

---

# **cob Documentation**

***Release 0.9.7***

**Joseph Jeffers, Rob Schaefer**

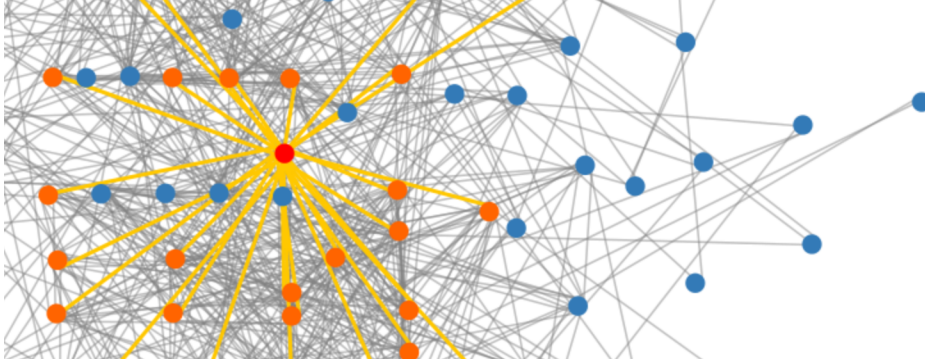
**Jan 29, 2020**



**SECTIONS:**

<b>1</b>	<b>Table of Contents</b>	<b>3</b>
1.1	Installation . . . . .	3
1.2	Running COB . . . . .	3
1.3	Configuration Files and Data Directories . . . . .	4
1.4	Docker . . . . .	6
1.5	Deploying . . . . .	6
<b>2</b>	<b>Paper</b>	<b>7</b>
<b>3</b>	<b>Acknowledgements</b>	<b>9</b>
<b>4</b>	<b>Funding</b>	<b>11</b>





COB is a complete client/server package built to browse gene co-expression networks created by [Camoco](#). The client is written in javascript, and the server is written in python.

A demo for COB is running at <http://lovelace.cs.umn.edu/cob>.



## TABLE OF CONTENTS

### 1.1 Installation

This package is entirely dependent on [Camoco](#). It is designed such that once the Camoco has been installed, COB can be added by, inside the camoco virtual environment, running:

```
$ pip install camoco-cob
```

### 1.2 Running COB

Once installed, COB has a convenient command line interface to manage the server. To run it with all of the default options, no options are required, just execute in the camoco virtual environment:

```
$ cob
```

This will start the server in the current terminal window. To see the site, navigate to <http://localhost:50000> in your web browser once the server has finished loading. To terminate the server, press *Ctrl+C* in the same terminal window. To run the server in the background, add the *-d* flag to the start command. To terminate all instances of the COB server, run *cob -k*. To define a specific server to kill, add the *-n* flag followed by the name of the server as such:

```
$ cob -k -n my_server
```

To use a specific configuration file for server settings, the file may be defined with the *-c* flag:

```
$ cob -c my_server.conf
```

If no configuration file is defined, COB then checks for a section *web* in the main camoco configuration file *~/camoco.conf*. If there are no settings in that file, it will load with default values. The full configuration options are discussed in the next section.

This is the full documentation for all *cob* CLI options, which can also be accessed by executing *cob -h*:

```
$ cob -h

usage: cob [-h] [-c USERCONF] [-d] [-k] [-l] [-n NAME]

Manage instances of the COB server.

optional arguments:
  -h, --help            show this help message and exit
  -c USERCONF, --config USERCONF
```

(continues on next page)

(continued from previous page)

	Provide a YAML formatted configuration file, if not provided, general Camoco config file is used.
<code>-d, --daemon</code>	Run gunicorn as a daemon (allows closing of this terminal).
<code>-k, --kill</code>	Kill running server. Use '-n' to define specific server to kill otherwise all will be.
<code>-l, --list</code>	Kill running server. Use '-n' to define specific server to kill otherwise all will be.
<code>-n NAME, --name NAME</code>	Name of server to start or kill.

## 1.3 Configuration Files and Data Directories

- `/home/camoco/camoco`

This is the storage directory for all camoco related data. This can be kept in the container, but it will not be persistent then. It is highly recommended to mount a directory for this.

- `/home/camoco/cob.conf`

This is the location of the cob configuration file that is used by cob by default. This will not work by default given no data is included in this docker image. Make sure to build your configuration based off your data and mount it here.

- `/home/camoco/camoco.conf`

This is the location of the camoco configuration file. By default the enable camoco config file in this repo is used. This should work fine for most use cases.

### 1.3.1 Configuration

A powerful configuration engine is provided to both set options for the server and also options for the content for the website, such as default option values. As mentioned above, these can either be provided through a standalone YAML configuration file `example.conf` is included in the repo) or in a section with the same format titled `web` in the main camoco configuration file (found at `~/camoco.conf`). One need not include one at all, this will just be started with the default values (seen below). This will also trigger all available Camoco networks and GWAS datasets to be loaded in. To prevent this, one may specify the desired datasets. The following is an annotated version of the default settings, showing all the potential configuration options:

### 1.3.2 Server Options

```
name: cob           # The name of this server instance, must be unique for
                    # each instance, can be overridden by '-n' flag
port: 50000         # Port to which the server will be attached
host: localhost    # The allowed hosts that can communicate with this server
                    # (must be 0.0.0.0 with docker or to allow external connections)
threads: 8          # How many individual threads the sever process may use
timeout: 500        # How long a thread maybe unresponsive before termination
dev: False         # Forces JS and CSS to be recompiled on every request
                    # Normally done only on server restart
```



### 1.3.3 Datasets

```
networks:           # Camoco networks that are to be loaded in the server.
- My_Network_1      # If this is not included, all available Camoco
- My_Network_2      # networks will be loaded.
gwas:               # GWAS datasets that will be loaded in the server. If
- My_GWAS_1         # this is not included, all GWAS datasets that
                    # correspond to loaded networks will be loaded.
```

### 1.3.4 Default Values

```
defaults:           # This is the dictionary containing all of the defaults
                    # for the options on the web site
logSpacing: True    # Spacing of genes in Polywas layout, log or true distance
visEnrich: True     # Only enrich genes visible on graph or all in table
fdrFilter: True     # Whether to use FDR to filter query results
nodeCutoff: 1       # How many edges a node must have to be visible
edgeCutoff: 3.0     # The cutoff for significance of edge scores
fdrCutoff: 0.35     # If the FDR Filter is used, the cutoff for being visible
windowSize: 50000   # Window size used in the query
flankLimit: 2       # Flank limit used in the query
visNeighbors: 25    # Default number of neighbors visible in custom network
nodeSize: 10        # Size of the nodes on the graph
pCutoff: 0.05       # P value cutoff for enrichment queries
minTerm: 5          # Minimum number of genes a GO term must have to be included
maxTerm: 300        # Maximum number of genes a GO term must have to be included
```

### 1.3.5 Reference Links

This section allows for linking directly from genes to an external website for more information. This can be configured for each different reference genome (RefGen) used to build the included networks. If not included, the option won't appear. To configure this, start by writing the name of the RefGen under the *refLinks* option, followed by a colon and a space as seen below. Then you must go to the database you wish to use for that RefGen, and search any gene. After finding this, copy the URL onto the line after the name of the RefGen. Finally replace the name of the gene in the URL with the string *{id}*. This allows the website to find where in the URL the gene name goes, and replace it with any gene for that organism. The following example works for maize, soybean, and medicago. Add or subtract species at will.

```
refLinks:
  Zm5bFGS: http://www.maizegdb.org/gene_center/gene/{id}
  Gmax_a2_V1: https://www.soybase.org/sbt/search/search_results.php?
    ↪category=FeatureName&version=Glyma2.0&search_term={id}
  Mt_4.0: http://medicago.jcvi.org/cgi-bin/medicago/manatee/shared/ORF_infopage.cgi?
    ↪db=mta4&user=access&password=access&identifier=locus&orf={id}
```

## 1.4 Docker

For your convenience, a demonstration version of COB is available to run as a Docker image with sample data. The data and scripts used to build this image are available as a [repository](#) as an example. The built image is also available on [Docker Hub](#) using the tag *maize-data*. Running the Docker image does not require any special configuration since it already has all of the data built and included. It can be run as follows:

```
$ docker pull linkageio/camoco-cob
$ docker run --name cob -p 50000:50000
```

This image can be seen running at <http://lovelace.cs.umn.edu/cob>. This demonstration server is provided as-is, and is not guaranteed to be maintained indefinitely. The Docker image is the preferred method to use this demo version.

Included in this repo there is a Dockerfile which contains camoco and cob in a container. It does not provide any data prebuilt, thus in order to use this, three different mounts are available which are explained below. To run the image you would want to use a command like this:

```
$ docker run -it --rm --name cob \
  --volume $HOME/.camoco:/home/camoco/.camoco
  --volume $HOME/cob.conf:/home/camoco/cob.conf
  --publish 50000:50000
  linkageio/camoco-cob
```

This will start cob based on the configuration and data provided on localhost port 50000. When doing this, it is important that you change the *host* configuration value in the cob config to *0.0.0.0*. If you wish to restrict access by IP, do so using docker arguments instead. To enter the container, just add *bash* to the end of the command, or enter a running container using *exec* with *bash*.

## 1.5 Deploying

If you care to make this site accessible to the web, you can add a reverse proxy to Apache, allowing for access by using a normal URL. An example of how to do this is provided here, but for more detailed documentation, see the [Apache docs](#).

```
<VirtualHost *:80>
    ProxyPass /cob http://127.0.0.1:50000
    ProxyPassReverse /cob/ http://127.0.0.1:50000
</VirtualHost>
```

The equivalent can be done in NGINX using the *proxy\_pass* directive.

---

## CHAPTER TWO

---

### PAPER

There is a publication for this updated version of cob in process. More information will be added to this doc when it's available.



## ACKNOWLEDGEMENTS

I would like to thank all members of the Myers Lab for their extensive and patient mentorship. I would also like to thank Max Franz, for all of his work on [Cytoscape.js](#), and assistance in getting it integrated optimally. Finally I would like to thank the open source web development community in general. This project was only possible because of the robust ecosystem of packages available for use by everyone.



**FUNDING**

J.J., R.S., and J.M. were partially supported by a Biomedical Informatics and Computational Biology (BICB) Fellowship. This work was supported by a grant from the National Science Foundation to N.M.S. (DBI-1237931) and by US Department of Agriculture Hatch funds to N.M.S. and C.L.M. R.S., R.B. and C.L.M. were partially supported by grants from the National Science Foundation (IOS 1126950 and DBI 0953881). C.L.M. is supported by the CIFAR Genetic Networks Program. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.