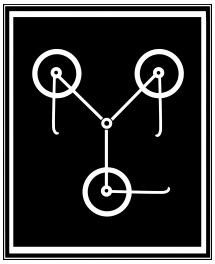
Flux Capacitor Toolkit for Systems Biology



User Manual

Daniel Ortiz Martínez
daniel.ortiz.phd@gmail.com

December 2017

CONTENTS

CHAPTER 1

INTRODUCTION

Flux Capacitor is an open source software toolkit for systems biology and flux balance analysis. Flux Capacitor is focused on the application of FBA to study metabolism.

1.1 Toolkit Features

The toolkit includes the following features:

- FBA implementation for metabolic models in SBML format.
- Implementation of tissue specific FBA (Shlomi et al. 2008).
- Flux Variability Analysis (FVA) implementation.
- Techniques to accelerate FVA calculations including parallel execution.
- Automated techniques to generate metabolic network representations.
- Metabolic network reduction methods.

1.2 Distribution Details

Flux Capacitor has been coded using C, C++, Python, R and shell scripting. Flux Capacitor is known to compile on Unix-like and Windows (using Cygwin) systems. As future work we plan to port the code to other platforms. See Section 1.4 section of this file if you experience problems during compilation.

It is released under the GNU Lesser General Public License (LGPL)[a].

[a] http://www.gnu.org/copyleft/lgpl.html

1.3 Current Status

The Flux Capacitor toolkit is under development. Basic usage instructions are being added. In addition to this, there are some toolkit extensions currently in preparation:

- Incorporate interactive Python mode (currently the toolkit offers a command-line interface).
- Enable use of alternative mathematical solvers for FBA and FVA (current version only supports CPLEX).

1.4 Documentation and Support

Project documentation is being developed. Such documentation include:

- Flux Capacitor website[b].
- The Flux Capacitor manual (`flux_capacitor_manual.pdf` under the `doc` directory).
- Daniel Ortiz' MSc thesis in bioinformatics[c]: the thesis explains the theoretical foundations of the functionality implemented in the toolkit and uses it to report results.
- MSc thesis slides[d]: provide a summary of the above mentioned MSc thesis.

If you need additional help, you can:

- use the github issue tracker[e].
- send an e-mail to the author[f].

[b] http://daormar.github.io/flux-capacitor/
[c] https://daormar.github.io/flux-capacitor/docsupport/dortiz_bio_msc_thesis.pdf
[d] https://daormar.github.io/flux-capacitor/docsupport/dortiz_bio_msc_thesis_slides.pdf
[e] https://github.com/daormar/flux-capacitor/issues
[f] daniel.ortiz.researcher@gmail.com

CHAPTER 2

INSTALLATION

2.1 Basic Installation Procedure

The code of the Flux Capacitor toolkit is hosted on github[a]. To install Flux Capacitor, first you need to install the autotools (autoconf, autoconf-archive, automake and libtool packages in Ubuntu). If you are planning to use Flux Capacitor on a Windows platform, you also need to install the Cygwin environment[b]. Alternatively, Flux Capacitor can also be installed on Mac OS X systems using MacPorts[c].

On the other hand, Flux Capacitor can be combined with third party software so as to enable extended functionality, see more information in Section 2.2.

Once the autotools are available (as well as other required software such as Cygwin, MacPorts), the user can proceed with the installation of Flux Capacitor by following the next sequence of steps:

1. Obtain the package using git:

```
$ git clone https://github.com/daormar/flux-capacitor.git
```

   Additionally, Flux Capacitor can be downloaded in a zip file[d].

2. `cd` to the directory containing the package's source code and type `./reconf`.

3. Type `./configure` to configure the package.

4. Type `make` to compile the package.

[a] https://github.com/daormar/flux-capacitor/
[b] https://www.cygwin.com/
[c] https://www.macports.org/
[d] https://github.com/daormar/flux-capacitor/archive/master.zip

5. Type `make install` to install the programs and any data files and documentation.
6. You can remove the program binaries and object files from the source code directory by typing `make clean`.

By default the files are installed under the `/usr/local` directory (or similar, depending of the OS you use); however, since Step 5 requires root privileges, another directory can be specified during Step 3 by typing:

```
$ configure --prefix=<absolute-installation-path>
```

For example, if `user1` wants to install the Flux Capacitor package in the directory `/home/user1/flux-capacitor`, the sequence of commands to execute should be the following:

```
$ make clean # This is recommended if the package has already been built
$ ./reconf
$ configure --prefix=/home/user1/flux-capacitor
$ make
$ make install
```

The installation process also creates three directories with additional information:

- **${PREFIX}/share/flux-capacitor/cfg_templates**: contains configuration files to be used with different Flux Capacitor utilities (see Chapter 3 for more details).
- **${PREFIX}/share/flux-capacitor/doc**: contains the documentation of Flux Capacitor, which currently consists in the Flux Capacitor manual (`flux_capacitor_manual.pdf`).

IMPORTANT NOTE: if Flux Capacitor is being installed in a PBS cluster (a cluster providing `qsub` and other related tools), it is important that the `configure` script is executed in the main cluster node, so as to properly detect the cluster configuration (do not execute it in an interactive session).

2.2 Third Party Software

Flux Capacitor requires certain third party software packages to take advantage of its whole functionality. Below we summarize the details of such packages.

2.2.1 CPLEX

Flux Capacitor internally uses CPLEX[e] as a mathematical solver to obtain the solutions required by FBA and FVA procedures. Therefore, users also need to install this package to be able to access most of the functionality of the toolkit.

2.2.2 Graphviz

Graphviz[f] open source graph visualization software is used to elaborate graphical representations of metabolic networks.

2.2.3 Python Modules

Flux Capacitor uses some Python modules to obtain results. Below we enumerate such modules:

- **SciPy**[g]: this package is used to determine whether a gene is expressed or not when working with RNA-Seq data.
- **scikit-learn**[h]: scikit-learn is used to perform statistical hypothesis testing.
- **StatsModels**[i]: similarly to scikit-learn, this package is also useful for hypothesis testing.
- **libSBML**[j]: this library is used to access to the information contained in the Recon X human metabolic reconstruction (Thiele et al. 2013).

2.2.4 Bioconductor

Flux Capacitor uses packages provided by the very well known Bioconductor software project[k] for the analysis and comprehension of genomic data. Bioconductor packages are used in Flux Capacitor to deal with microarray data.

2.3 Add Flux Capacitor to the System PATH

To end the installation process, it might be useful to add Flux Capacitor to the system PATH. This will allow us to easily execute commands provided in the package without the necessity of providing the whole Flux Capacitor installation path.

For this purpose, we can execute the following commands:

[e] https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/
[f] https://www.graphviz.org/
[g] https://www.scipy.org/
[h] http://scikit-learn.org/
[i] http://www.statsmodels.org/stable/index.html
[j] http://sbml.org/Software/libSBML/docs/python-api/
[k] https://www.bioconductor.org/

```
$ FCAP_HOME_DIR=<absolute-installation-path>
$ export PATH=$PATH:${FCAP_HOME_DIR}/bin
```

These variable definitions can be added to the `.bashrc` user profile file, so as to define them automatically whenever a new interactive shell session is started.

CHAPTER 3
USER GUIDE

This chapter provides usage information for the Flux Capacitor toolkit. Chapter content is organized as follows: first, a toolkit overview is given in Section 3.1, including the provided functionality as well as a general vision of the implemented bioinformatic pipeline. Next, the main software tools incorporated in the toolkit are explained in Section 3.2. Finally, a detailed description of how to execute the bioinformatic pipeline provided by Flux Capacitor is given in Section 3.4.

To better understand the technical background of the toolkit, it is highly recommended that the reader consults the work presented in (Ortiz-Martínez 2016).

3.1 Toolkit Overview

In this section we introduce the main functionality of Flux Capacitor as well as the bioinformatic pipeline it executes.

3.1.1 Main Functionality

Flux Capacitor is focused on the application of flux balance analysis (FBA) (Fell and Small 1986) techniques to study metabolism. The toolkit is currently under development. Below there is a list of its main functionalities:

- **FBA**: the toolkit implements FBA for metabolic models given in SBML format, maximizing the biomass function and returning the flux values that correspond to the optimal solution.
- **FVA**: a parallel version of the so-called flux variability analysis (FVA) procedure (Mahadevan and Schilling 2003) is included. Flux Capacitor also incorporates the techniques proposed in (Gudmundsson and Thiele 2010) to accelerate the calculations.
- **Tissue-specific FBA**: the package provides an implementation of the tissue-specific FBA procedure proposed by Shlomi et al. 2008. The application of tissue-specific FBA requires the generation of lists of lowly and highly expressed reactions. The

procedure needed to obtain such lists depends on whether the gene expression data comes from a microarray experiment or from an RNA-Seq experiment. An overview of the steps required for both procedures is explained in (Ortiz-Martínez 2016). The two procedures are implemented in Flux Capacitor.

- **Statistical testing**: Flux Capacitor allows to apply statistical hypothesis tests for case/-control samples. In particular, the t-test and the Mann-Whitney's U-test can be executed.
- **Network visualization**: the package allows to generate automated graphical representations of metabolic networks in SBML format. For this purpose, the open-source graph visualization tool called Graphviz (Gansner and North 2000) is used.
- **Network reduction**: Flux Capacitor includes an implementation of the *NetworkReducer* algorithm proposed by Erdrich et al. 2015 as well as a fast version of it specifically implemented in this toolkit (Ortiz-Martínez 2016).

3.1.2 Bioinformatic Pipeline

The toolkit functionality introduced in the previous section is used to execute a whole bioinformatic pipeline whose purpose is to study cancer metabolism by means of systems biology techniques.

Figure 3.1 shows a diagram representing the different steps that compose the pipeline. For each step the diagram also incorporates information regarding the methods that are applied as well the external software that is used (if any).

3.2 Main Tools

The functionality of Flux Capacitor is provided by means of a set of tools executing modular tasks. Next, we provide a list of the most important of such tools, briefly describing the input parameters they expect as well as their dependencies with other software:

- **`extract_sbml_model_info`**: extracts information from a metabolic model in SBML format. The program takes as input a file in SBML format and generates a list of text files with varied information (reaction and metabolite names, stoichiometric matrix, etc.). It is implemented in R.
- **`auto_fba`**: automates an FBA procedure. The tool receives as input the name of the SBML file containing the metabolic model and the type of optimization to be computed: biomass function or tissue-specific. If tissue-specific FBA is to be applied, then the program requires transcriptomic information, that can be provided as a set of CEL files for microarray data or as a file with RNA-Seq counts. `auto_fba` is implemented as a UNIX shell script.
- **`auto_fva`**: automates a whole FVA procedure. The program takes as input the prefix of the files in `lp` format representing the initial FBA problem to be solved (they are

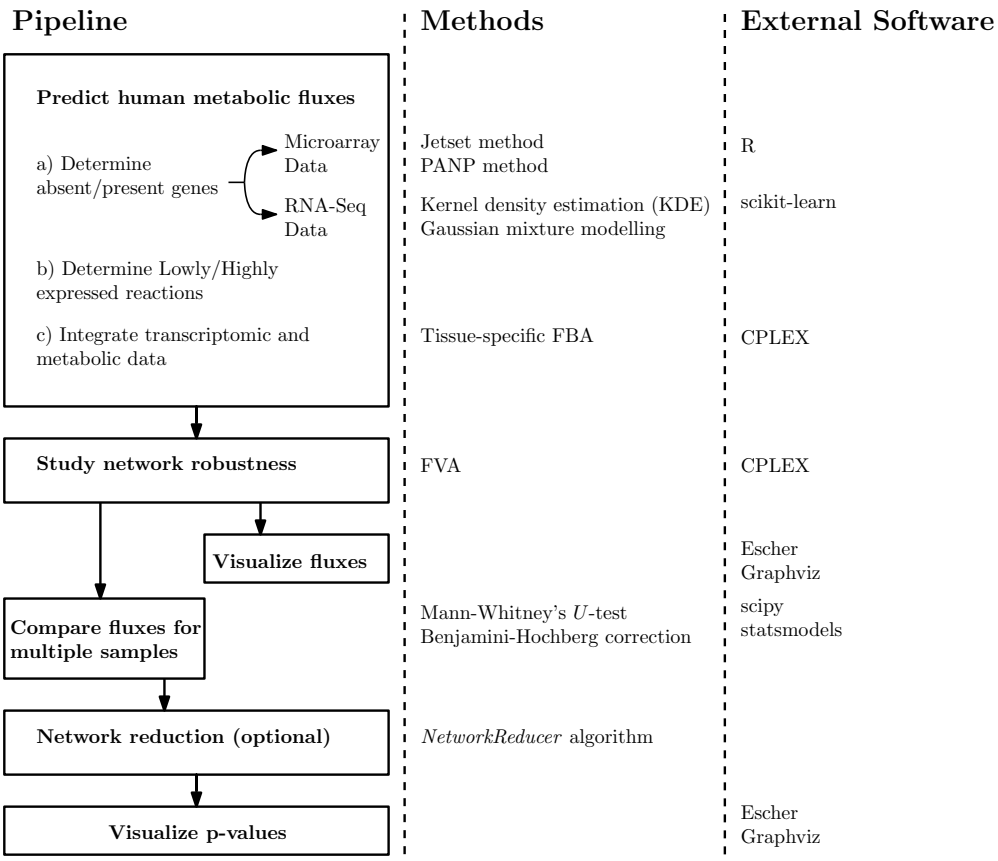


Figure 3.1: Overview of the bioinformatic pipeline implemented by the Flux Capacitor toolkit.

obtained by means of the `auto_fba` tool). In addition to this, `auto_fva` also takes additional parameters to control process efficiency. This tool is implemented as a UNIX shell script.

- **`test_samples`**: performs statistical tests for a set of samples classified into cases and controls. The tool expects as input a CSV file with the sample data and another one with the phenotype data. `test_samples` is a Python program using the `scipy` and the `statsmodels` modules.

- **`correct_pvalues`**: corrects a set of p-values using the Benjamini-Hochberg procedure. It receives as input a file with p-values generated by means of `test_samples` and the value of α. The tool is written in Python and uses the `statsmodels` module.

- **`plot_metab_network`**: generates files in Graphviz format representing metabolic networks. Such files can later be converted to graphics files in different formats. The tool

takes as input the plot type to be generated, the prefix of a series of files representing the metabolic network generated with the `extract_sbml_model_info` tool, a file containing the identifiers of the reactions to be included in the plot, another file with data about the reactions (e.g. flux values, p-values) and optionally, a list of identifiers of external metabolites. `plot_metab_network` is written in Python.

- **`network_reducer`**: reduces the number of elements of a metabolic network. It is designed to work with the output of the `auto_fba` tool. `network_reducer` is a UNIX shell script.

All of the tools included in the package can display help messages describing their expected input parameters.

3.3 Relevant Data

The techniques and tools implemented by the Flux Capacitor toolkit require data to work properly. In this section we briefly discuss the details and location of some data sources that are later used in this manual to illustrate how the toolkit works.

3.3.1 Human Metabolic Reconstruction

Flux Capacitor works with the reconstruction of human metabolism in SBML format provided by the Recon 2 metabolic model (Thiele et al. 2013). Recon 2 exists in different versions, for testing purposes we will work with version 2.03, which can be freely downloaded from the webpage of the Biomodels Database[a].

3.3.2 Sample Data

Flux Capacitor provides some sample data useful to test the toolkit functionality. In particular, such data includes:

- **Microarray data**[b]: a subset of 6 microarray data samples belonging to the GSE40595 Series[c] of the Gene Expression Omnibus public repository for high throughput data[d]. The GSE40595 Series is focused on the study of ovarian cancer. For the sample data, 3 samples of healthy cells and another 3 of cancerous cell were selected.

- **RNA-Seq data**[e]: RNA-seq data corresponding to a subset of the kidney renal clear cell carcinoma (KIRC) data collection[f], which is part of the TCGA database. In particular, the subset is composed of 60 samples coming from healthy cells and another 60 from cancerous cells so as to obtain a balanced experiment design.

[a] http://www.ebi.ac.uk/biomodels-main/MODEL1504130000
[b] https://daormar.github.io/flux-capacitor/sample_data/marray_data.tar.gz
[c] https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE40595
[d] https://www.ncbi.nlm.nih.gov/geo/
[e] https://daormar.github.io/flux-capacitor/sample_data/rnaseq_data.tar.gz
[f] ttps://gdc-portal.nci.nih.gov/projects/TCGA-KIRC

- **Auxiliary data for network visualization and reduction**[g]: is composed of a set of plain text files providing information useful during network and visualization processes. The details of such files are discussed during the step by step pipeline explanation provided in the next section.

3.4 Step by Step Pipeline Execution

In this section we will describe how to use the tools provided by the Flux Capacitor toolkit to execute the different steps of the bioinformatic pipeline described in Figure 3.1. Figure 3.2 graphically represents the process.

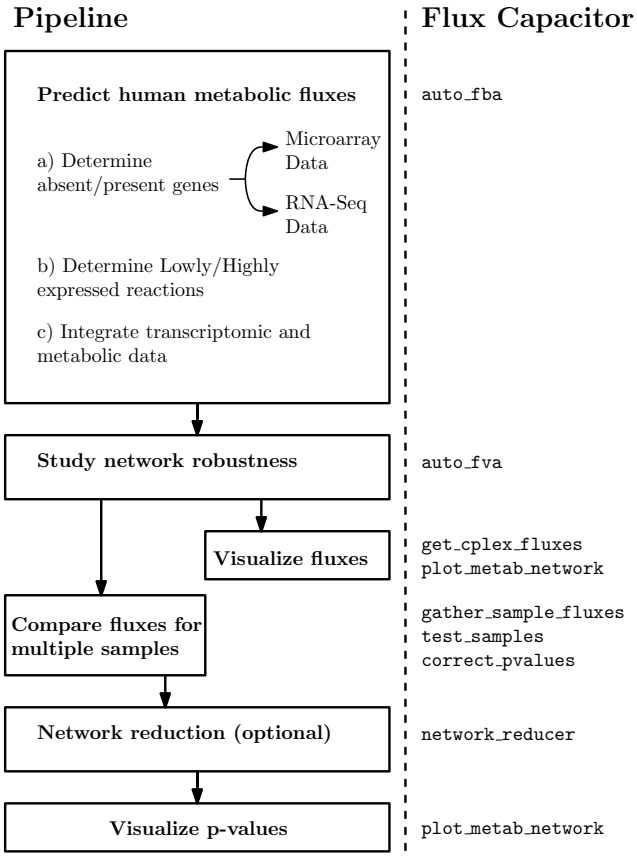


Figure 3.2: Overview of the Flux Capacitor tools used to execute the steps of the bioinformatic pipeline implemented by the toolkit.

In order to appropriately structure the information, we first create a specific working directory:

[g] https://daormar.github.io/flux-capacitor/sample_data/aux_net_data.tar.gz

```
mkdir working_dir
cd working_dir
```

After that, we download the human metabolic model mentioned in Section 3.3.1:

```
mkdir recon
wget -O recon/recon_2.03.xml \
     http://www.ebi.ac.uk/biomodels-main/download?mid=MODEL1504130000
```

In addition to this, we also download some auxiliary data that will be useful for network visualization and reduction (see Section 3.3.2):

```
wget https://daormar.github.io/flux-capacitor/sample_data/aux_net_data.tar.gz
tar -zxvf aux_net_data.tar.gz
```

3.4.1 Predict Human Metabolic Fluxes

The first step of the bioinformatic pipeline is to calculate human metabolic fluxes according to a given metabolic model using the so-called FBA method. For this purpose, Flux Capacitor implements the `auto_fba` tool.

Due to the fact that `auto_fba` internally uses the CPLEX solver to calculate metabolic fluxes, we need to define an environment variable providing the complete path to the directory storing the `cplex` binary:

```
export CPLEX_BINARY_DIR=<directory_where_cplex_binary_is_stored>
```

After defining the previous variable, we are ready to execute FBA procedures implemented by `auto_fba`. For instance, we can apply standard FBA by means of the following command:

```
auto_fba -m recon/recon_2.03.xml -c 0 -o standard_fba
```

The results are stored in the `standard_fba` folder.

However, the main interest of the Flux Capacitor toolkit is to apply tissue-specific FBA (Shlomi et al. 2008), where transcriptomic data is combined with metabolic information. The transcriptomic data that can be used at this step differ depending on whether it was produced by a microarray or an RNA-Seq experiment.

Microarray Data

Flux Capacitor allows to work with microarray data when applying tissue-specific FBA. Here we will work with a data set composed of six samples explained in Section 3.3.2. First of all, we download the data:

```
wget https://daormar.github.io/flux-capacitor/sample_data/marray_data.tar.gz
tar -zxvf marray_data.tar.gz
```

Once the data is available, we launch a complete tissue-specific FBA procedure by means of the following command:

```
auto_fba -m recon/recon_2.03.xml -c 1 -o shlomi_fba -d marray_data \
         -p marray_data/gse40595_subset_pdata.txt
```

Results are stored in the `shlomi_fba` directory.

RNA-Seq Data

The toolkit is also able to handle RNA-Seq data. Again, in this pipeline explanation we will work with a predefined data set previously mentioned in Section 3.3.2. We can obtain the data by executing:

```
wget https://daormar.github.io/flux-capacitor/sample_data/rnaseq_data.tar.gz
tar -zxvf rnaseq_data.tar.gz
```

After obtaining the data, we can perform tissue-specific FBA using the next command:

```
auto_fba -m recon/recon_2.03.xml -c 1 -o shlomi_fba \
         -r rnaseq_data/KIRC_120_sample_RPKM.txt \
         -p rnaseq_data/KIRC_120_sample_pdata_simplified.txt
```

Again, results are stored in `shlomi_fba` folder. To simplify calculations, we restrict to a subset of only 4 samples of the total set composed of 120 samples. This is achieved by providing the file `rnaseq_data/KIRC_120_sample_pdata_simplified.txt` as argument for the `-p` option.

3.4.2 Study Network Robustness

Once the flux values have been computed, we can study the robustness of the network using the so-called FVA method (Mahadevan and Schilling 2003). Flux Capacitor incorporates the `auto_fva` tool to perform this kind of studies.

FVA can be carried out for the tissue-specific FBA results of a particular sample. For instance, below we show the command required to apply this procedure for a particular microarray sample:

```
auto_fva -l shlomi_fba/lp/GSM997591_NS5.CEL.gz \
         -o fva_for_sample -g 0.8
```

Results are stored in the `fva_for_sample` folder.

FVA can also be applied to FBA results of a particular RNA-Seq sample:

```
auto_fva -l shlomi_fba/lp/TCGA.B0.4833.01A.01R.1305.07 \
         -o fva_for_sample -g 0.8
```

Again, directory `fva_for_sample` stores the results.

3.4.3 Visualize Fluxes

After analyzing network robustness, we can proceed to graphically represent fluxes. Due to the fact that current human metabolic reconstructions are composed of thousands of reactions, in order to obtain meaningful representations it is important to restrict the visualization to those reactions composing *metabolic subsystems* of interest.

Using Flux Capacitor we can visualize fluxes in two different ways, namely, by means of the Escher external application or by means of the Graphviz software package. First of all, we create a directory specific to store visualization-related information:

```
mkdir flux_visualiz
```

Escher Visualization

If we want to use Escher, first it is necessary to provide the flux information in a file format accepted by the application. One example of such file formats is the JSON (JavaScript Object Notation) format. When using microarray data, we can generate the required information for a particular sample by means of the `get_cplex_fluxes` tool:

```
get_cplex_fluxes -f shlomi_fba/sol/GSM997591_NS5.CEL.gz.sol \
                 -m shlomi_fba/minfo/model -of 1 \
                 > flux_visualiz/GSM997591_NS5_fluxes.json
```

The resulting file can be loaded in Escher, where we should previously choose the metabolic subsystem we want to observe. One example of such subsystems would be the so-called tricarboxylic acid (TCA) cycle, which in the application is identified with the label "*Glycolysis TCA PPP (RECON1)*".

We can proceed in the same way for a particular RNA-Seq sample by executing the following command:

```
get_cplex_fluxes -f shlomi_fba/sol/TCGA.B0.4833.01A.01R.1305.07.sol \
                 -m shlomi_fba/minfo/model -of 1 \
                 > flux_visualiz/TCGA.B0.4833.01A.01R.1305.07_fluxes.json
```

Graphviz Visualization

Alternatively, Flux Capacitor incorporates a more general visualization mechanism based on the Graphviz tool. The first required step is to generate a `csv` file with the metabolic fluxes information, in a very similar way as we explained above. Assuming that we want to represent fluxes for a particular microarray sample, the corresponding `csv` file can be generated as follows:

```
get_cplex_fluxes -f shlomi_fba/sol/GSM997591_NS5.CEL.gz.sol \
                 -m shlomi_fba/minfo/model -of 0 \
                 > flux_visualiz/GSM997591_NS5_fluxes.csv
```

The output `csv` file can be processed by the `plot_metab_network` provided by Flux Capacitor to generate a file in `gv` format, which is the one expected by Graphviz. The required command would be as follows:

```
plot_metab_network -s shlomi_fba/minfo/model \
                   -d flux_visualiz/GSM997591_NS5_fluxes.csv \
                   -f aux_net_data/tca_reactid.txt \
                   -e aux_net_data/tca_extern_metid.txt -t 2 \
                   > flux_visualiz/GSM997591_NS5_fluxes_tca.gv
```

In the previous command, we use the `aux_net_data/tca_reactid.txt` file (that was previously downloaded, see Section 3.4) in order to restrict network representation to those reactions that compose the TCA cycle.

After obtaining the `gv` file, we can generate a graphic representation by means of the `neato` tool provided by the Graphviz package:

```
neato -T pdf flux_visualiz/GSM997591_NS5_fluxes_tca.gv \
      > flux_visualiz/GSM997591_NS5_fluxes_tca.pdf
```

The steps required to represent information related to an RNA-Seq sample would be very similar to those explained above for microarray data, but changing the generation of the `csv` file when executing the `get_cplex_fluxes` tool.

3.4.4 Compare Fluxes for Multiple Samples

One of the main goals of the bioinformatic pipeline implemented by Flux Capacitor is to study differentially expressed reactions between healthy and cancerous cells. For this purpose we need to compare the previously computed fluxes for the cell samples included in the study.

First we create a directory to store information related to hypothesis testing:

```
mkdir hyp_testing
```

Second, we gather the fluxes for the different samples in one `csv` file by using the `gather_sample_fluxes` tool:

```
gather_sample_fluxes -d shlomi_fba > hyp_testing/shlomi_fba_sample_fluxes.csv
```

Third, we compute p-values using the Mann-Whitney's U-test by means of the `test_samples` tool:

```
test_samples -p rnaseq_data/gse40595_subset_pdata.txt \
             -s hyp_testing/shlomi_fba_sample_fluxes.csv -u \
             > hyp_testing/shlomi_fba_sample_fluxes_pval.csv
```

Finally, we can correct the p-values previously computed by means of the `correct_pvalues` command, which applies the so-called Benjamini-Hochberg correction:

```
correct_pvalues -p hyp_testing/shlomi_fba_sample_fluxes_pval.csv \
                > hyp_testing/shlomi_fba_sample_fluxes_pval_corrected.csv
```

3.4.5 Network Reduction

One possible (and optional) strategy to obtain network representations that are easier to understand is to reduce the size of the network by removing some of its elements according to certain criteria. Flux Capacitor incorporates this functionality in the `network_reducer` tool. `network_reducer` implements the algorithm of the same name proposed in (Erdrich et al. 2015), incorporating some modifications to speed up calculations, so as to allow its use with human metabolic networks, which are larger than those used in the paper to test the algorithm. The algorithm works by iteratively removing reactions from the input network, providing that certain constraints are met. For instance, the algorithm accepts a list of *protected reactions* that cannot be removed during the reduction process.

The following command line executes a network reduction process, creating a new network that is stored in the `reduced_network` folder:

```
network_reducer -a standard_fba -lpm aux_net_data/empty_protected_metabid.txt \
                -lpr aux_net_data/retinol_reactid.txt \
                -md 1000 -mr 1000 -o reduced_network -li 10 -g 0.9
```

In the previous example, `network_reducer` is instructed to protect the reactions related to the retinol metabolism by means of the `aux_net_data/retinol_reactid.txt` file provided as argument of the `-lpr` option.

3.4.6 Visualize p-values

The last pipeline step consists in visualizing the p-values for the differential reaction expression experiment previously executed. For this purpose, we can work with the original metabolic network or with the reduced one obtained in the previous section.

First of all, we create a specific directory where we will store information related to p-value visualization:

```
mkdir pval_visualiz
```

Next, we use the `plot_metab_network` tool to generate a file in `gv` format, which is the format expected by Graphviz:

```
plot_metab_network -s reduced_network/model \
                   -d hyp_testing/shlomi_fba_sample_fluxes_pval_corrected.csv \
                   -f aux_net_data/retinol_reactid.txt \
                   -e aux_net_data/retinol_extern_metid -t 5 \
                   > pval_visualiz/diff_react_expression_pval_corrected.gv
```

Again, in the previous example, we have used the `-f` option provided by `plot_metab_network` to focus the graphical representation in a certain metabolic subsystem. In particular, such subsystem corresponds to the retinol metabolism, which is composed by the list of reactions given in the `aux_net_data/retinol_reactid.txt` file.

Finally, we obtain a graphical representation of the `gv` file by using the `neato` tool included in Graphviz:

```
neato -T pdf pval_visualiz/diff_react_expression_pval_corrected.gv \
      > pval_visualiz/diff_react_expression_pval_corrected.pdf
```

3.5 General Sample Uses

In this section we provide step by step instructions demonstrating general toolkit use cases.

3.5.1 Pipeline Execution Using Microarray Data

Below we show the complete sequence of steps involved in the execution of the pipeline depicted in Figure 3.1 when working with microarray data.

```
# Create and access working directory
```

```
mkdir working_dir
cd working_dir

# Download and decompress metabolic data
mkdir recon
wget -O recon/recon_2.03.xml \
     http://www.ebi.ac.uk/biomodels-main/download?mid=MODEL1504130000

# Download auxiliary metabolic network data
wget https://daormar.github.io/flux-capacitor/sample_data/aux_net_data.tar.gz
tar -zxvf aux_net_data.tar.gz

# Download microarray sample data
wget https://daormar.github.io/flux-capacitor/sample_data/marray_data.tar.gz
tar -zxvf marray_data.tar.gz

# Define CPLEX_BINARY_DIR variable
export CPLEX_BINARY_DIR=<directory_where_cplex_binary_is_stored>

# Predict human metabolic fluxes
auto_fba -m recon/recon_2.03.xml -c 0 -o standard_fba
auto_fba -m recon/recon_2.03.xml -c 1 -o shlomi_fba -d marray_data \
         -p marray_data/gse40595_subset_pdata.txt

# Study network robustness for an individual sample (it takes a while)
auto_fva -l shlomi_fba/lp/GSM997591_NS5.CEL.gz \
         -o fva_for_sample -g 0.8

# Visualize fluxes of a particular sample for the TCA cycle
mkdir flux_visualiz

## Generate csv file required by plot_metab_network
get_cplex_fluxes -f shlomi_fba/sol/GSM997591_NS5.CEL.gz.sol \
                 -m shlomi_fba/minfo/model -of 0 \
                 > flux_visualiz/GSM997591_NS5_fluxes.csv

plot_metab_network -s shlomi_fba/minfo/model \
                   -d flux_visualiz/GSM997591_NS5_fluxes.csv \
                   -f aux_net_data/tca_reactid.txt \
                   -e aux_net_data/tca_extern_metid.txt -t 2 \
                   > flux_visualiz/GSM997591_NS5_fluxes_tca.gv

neato -T pdf flux_visualiz/GSM997591_NS5_fluxes_tca.gv \
      > flux_visualiz/GSM997591_NS5_fluxes_tca.pdf

# Compare fluxes for multiple samples
mkdir hyp_testing
gather_sample_fluxes -d shlomi_fba > hyp_testing/shlomi_fba_sample_fluxes.csv
test_samples -p rnaseq_data/gse40595_subset_pdata.txt \
             -s hyp_testing/shlomi_fba_sample_fluxes.csv -u \
             > hyp_testing/shlomi_fba_sample_fluxes_pval.csv
correct_pvalues -p hyp_testing/shlomi_fba_sample_fluxes_pval.csv \
                > hyp_testing/shlomi_fba_sample_fluxes_pval_corrected.csv

# Network reduction
```

```
network_reducer -a standard_fba -lpm aux_net_data/empty_protected_metabid.txt \
                -lpr aux_net_data/retinol_reactid.txt \
                -md 1000 -mr 1000 -o reduced_network -li 10 -g 0.9

# Visualize p-values
mkdir pval_visualiz
plot_metab_network -s reduced_network/model \
                   -d hyp_testing/shlomi_fba_sample_fluxes_pval_corrected.csv \
                   -f aux_net_data/retinol_reactid.txt \
                   -e aux_net_data/retinol_extern_metid -t 5 \
                   > pval_visualiz/diff_react_expression_pval_corrected.gv
neato -T pdf pval_visualiz/diff_react_expression_pval_corrected.gv \
      > pval_visualiz/diff_react_expression_pval_corrected.pdf
```

3.5.2 Pipeline Execution Using RNA-Seq Data

When RNA-Seq expression data is provided, the commands required for the execution of the pipeline depicted in Figure 3.1 are the following:

```
# Create and access working directory
mkdir working_dir
cd working_dir

# Download and decompress metabolic data
mkdir recon
wget -O recon/recon_2.03.xml \
     http://www.ebi.ac.uk/biomodels-main/download?mid=MODEL1504130000

# Download auxiliary metabolic network data
wget https://daormar.github.io/flux-capacitor/sample_data/aux_net_data.tar.gz
tar -zxvf aux_net_data.tar.gz

# Download RNA-Seq sample data
wget https://daormar.github.io/flux-capacitor/sample_data/rnaseq_data.tar.gz
tar -zxvf rnaseq_data.tar.gz

# Define CPLEX_BINARY_DIR variable
export CPLEX_BINARY_DIR=<directory_where_cplex_binary_is_stored>

# Predict human metabolic fluxes
auto_fba -m recon/recon_2.03.xml -c 0 -o standard_fba
auto_fba -m recon/recon_2.03.xml -c 1 -o shlomi_fba \
         -r rnaseq_data/KIRC_120_sample_RPKM.txt \
         -p rnaseq_data/KIRC_120_sample_pdata_simplified.txt

# Study network robustness for an individual sample (it takes a while)
auto_fva -l shlomi_fba/lp/TCGA.B0.4833.01A.01R.1305.07 \
         -o fva_for_sample -g 0.8
```

```
# Visualize fluxes of a particular sample for the TCA cycle
mkdir flux_visualiz

## Generate csv file required by plot_metab_network
get_cplex_fluxes -f shlomi_fba/sol/TCGA.B0.4833.01A.01R.1305.07.sol \
                 -m shlomi_fba/minfo/model -of 0 \
                 > flux_visualiz/TCGA.B0.4833.01A.01R.1305.07_fluxes.csv

plot_metab_network -s shlomi_fba/minfo/model \
                   -d flux_visualiz/TCGA.B0.4833.01A.01R.1305.07_fluxes.csv \
                   -f aux_net_data/tca_reactid.txt \
                   -e aux_net_data/tca_extern_metid.txt -t 2 \
                   > flux_visualiz/TCGA.B0.4833.01A.01R.1305.07_fluxes_tca.gv

neato -T pdf flux_visualiz/TCGA.B0.4833.01A.01R.1305.07_fluxes_tca.gv \
      > flux_visualiz/TCGA.B0.4833.01A.01R.1305.07_fluxes_tca.pdf

# Compare fluxes for multiple samples
mkdir hyp_testing
gather_sample_fluxes -d shlomi_fba > hyp_testing/shlomi_fba_sample_fluxes.csv
test_samples -p rnaseq_data/KIRC_120_sample_pdata_simplified.txt \
             -s hyp_testing/shlomi_fba_sample_fluxes.csv -u \
             > hyp_testing/shlomi_fba_sample_fluxes_pval.csv
correct_pvalues -p hyp_testing/shlomi_fba_sample_fluxes_pval.csv \
                > hyp_testing/shlomi_fba_sample_fluxes_pval_corrected.csv

# Network reduction
network_reducer -a standard_fba -lpm aux_net_data/empty_protected_metabid.txt \
                -lpr aux_net_data/retinol_reactid.txt \
                -md 1000 -mr 1000 -o reduced_network -li 10 -g 0.9

# Visualize p-values
mkdir pval_visualiz
plot_metab_network -s reduced_network/model \
                   -d hyp_testing/shlomi_fba_sample_fluxes_pval_corrected.csv \
                   -f aux_net_data/retinol_reactid.txt \
                   -e aux_net_data/retinol_extern_metid -t 5 \
                   > pval_visualiz/diff_react_expression_pval_corrected.gv
neato -T pdf pval_visualiz/diff_react_expression_pval_corrected.gv \
      > pval_visualiz/diff_react_expression_pval_corrected.pdf
```

BIBLIOGRAPHY

Erdrich, P., R. Steuer, and S. Klamt (2015). "An algorithm for the reduction of genome-scale metabolic network models to meaningful core models". In: *BMC Syst Biol* 9, p. 48.

Fell, D. A. and J. R. Small (1986). "Fat synthesis in adipose tissue. An examination of stoichiometric constraints". In: *Biochem. J.* 238.3, pp. 781–786.

Gansner, E. R. and S. C. North (2000). "An open graph visualization system and its applications to software engineering". In: *SOFTWARE - PRACTICE AND EXPERIENCE* 30.11, pp. 1203–1233.

Gudmundsson, S. and I. Thiele (2010). "Computationally efficient flux variability analysis". In: *BMC Bioinformatics* 11, p. 489.

Mahadevan, R. and C. H. Schilling (2003). "The effects of alternate optimal solutions in constraint-based genome-scale metabolic models". In: *Metab. Eng.* 5.4, pp. 264–276.

Ortiz-Martínez, Daniel (2016). "Systems Biology Strategies to Study Cancer Metabolism". MA thesis. Universitat de València. URL: https://daormar.github.io/flux-capacitor/docsupport/dortiz_bio_msc_thesis.pdf.

Shlomi, T., M. N. Cabili, M. J. Herrgård, B. Ø. Palsson, and Eytan Ruppin (2008). "Network-based prediction of human tissue-specific metabolism". In: *Nat. Biotechnol.* 26(9), pp. 1003–10.

Thiele, I. et al. (2013). "A community-driven global reconstruction of human metabolism". In: *Nat. Biotechnol.* 31.5, pp. 419–425.