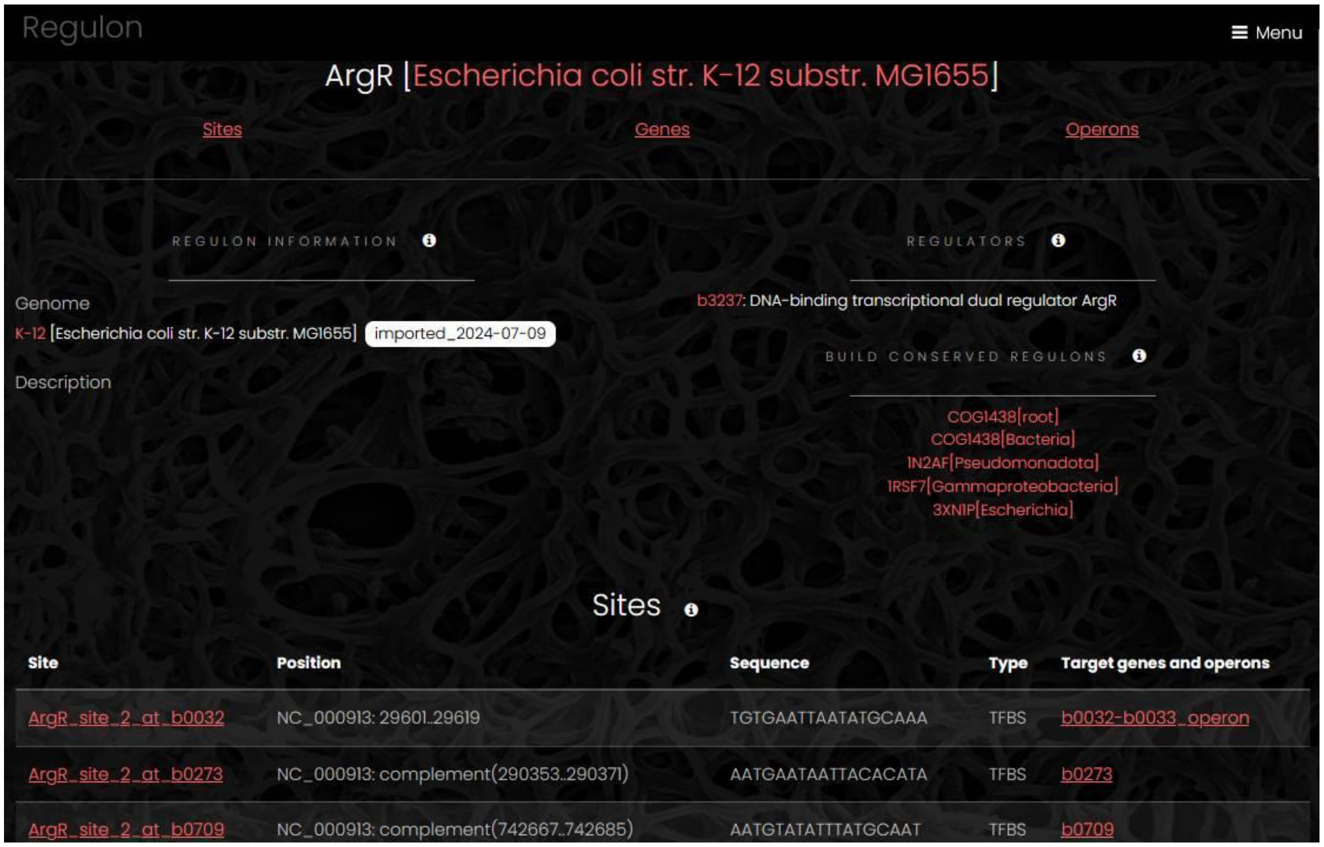
Conserved operon

A conserved operon app displays the result of operon comparative analysis. To build a conserved operon, at first, the app selects an eggNOG family for each gene in the operon at the lowest taxonomic level, so only closest homologs are selected for the analysis. Next, a database search identifies all operons that include members of the selected eggNOG families. Genes from these operons are collected for functional profiling. The resulting conserved operon page has three parts: taxonomic profile, functional profile and a list of operons. The taxonomic profile is a sunburst chart similar to the chart on the genomes list page with operons or genes in outermost sectors. The functional profile is a treemap chart displaying functional annotations of all genes in the conserved operon. The operon list contains links to all operons and all genomes in the conserved operon with the location and the number of genes for each operon.



Regulon

A regulon page has four sections: regulon information, a list of sites, a list of operons and a list of genes. The regulon information section contains a regulon description and links to the genome page, regulatory protein page and conserved regulon analysis app. The list of sites displays for every site a link to site page, site position, sequence and a link to target gene or operon page. The list of genes displays a link to the gene page, gene name, position and product for each target gene, which is not a member of an operon. The list of target operons displays a link to the operon page, operon position and list of genes in the operon.



ArgR [Escherichia coli str. K-12 substr. MG1655]

Sites Genes Operons

REGULON INFORMATION

Genome

K-12 [Escherichia coli str. K-12 substr. MG1655] imported_2024-07-09

Description

REGULATORS

b3237: DNA-binding transcriptional dual regulator ArgR

BUILD CONSERVED REGULONS

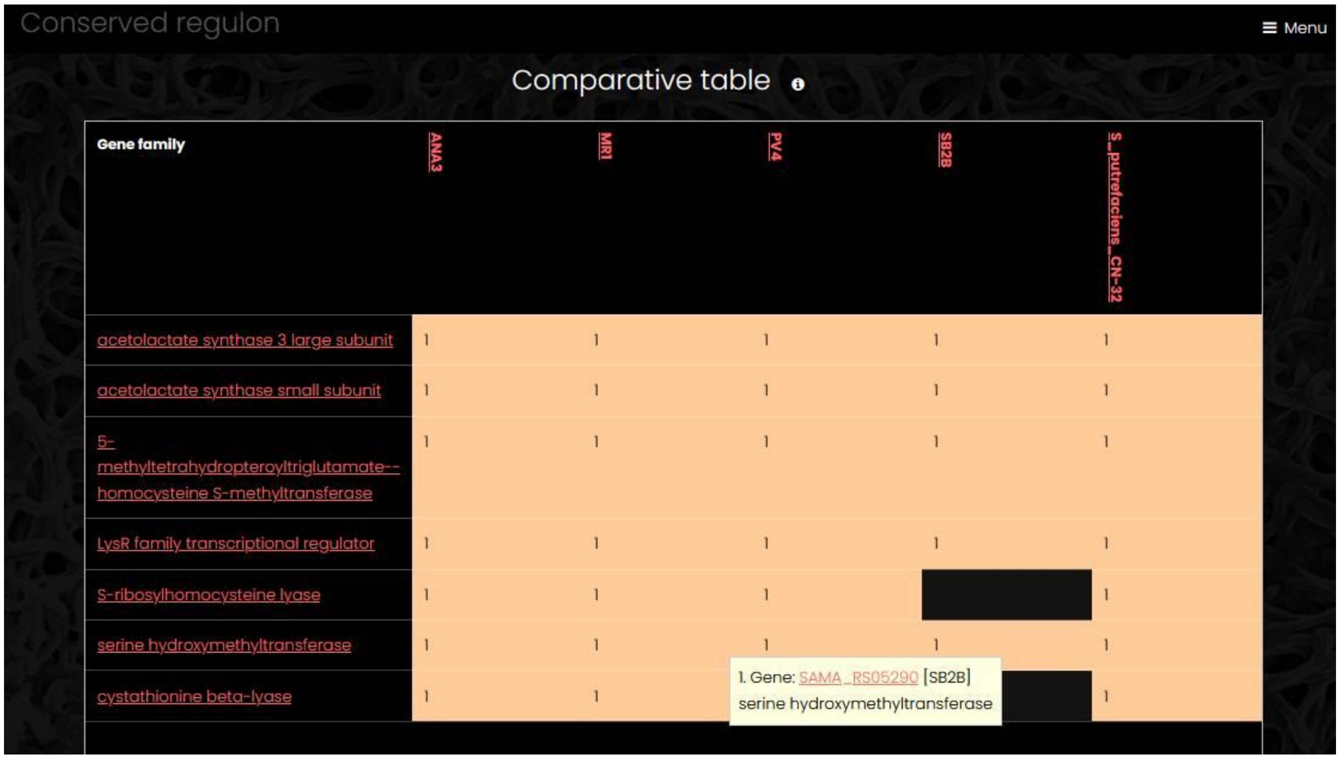
COG1438[root]
COG1438[Bacteria]
1N2AF[Pseudomonadota]
1RSF7[Gammaproteobacteria]
3XNIP[Escherichia]

Sites

Site	Position	Sequence	Type	Target genes and operons
ArgR_site_2_at_b0032	NC_000913: 29601..29619	TGTGAATTAATATGCAAA	TFBS	b0032-b0033_operon
ArgR_site_2_at_b0273	NC_000913: complement(290353..290371)	AATGAATAATTACACATA	TFBS	b0273
ArgR_site_2_at_b0709	NC_000913: complement(742667..742685)	AATGTATATTTATGCAAT	TFBS	b0709

Conserved regulon

A conserved regulon app displays the result of comparative regulon analysis. Regulons controlled by regulatory proteins from the same eggNOG family constitute a conserved regulon. Since a regulatory protein can belong to more than one eggNOG family at different taxonomic levels, several conserved regulons can be constructed for one regulatory protein. The conserved regulon page has four sections. The regulator section shows a list of gene products in a selected eggNOG family and a link to the list of all eggNOG family members. The comparative table section displays eggNOG families in rows and regulons in columns. The cells marked orange show number of target genes from an eggNOG family in a regulon. The regulons section contains links to regulons and regulatory genes included into the conserved regulon. The sites section contains list of all sites from all regulons included into the conserved regulon.



Comparative table

Gene family	ANA3	MR1	PV4	SB2B	S_putrefaciens_CN-32
acetolactate synthase 3 large subunit	1	1	1	1	1
acetolactate synthase small subunit	1	1	1	1	1
5-methyltetrahydropteroyltriglutamate--homocysteine S-methyltransferase	1	1	1	1	1
LysR family transcriptional regulator	1	1	1	1	1
S-ribosylhomocysteine lyase	1	1	1		1
serine hydroxymethyltransferase	1	1	1	1	1
cystathionine beta-lyase	1	1			1

1. Gene: SAMA_RS05290 [SB2B]
serine hydroxymethyltransferase

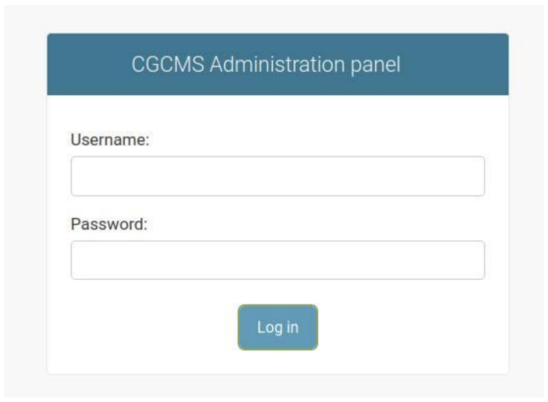
Site

A site page has three sections. The site information section contains site coordinates, site sequence and a link to the genome page. The target operons section displays a link to the operon page, operon position and a list of genes for each target operon. The genome viewer section displays the genome with the site highlighted.

Portal administration guide

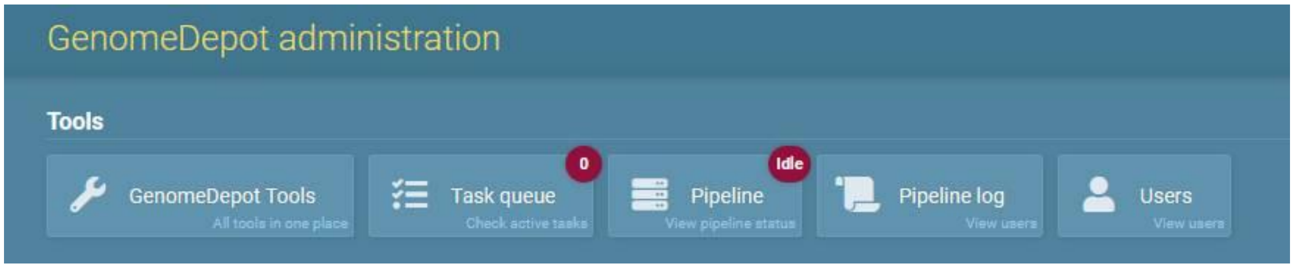
There are two ways for administering a GenomeDepot-based web portal: Django’s command-line utility and GenomeDepot administration panel. The command-line utility works in an active virtual environment and supports a limited set of GenomeDepot commands mostly for import and export of large sets of data (see the command-line utility section). The GenomeDepot administration panel provides access to individual data entries and GenomeDepot tools for data import and analysis.

Log into administration panel



The administration panel link is https://your_site_URL/admin/. Sign in with the superuser username and password created during the GenomeDepot installation.

Administration portal menu

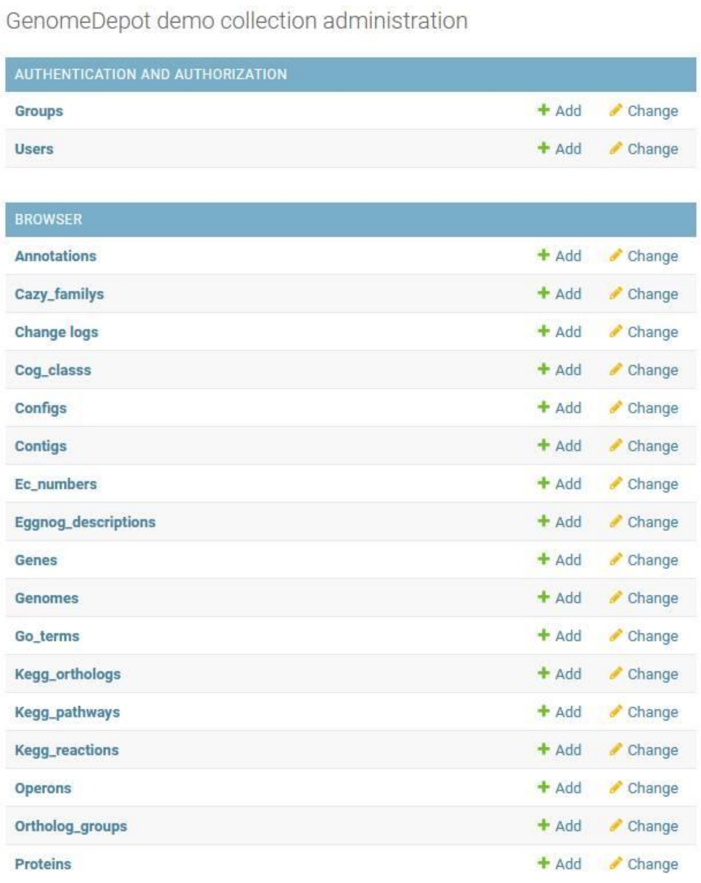


The tools menu at the top of administration panel pages has five buttons:

- GenomeDepot tools: opens a page with links to GenomeDepot administration tools,
- Tasks: opens a list of tasks in the queue,
- Pipeline: opens a list of Django Q clusters,
- Pipeline log: opens a page with GenomeDepot pipeline log file,
- Users: opens a list of GenomeDepot users,

Content management pages

The main page of the administration panel displays a list of content types. You can click on a content type to go to a page that lists all associated records. On those pages, you can view, modify, add or delete records (see the “Managing data in administration panel pages” section). You can also directly click the Add link next to each content type to start creating a record of that type.


GenomeDepot demo collection administration

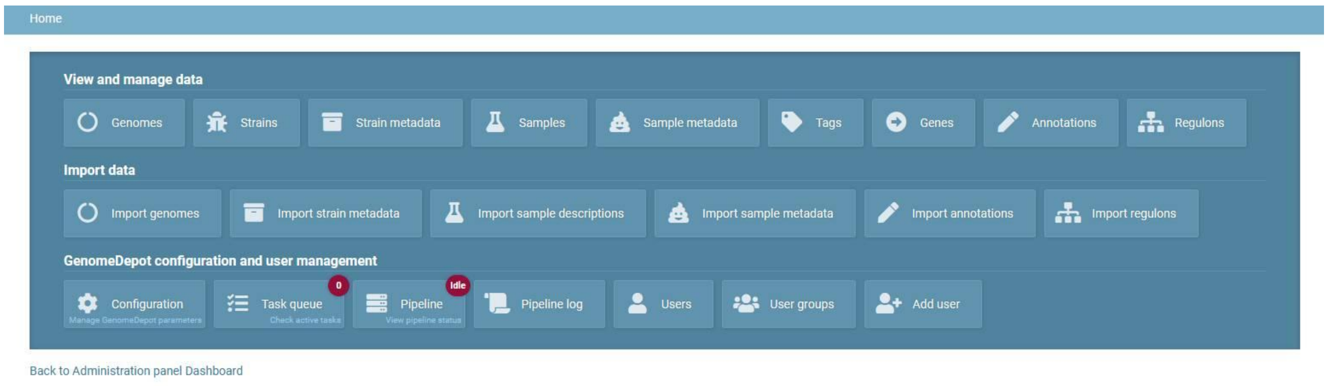
AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

BROWSER		
Annotations	+ Add	Change
Cazy_familys	+ Add	Change
Change logs	+ Add	Change
Cog_classs	+ Add	Change
Configs	+ Add	Change
Contigs	+ Add	Change
Ec_numbers	+ Add	Change
Eggnog_descriptions	+ Add	Change
Genes	+ Add	Change
Genomes	+ Add	Change
Go_terms	+ Add	Change
Kegg_orthologs	+ Add	Change
Kegg_pathways	+ Add	Change
Kegg_reactions	+ Add	Change
Operons	+ Add	Change
Ortholog_groups	+ Add	Change
Proteins	+ Add	Change

Pipeline status

The Tasks queue and Pipeline buttons in the top menu of administration pages show the number of currently active tasks in red circle and the status of the GenomeDepot pipeline. Normally, the numbers of tasks is either 0 or 1, because the administration interface prevents submissions of new tasks if there are tasks in the queue. The number on the Tasks button is 1 if there is a task currently running and 0 if there is no such task. The status on the Pipeline button can be “Off” is the Django Q worker was not started, “Idle” if there are no active tasks, or “Busy” if the pipeline is running a job. Running more than one Django Q worker for one genome collection is potentially dangerous for the data integrity and should be avoided at all times.

Tools page



The tools page has three sections: View and manage data, Import data, GenomeDepot configuration and user management.

The View and manage data section contains links to lists of genomes, strains, strain metadata, samples, sample metadata, genome tags, genes, annotations and regulons. These links open tables of existing data.

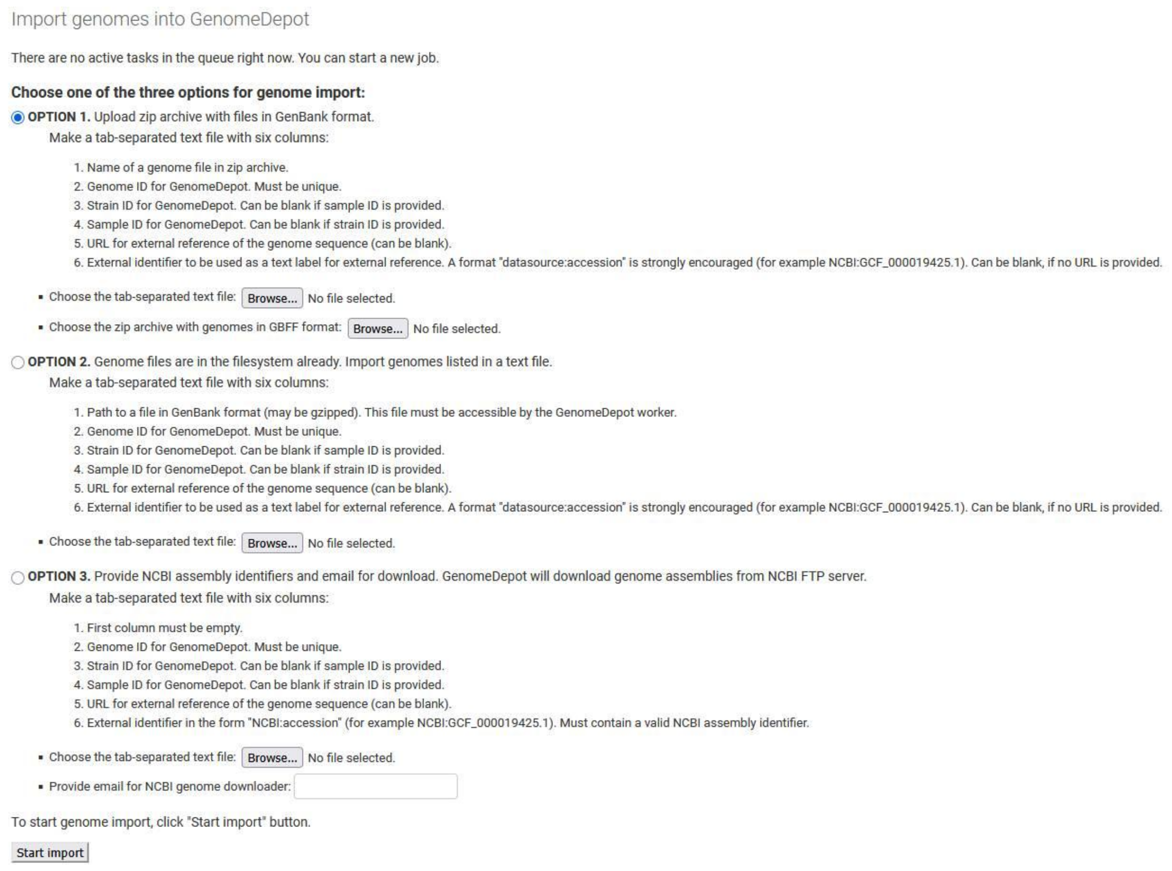
The Import data section contains links to import tools for genomes, strain metadata, sample descriptions, sample metadata, gene annotations and regulons.

The last section on the tools page has the same links as the menu on the administration panel dashboard (see above), with an additional links to the User groups and Add user pages.

Use GenomeDepot administration tools for data import

Add genomes

GenomeDepot stores genome sequences in GenBank format files and protein sequences in the database. The genome import pipeline generates the genome files and creates database entries for genomes, strains, genes and proteins. The import pipeline also runs EggNOG-mapper to assign proteins to ortholog families and POEM to predict operons. In addition to ortholog families, EggNOG-mapper associates proteins with KEGG orthologs, KEGG reactions, EC numbers, TC families and other functional classifications.



Import genomes into GenomeDepot

There are no active tasks in the queue right now. You can start a new job.

Choose one of the three options for genome import:

[✓ **OPTION 1.** Upload zip archive with files in GenBank format.]

Make a tab-separated text file with six columns:

1. Name of a genome file in zip archive.
2. Genome ID for GenomeDepot. Must be unique.
3. Strain ID for GenomeDepot. Can be blank if sample ID is provided.
4. Sample ID for GenomeDepot. Can be blank if strain ID is provided.
5. URL for external reference of the genome sequence (can be blank).
6. External identifier to be used as a text label for external reference. A format "datasource:accession" is strongly encouraged (for example NCBI:GCF_000019425.1). Can be blank, if no URL is provided.

- Choose the tab-separated text file: Browse... No file selected.
- Choose the zip archive with genomes in GBFF format: Browse... No file selected.

OPTION 2. Genome files are in the filesystem already. Import genomes listed in a text file.

Make a tab-separated text file with six columns:

1. Path to a file in GenBank format (may be gzipped). This file must be accessible by the GenomeDepot worker.
2. Genome ID for GenomeDepot. Must be unique.
3. Strain ID for GenomeDepot. Can be blank if sample ID is provided.
4. Sample ID for GenomeDepot. Can be blank if strain ID is provided.
5. URL for external reference of the genome sequence (can be blank).
6. External identifier to be used as a text label for external reference. A format "datasource:accession" is strongly encouraged (for example NCBI:GCF_000019425.1). Can be blank, if no URL is provided.

- Choose the tab-separated text file: Browse... No file selected.

OPTION 3. Provide NCBI assembly identifiers and email for download. GenomeDepot will download genome assemblies from NCBI FTP server.

Make a tab-separated text file with six columns:

1. First column must be empty.
2. Genome ID for GenomeDepot. Must be unique.
3. Strain ID for GenomeDepot. Can be blank if sample ID is provided.
4. Sample ID for GenomeDepot. Can be blank if strain ID is provided.
5. URL for external reference of the genome sequence (can be blank).
6. External identifier in the form "NCBI:accession" (for example NCBI:GCF_000019425.1). Must contain a valid NCBI assembly identifier.

- Choose the tab-separated text file: Browse... No file selected.
- Provide email for NCBI genome downloader:

To start genome import, click "Start import" button.

Start import

In the Tools page of the administration panel, click the Import genomes link. Before genome import, prepare input files. There are three ways to import genome files into GenomeDepot:

1. Copy genome files in Genbank format (can be gzipped) to the server's directory with read permissions for the GenomeDepot user. Make a tab-separated file with six columns:
    - full path of a Genbank file
    - genome ID (can contain letters, digits, dots, hyphens and underscores only)
    - strain ID (can contain letters, digits, dots, hyphens and underscores only)
    - sample ID (can contain letters, digits, dots, hyphens and underscores only)
    - URL (link to NCBI genome assembly etc., optional)
    - External ID (will be displayed on the genome page. For example, "NCBI:GCF_000006945.2" for NCBI assembly. Keep it short but meaningful.)

    Click the "Browse" button to find and select the tab-separated file for upload. Then press the "Start import" button.

2. Make a zip archive with genome files in Genbank format. Make a tab-separated file with six columns:
    - name of a Genbank file
    - genome ID (can contain letters, digits, dots, hyphens and underscores only)
    - strain ID (can contain letters, digits, dots, hyphens and underscores only)
    - sample ID (can contain letters, digits, dots, hyphens and underscores only)
    - URL (link to NCBI genome assembly etc., optional)
    - External ID (will be displayed on the genome page. For example, "NCBI:GCF_000006945.2" for NCBI assembly. Make it short but meaningful.)

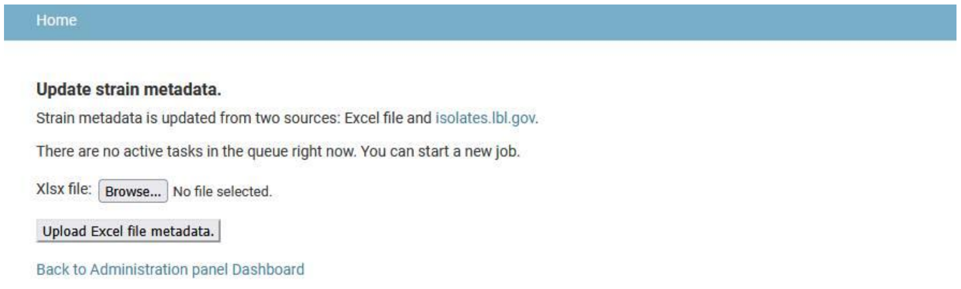
    Click the "Browse" button to find and select the tab-separated file for upload. Click the next "Browse" button to find and choose a Zip archive with genomes. Then press the "Start import" button.

3. GenomeDepot can download genome assemblies from NCBI over ftp. It can save time, but if internet connection is not stable, downloaded files can be truncated. Make a tab-separated file with six columns:
    - NCBI genome assembly identifier (for example, GCF_000006945.2)
    - genome ID (can contain letters, digits, dots, hyphens and underscores only)
    - strain ID (can contain letters, digits, dots, hyphens and underscores only)
    - sample ID (can contain letters, digits, dots, hyphens and underscores only)
    - URL (link to NCBI genome assembly etc., optional)
    - External ID (will be displayed on the genome page. For example, "NCBI:GCF_000006945.2" for NCBI assembly. Make it short but meaningful.)

    Click the "Browse" button to find and select the tab-separated file for upload. Enter an email into the text field (the email is not saved by GenomeDepot, it is only used by NCBI e-utils). Then press the "Start import" button.

Add strain metadata

Upload any organism metadata from an Excel file since the metadata is not collected by the genome import pipeline. The Excel file has to have one spreadsheet with strain identifiers in the leftmost column, metadata categories in the top row and metadata values in other cells. Empty cells and cells containing "None" are ignored. All imported metadata entries have "User-defined data" as a source and display a "No external link" message instead of link to external resource. An alternative method for organism metadata upload is using the command-line command "update_strain_metadata".



Update strain metadata.

Strain metadata is updated from two sources: Excel file and isolates.lbl.gov.

There are no active tasks in the queue right now. You can start a new job.

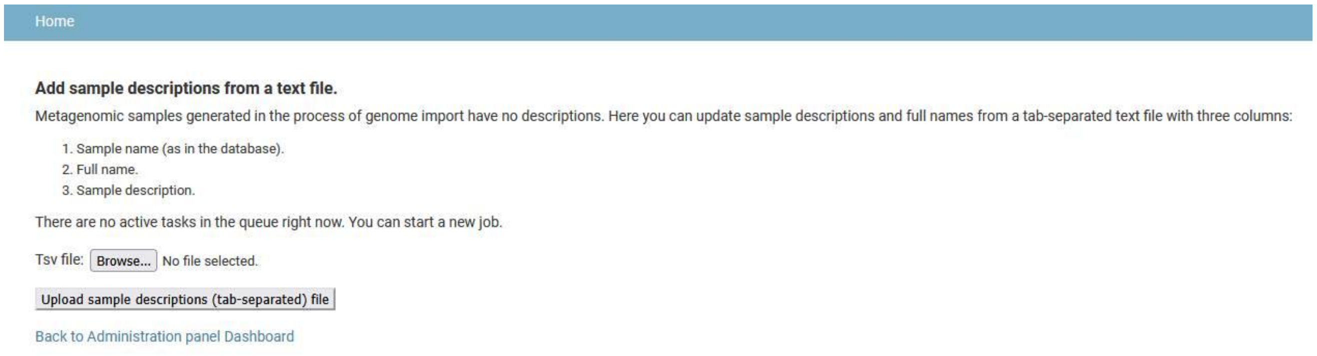
Xlsx file: Browse... No file selected.

Upload Excel file metadata.
Back to Administration panel Dashboard

Add sample descriptions

There are no descriptions in sample entries generated by genome import pipeline for metagenome-assembled genomes. Such descriptions can be uploaded from a tab-separated file with three columns:

- Sample name (as in the database).
- Sample full name.
- Description (text).



Add sample descriptions from a text file.

Metagenomic samples generated in the process of genome import have no descriptions. Here you can update sample descriptions and full names from a tab-separated text file with three columns:

1. Sample name (as in the database).
2. Full name.
3. Sample description.

There are no active tasks in the queue right now. You can start a new job.

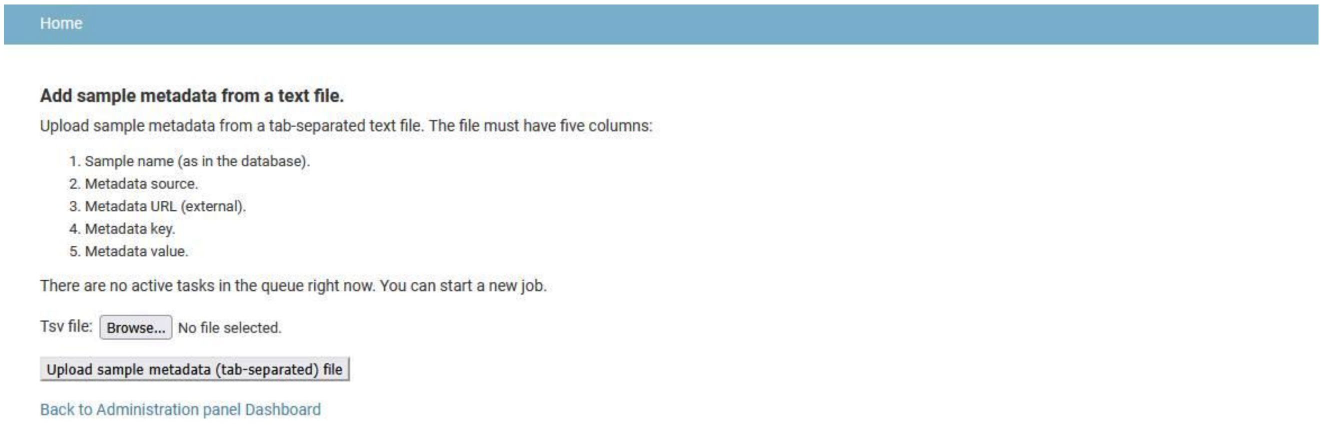
Tsv file: Browse... No file selected.

Upload sample descriptions (tab-separated) file
Back to Administration panel Dashboard

Add sample metadata

Sample metadata is not collected from genome files by the genome import pipeline, so users have to upload it later from a tab-separated file. The file must have five columns:

- Sample name (as in the database).
- Metadata source.
- Metadata URL (link to external resource).
- Metadata key (category name).
- Metadata value.


Home

Add sample metadata from a text file.

Upload sample metadata from a tab-separated text file. The file must have five columns:

1. Sample name (as in the database).
2. Metadata source.
3. Metadata URL (external).
4. Metadata key.
5. Metadata value.

There are no active tasks in the queue right now. You can start a new job.

Tsv file: Browse... No file selected.

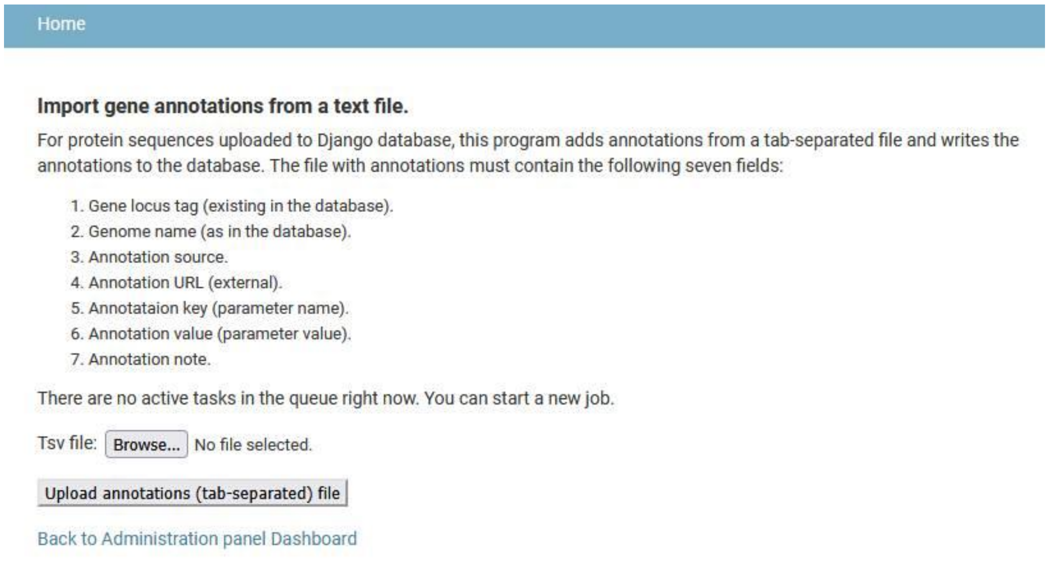
Upload sample metadata (tab-separated) file

Back to Administration panel Dashboard

Add gene annotations

Users can upload additional annotations for any gene in the database. A tab-separated file with annotations must contain seven columns fields:

- Gene locus tag (as in the database).
- Genome name (as in the database).
- Annotation source (name of a tool producing the annotation, a database, etc. Up to 30 symbols long).
- Annotation URL (link to external resource. Up to 300 symbols long).
- Annotation key (Short category name, like “Protein family” or “Domain”. Up to 30 symbols long).
- Annotation value (For example, identifier of a protein family or domain. Up to 50 symbols long).
- Annotation note (free-text description).


Home

Import gene annotations from a text file.

For protein sequences uploaded to Django database, this program adds annotations from a tab-separated file and writes the annotations to the database. The file with annotations must contain the following seven fields:

1. Gene locus tag (existing in the database).
2. Genome name (as in the database).
3. Annotation source.
4. Annotation URL (external).
5. Annotataion key (parameter name).
6. Annotation value (parameter value).
7. Annotation note.

There are no active tasks in the queue right now. You can start a new job.

Tsv file: Browse... No file selected.

Upload annotations (tab-separated) file

Back to Administration panel Dashboard

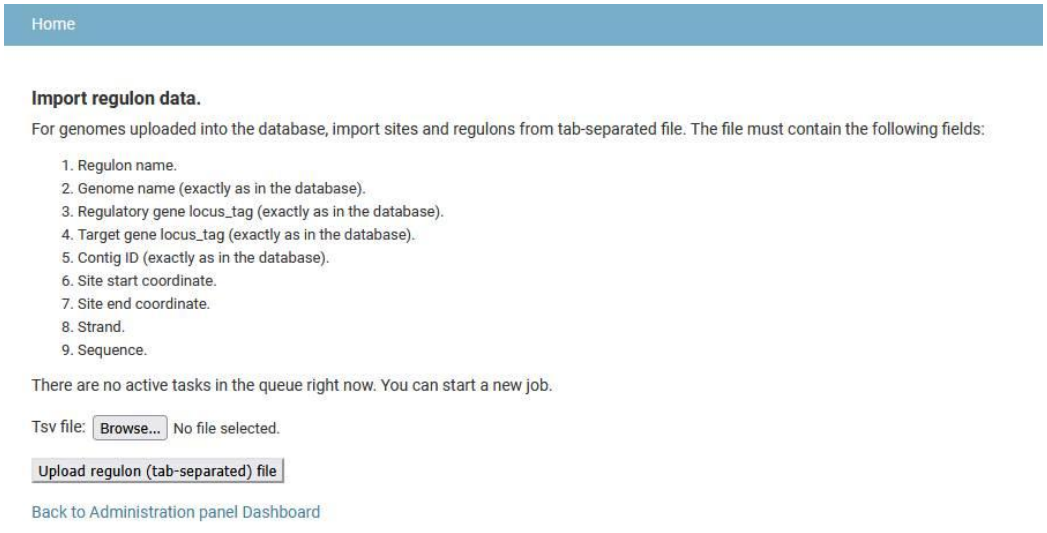
Add regulons

Regulons and sites can be uploaded from a tab-separated file. Each line in the file describes a regulatory interaction between a regulatory gene, a target gene and a binding site. A regulon groups together regulatory interactions for a single regulatory protein. The input file must contain the nine fields:

- Regulon name.

- Genome name (exactly as in the database)
- Regulatory gene locus_tag (exactly as in the database)
- Target gene locus_tag (exactly as in the database)
- Contig ID (exactly as in the database)
- Site start position
- Site end position
- Site strand
- Site sequence

If the target gene is the first gene in an operon, the site will be linked to this operon when imported.



Import regulon data.

For genomes uploaded into the database, import sites and regulons from tab-separated file. The file must contain the following fields:

1. Regulon name.
2. Genome name (exactly as in the database).
3. Regulatory gene locus_tag (exactly as in the database).
4. Target gene locus_tag (exactly as in the database).
5. Contig ID (exactly as in the database).
6. Site start coordinate.
7. Site end coordinate.
8. Strand.
9. Sequence.

There are no active tasks in the queue right now. You can start a new job.

Tsv file: Browse... No file selected.

Upload regulon (tab-separated) file
Back to Administration panel Dashboard

Managing users

Click the Add user button in the administration panel menu to add a new user with administration permissions. Users can have permissions to view, add or edit the data, manage other user accounts, view, add or delete tasks in the jobs queue etc. You can find more about Django administration panel in this tutorial:

https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Admin_site

Managing data in administration panel pages

Use administration panel pages to view, modify, add or delete data records. A page displays a list of records of a certain type.

To add a record, click an Add link in the top right part of the page.

To delete one or more records, select the records by clicking on checkboxes, choose the Delete action from the dropdown Actions menu and press Go.

To modify a record, click on it. Make the changes in the edit page for the record, then press SAVE, Save and add another, or Save and continue editing to save the record.

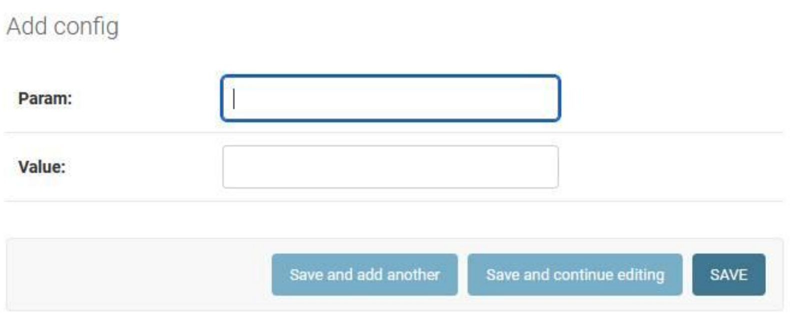
Records of many content types are auto-generated during genome import and do not need human intervention. Other content types (for example, regulons, gene annotations or strain and sample metadata records) have specialized tools for data import. So, only some content types are described in detail in this section: configuration parameters, genomes and tags.

Configuration parameters (configs)

View configuration parameters path: admin/browser/config/

To modify a configuration parameter, click on its name, change its properties and press the save button. Param is a parameter name, Value is a parameter value.

Add configuration parameter path: admin/browser/config/add/


Add config

Param:	
Value:	

Genomes

View genomes path: admin/browser/genome/

Click on the Add genome link opens the genomes import tool (see “Use GenomeDepot administration tools for data import”).

Commands available from the Actions menu for genomes

Delete genomes: select genomes in the list, choose “Delete genomes” in the Actions list and press “Go”. This action correctly deletes all genes, operons and sites associated with the deleted genomes and re-builds nucleotide and protein BLAST databases. Do not use the “Delete” button in the genome change page.

Run annotation tools: select genomes in the list, choose “Run annotation tools” in the Actions list and press “Go”. In the new window, check the tools to run and press “Start tools”.

Add genome tag: select genomes in the list, choose “Add a tag” in the Actions list and press “Go”. In the new window, choose a tag from the dropdown menu and press “Change”.

Remove genome tag: select genomes in the list, choose “Remove a tag” in the Actions list and press “Go”. In the new window, choose a tag from the dropdown menu and press “Change”.

Update static files and re-build search databases: this command re-creates the files for the interactive genome viewer for selected genomes and completely re-builds nucleotide and protein BLAST databases.

Genome tags

Upon genome import, a tag "imported" with the import date is auto-generated for a batch of the genomes processed. To add another tag, click the Add tag link on the Tags page (admin/browser/tag/). On the add page, enter a name and a description of the new tag, choose a background color and a text color, then click SAVE. A tag can be associated with genomes using Add genome tag action in the Genomes administration page.

GenomeDepot command-line utility

In addition to the administration panel, GenomeDepot uses Django's command-line utility for administration tasks. This utility is particularly useful for adding, deleting or changing multiple records at a time. The utility can be used even when the Apache web server was stopped. Before using the command-line utility, deactivate all conda environments (if any) and activate GenomeDepot virtual environment:

```
conda deactivate
source /path/to/genomedepot/genomedepot-venv/bin/activate
cd /path/to/genomedepot/apps/mygenomes/genome-depot/genomebrowser
```

then run

```
python manage.py <command> [parameters]
```

where command should be one of the commands listed in this document; parameters can be zero or more of the parameters available for that command.

Available commands

```
python manage.py delete_all_data
```

Deletes all data records from the GenomeDepot database except configuration parameters. Once deleted, the data cannot be restored. This command does not delete static genome files.

```
python manage.py delete_all_genomes
```

Deletes **all** Annotation, Gene, Protein, Genome, Strain, Sample records from the database.

```
python manage.py delete_genome [-g]
```

Deletes one genome with all genes and annotations from the database.

Parameters:

- **-g** Genome name

```
python manage.py delete_genomes [-i]
```

Deletes one or more genomes with all genes and annotations from the database

Parameters:

- **-i** File with a list of genome names

```
python manage.py export_config [-o]
```

Exports configuration parameters into a text file with key/value entries separated by "=" symbol

Parameters:

- **-o** Output file name

```
python manage.py export_genomes [-g] [-o]
```

Export genomes with all gene annotations in Genbank format

Parameters:

- **-g** Comma-separated list of genome names
- **-o** Output file name

```
python manage.py generate_static_files [-i]
```

Generates genome viewer static files for genomes from the input file and re-creates search databases

Parameters:

- **-i** File with a list of genome names

```
python manage.py import_annotations [-i]
```

Imports annotations for protein-coding genes from a tab-separated file and writes the annotations into the GenomeDepot database. The genes must be in the database. The input file with annotations must have seven fields:

- Gene locus tag (as in the database).
- Genome name (as in the database).
- Annotation source.
- Annotation URL (external).
- Annotation key.
- Annotation value.
- Annotation note.

Parameters:

- **-i** Input file name

```
python manage.py import_config [-i] [--overwrite]
```

Imports settings from a config file (text file with key/value entries separated by “=” symbol)

Parameters:

- **-i** Input file name
- **–overwrite** If config parameter exists, overwrite its value (default: false)

```
python manage.py import_genome_descriptions [-i]
```

Updates descriptions of existing genome records and/or genome names. The input file must contain the following fields:

- Genome name (as in the database).
- New genome name (same as [1] if no change needed).
- Description text.
- External URL.
- External ID.

Parameters:

- **-i** Input file name

```
python manage.py import_genomes [-i]
```

Imports genomes from Genbank files. The files must be in the local file system. The input file must have six tab-separated fields:

- Full path to a Genbank file
- genome name (no spaces or special symbols)
- strain name (no spaces or special symbols, can be blank)
- sample name (no spaces, can be blank)
- External URL (link to NCBI genome assembly etc.)
- External identifier (for example, “NCBI:GCF_000006945.2”)

Parameters:

- **-i** Input file name

```
python manage.py import_regulons [-i]
```

Imports regulons from a tab-separated file. Genomes must be uploaded into the GenomeDepot database before running this command. The input file must contain the following fields:

- Regulon name.
- Genome name (as in the database).
- Regulatory gene locus_tag.
- Target gene locus_tag.
- Contig ID (as in the database).
- Site start.
- Site end.
- Site strand.
- Site sequence.

Parameters:

- **-i** Input file name

```
python manage.py import_sample_descriptions [-i]
```

For metagenomic samples uploaded into the Django database, this program changes sample descriptions. The input file must contain the following fields (tab-separated):

- Sample name (as in the database).
- Full name.
- Description text

Parameters:

- **-i** Input file name

```
python manage.py import_sample_metadata [-i]
```

Imports sample metadata from tab-separated file into the Django database. Sample records must be created before running this program. The input file must contain the following fields:

- Sample name (as in the database).
- Metadata source.
- Metadata URL (external).
- Metadata key.
- Metadata value.

Parameters:

- **-i** Input file name

```
python manage.py import_strain_metadata [-i]
```

Imports strain metadata records from Excel spreadsheet and from isolates.genomics.lbl.gov API. The spreadsheet must contain strain identifiers in the first column and names of metadata categories in the first row. All other non-empty cells will be considered values.

Parameters:

- **-i** Input file name

```
python manage.py recreate_genome_viewer [-g]
```

Deletes and re-creates static files for one genome.

Parameters:

- **-g** Genome name

```
python manage.py recreate_search_databases
```

Removes and re-generates nucleotide and protein search databases. Use it if the files are missing or corrupted, or if the genome import pipeline crashed before creating the search database files.

```
python manage.py run_annotation_pipeline [-i] [-t/--all]
```

For protein-coding genes from input genomes, this command runs one or more annotation tools. Existing annotations for these tools in the input genomes will be deleted. Input is either a file with list of genome names or a tab-separated file with six columns:

- path to Genbank file
- genome ID (no spaces)
- strain name (no spaces)
- sample ID (no spaces)
- External URL (link to NCBI genome assembly etc.)
- External ID (for example, "NCBI:GCF_000006945.2")

Parameters:

- **-i** Input file name
- **-t** Comma-separated list of annotation tool plugins
- **–all** Ignore -t parameter and run all annotation tools

```
python manage.py update_tags [-i] [-t]
```

Assigns one or more tags to genomes listed in a text or a tab-separated file. For the format of the input file, see description of the run_annotation_pipeline command.

Parameters:

- **-i** Input file name
- **-t** Comma-separated list of tags

```
python manage.py update_taxonomy
```

Updates taxonomy with new data downloaded from NCBI.

Developer guide

Introduction

GenomeDepot is based on the Django framework. Django architecture follows the Model-View-Template pattern, where Model handles the data representation, Template handles the presentation logic and View handles the business logic that takes up information from the user and displays model data to the user. GenomeDepot communicates with the web server over the Web Server Gateway Interface (WSGI), so Django views in GenomeDepot are synchronous in nature. However, there are several pages in GenomeDepot that employ asynchronous Javascript: BLAST search, database search in genes and annotations, comparative analyses, data export. In addition, long-running tasks started from the GenomeDepot administration panel, such as genome import or re-building BLAST databases, are executed asynchronously by the Django Q worker. Data models are defined in genomebrowser/browser/models.py.

Where GenomeDepot stores sequence data

In GenomeDepot, only protein sequences are stored in the MySQL database. Nucleotide sequences are always stored in static files. STATIC_ROOT variable in the .env file defines the directory for static files. This directory has the “genomes” subdirectory for genome files. The genomes/gbff directory contains genomes in the Genbank format and the genomes/jbrowse directory contains files served by the embedded genome viewer (indexed FASTA file, indexed GFF3 files etc.) for each genome. The web server must have access to static files. BLAST databases are stored in the appdata subdirectory of the project directory.

Genome import pipeline

GenomeDepot imports genomes in batches. Since each run of the genome import pipeline re-generates BLAST databases and runs eggNOG-mapper, single genome import is impractical. A reasonable size of a batch for import is between 50 and 1000 genomes.

The genome import pipeline runs several external tools generating ortholog family mappings, predicted operons and gene functional assignments. The first of them is eggNOG-mapper for fast functional annotation and orthology predictions. It processes all proteins from the imported genomes in chunks containing 200,000 sequences. GenomeDepot relies on eggNOG-mapper orthology predictions in comparative analyses and functional classifications. After the import of eggNOG-mapper output into database, the pipeline runs POEM for operon predictions, for one genome at a time. After importing operon predictions, the pipeline generates static files for the embedded Jbrowse browser and BLAST databases. At this moment, imported genomes become visible to web portal visitors. And finally, the genome import pipeline starts annotation pipeline, which runs an array of annotation tools.

Annotation pipeline

The GenomeDepot annotation pipeline is an extensible toolkit of specialized annotation tools. Each tool in the pipeline can be configured and turned on or off in the administration panel. The annotation pipeline can be started as a part of the genome import process or independently for genomes that have been imported into GenomeDepot. The annotation pipeline runs gene annotation tools one by one for selected genomes, one genome at a time.

Architecture of annotation pipeline plug-ins

The annotation pipeline of GenomeDepot can be expanded with additional annotation tools. The annotation tools accept nucleotide or protein sequences as an input and generate functional annotations for individual genes. GenomeDepot interacts with annotation tools through plug-in Python modules that organize input files, execute external tools, process tool-specific outputs and generate tab-separated files imported by the GenomeDepot pipeline into the database.

A plug-in module must contain the application function that accepts two arguments. The first argument is an object of the Annotator class, and the second is a dictionary of genomes, with genome name as a key and a path to the GenBank file as a value. The application function returns full path of tab-separated text file with gene annotations.

Typically, a plug-in module implements three functions: preprocess, run and postprocess. The preprocess function creates a working directory in the GenomeDepot temporary directory and writes all input files into the working directory. The preprocess function also generates bash script that activates a Conda environment for the annotation tool, executes the tool and deactivates the Conda environment. The run function executes the bash script created by preprocess. The postprocess function reads output file(s) generated by the annotation tool and creates an output file for import into GenomeDepot in the temporary directory. Finally, the postprocess function deletes the working directory.

Example:

```
from subprocess import Popen, PIPE, CalledProcessError
def application(genomes, annotator):
    wrapper_script, working_dir = preprocess(genomes, annotator)
    run(wrapper_script)
    output_filename = postprocess(working_dir)
    return output_filename
def preprocess(genomes, annotator):
    # create working directory
    # create input file
    # create wrapper bash script for a tool
    return wrapper_script, working_dir
def run(wrapper_script):
    with Popen([‘bash’, wrapper_script], stdout=PIPE, bufsize=1,
universal_newlines=True) as proc:
        for line in proc.stdout:
            print(line, end='')
    if proc.returncode != 0:
        raise CalledProcessError(proc.returncode, proc.args)
def postprocess(working_dir)
    # read tool output from working_dir
    # write annotations into output_file in temporary directory
    # delete working directory
    return output_file
```

Plug-in file name and location

Plug-in module files have to be placed into the genomebrowser/browser/pipeline/plugins directory, and file name should start with “genomedepot-“. Fo example, if a tool name is mytool, the plug-in file name is genomebrowser/browser/pipeline/plugins/genomedepot-mytool.py.

Annotation tool installation

Genome annotation tools for the GenomeDepot annotation pipeline have to be installed in separate conda environments. A name for a tool must start with “genomedepot-” (for example, genomedepot-mytool) to avoid conflicts with existing conda environments.

If there is no conda package for an annotation tool, the tools should be installed in a subdirectory of genomedepot/external_tools (for example, genomedepot/external_tools/mytool). Reference data files should be copied into a subdirectory of genomedepot/external_refdata (for example, genomedepot/external_refdata/mytool). Paths to executable files and reference data of the tool can be stored in GenomeDepot configuration parameters.

Plug-in configuration

Plug-in configuration parameters must include a name matching the name of the plug-in file. All plug-in parameters start with “plugins.” follwed by the tool name. For example, for the genomedepot-mytool.py module, configuration parameters start with “plugins.mytool.”.

Annotation tool plug-ins can be enabled or disabled in the GenomeDepot administration portal. To enable a plug-in, add or edit the ***“plugins..enable”** parameter in the Configuration page (admin/browser/config/) and set its value to 1. To disable a plug-in, set the value to 0.

Other common plug-in configuration parameters are:

- **display_name:** the plug-in name displayed in the administration portal pages (for example, parameter plugins.mytool.display_name defines the name of mytool in the “Run annotation tools” page)
- **conda_env:** the name of Conda environment where the tool is installed. Usually, conda environments are called genomedepot- (for example, "genomedepot-mytool") to avoid conflicts with existing conda environments.
- **threads:** number of threads for execution of the tool

Additional configuration parameters may be used to store location of reference data files for the tool, threshold values etc.

The annotation pipeline calls the application function of a plug-in module passing two arguments. The second argument is an object of the Annotator class called annotator, which has the annotator.config dictionary with all configuration parameters.

For example, annotator.config[‘plugins.amrfinder.threads’] stores a number of threads available for AMRFinderPlus (as a string).

Writing input files with nucleotide and protein sequences for annotation tools

Many annotation tools accept input sequence files in one of standard file formats. GenomeDepot has utility functions for export genome and protein sequences in FASTA and GenBank formats in the browser.pipeline.util module:

- **export_proteins**: exports proteins from one or more genomes into a single FASTA file. Arguments: list of protein identifiers, output file name
- **export_proteins_bygenome**: exports proteins from one or more genomes into FASTA files, one file per genome. Arguments: dictionary of genome names and GenBank file paths, output directory
- **export_nucl_bygenome**: exports genome sequences into FASTA files. Arguments: dictionary of genome names and GenBank file paths, output directory

Example:

```
from pathlib import Path
from browser.models import Genome
from browser.pipeline.util import export_proteins, export_nucl_bygenome
```

```
# Exports all proteins from all genomes into the proteins.faa file
output_fasta_file = 'proteins.faa'
genome_ids = Genome.objects.values_list('id', flat=True)
export_proteins(genome_ids, output_fasta_file)

# Exports nucleotide sequences of all genomes into the /tmp/fna directory
output_dir = '/tmp/fna'
Path(output_dir).mkdir(exist_ok=True)
genomes = Genome.objects.values_list('name', 'gbk_filepath')
genome_data = {x[0]:x[1] for x in genomes}
export_nucl_bygenome(genome_data, output_dir)
```

Temporary files

A temporary directory is stored in configuration parameter *core.temp_dir*. All temporary files should be kept in the temporary directory. It is highly recommended to create a working directory for each run of an annotation tool inside the temporary directory, and remove the working directory when the output files are no longer needed.

Example:

```
from pathlib import Path

def application(genomes, annotator):
        preprocess(genomes, annotator)

def preprocess(genomes, annotator):
        tool_name = 'mytool'
        temp_dir = annotator.config['core.temp_dir']
        working_dir = os.path.join(temp_dir, tool_name)
        Path(working_dir).mkdir(parents=True,exist_ok=True)
```

Plug-in output file

The application function of a plug-in module must return an absolute path for a tab-separated text file with gene annotations. The file must contain seven columns:

- Gene locus tag (as in the database).
- Genome name (as in the database).
- Annotation source (name of a tool producing the annotation, a database, etc. Up to 30 symbols long).
- Annotation URL (link to external resource. Up to 300 symbols long).
- Annotation key (Short category name, like “Protein family” or “Domain”. Up to 30 symbols long).
- Annotation value (For example, identifier of a protein family or domain. Up to 50 symbols long).
- Annotation note (free-text description).

GenomeDepot annotation pipeline imports gene annotations into the database and links them to genes with locus tag and genome name from the first and second columns. Lines starting with # are ignored.

License

Project maintained by aekazakov