sysbiotech

Gi Bae Kim

Jul 19, 2021

CONTENTS:

CHAPTER

ONE

GETPRA

The GeTPRA framework

This project is to develop a framework that systematically predicts Gene-Transcript-Protein-Reaction Associations (GeTPRA) in human metabolims and updates a human genome-scale metabolic model (GEM) accordingly. This source code implements the GeTPRA framework.

Features This source code executes following steps in the GeTPRA framework in order:

- Get reactions from biochemical database using EC number
- Standardize metabolite information
- Compartmentalize metabolic reactions
- Generate GeTPRA
- Check 'Exist in Recon 2M.1'
- Check 'Blocked reaction'
- Check 'Experimental evidence available'

See below for the implementation of EFICAz and Wolf PSort (optional features)

Installation Procedure Note: This source code was developed in Linux, and has been tested in Ubuntu 14.04.5 LTS (i7-4770 CPU @ 3.40GHz)

1. Clone the repository
2. Create and activate virtual environment

```
$ virtualenv venv
$ source venv/bin/activate
```

3. Install packages at the root of the repository

```
$ pip install pip --upgrade
$ pip install -r requirements.txt
```

Input files for the GeTPRA framework Following working input files can be found in: *getpra/input_data/getpra_inputs/*. These files were used for the data presented in the manuscript.

Gene-transcript ID annotation file format - Download gene-transcript ID annotation file from [Ensembl BioMarts](http://www.ensembl.org/biomart/martview)

```
NCBI gene ID        Gene stable ID  Transcript stable ID    RefSeq mRNA ID  UCSC Stable␣
↪ID
2733        ENSG00000119392 ENST00000309971 NM_001003722    uc004bvj.4
2733        ENSG00000119392 ENST00000372770 NM_001499       uc004bvi.4
5690        ENSG00000126067 ENST00000373237 NM_002794       uc001bzf.4
5690        ENSG00000126067 ENST00000373237 NM_001199779    uc001bzf.4
5690        ENSG00000126067 ENST00000621781 NM_001199780    uc021olh.3
```

Download procedure

1. Go to [Ensembl BioMarts](http://www.ensembl.org/biomart/martview)
2. Click *Dataset* on the left menu

   Select *Ensembl Genes 89* in the drop-down menu *CHOOSE DATABASE*

   Select *Human genes (GRCh38.p10)* in the drop-down menu *CHOOSE DATASET*
3. Click *Filters* on the left menu

   Click *GENE:* in the main menu (center)

   Check *Gene type* and select *protein_coding*
4. Click *Attributes* on the left menu

   Check *Features* in the center
5. Click both *GENE:* and *EXTERNAL:* in the main menu (center)

   GENE: -> Ensembl -> Uncheck *Gene stable ID* and *Transcript stable ID*

   Check following items in order:

   - *EXTERNAL: -> External References (max 3) -> NCBI gene ID*
   - *GENE: -> Ensembl -> Gene stable ID*
   - *GENE: -> Ensembl -> Transcript stable ID*
   - *EXTERNAL: -> External References (max 3) -> RefSeq mRNA ID*
   - *EXTERNAL: -> External References (max 3) -> UCSC Stable ID*
6. Click the button *Results* on the top left
7. Click the button *Go* in the top center

- **File names in the source:**
  - *Ensembl_GRCh38_EnsemblDB_v84.txt*
  - *Ensembl_GRCh38_EnsemblDB_v85.txt*
  - *Ensembl_GRCh38_EnsemblDB_v86.txt*
  - *Ensembl_GRCh38_EnsemblDB_v87.txt*
  - *Ensembl_GRCh38_EnsemblDB_v88.txt*

`chem_xref.tsv` from `MetaNetX` - Download [chem_xref.tsv](http://www.metanetx.org/cgi-bin/mnxget/mnxref/chem_xref.tsv) from [MetaNetX](http://www.metanetx.org/) - File name in the source: *chem_xref.tsv*

`gene2ensembl.gz` from `NCBI FTP` - Download [gene2ensembl.gz](ftp://ftp.ncbi.nlm.nih.gov/gene/DATA/gene2ensembl.gz) from [NCBI FTP](ftp://ftp.ncbi.nlm.nih.gov/) - File name in the source: *gene2ensembl.gz*

`appris_data.principal.txt` from `APPRIS` - Download [appris_data.principal.txt](http://apprisws.bioinfo.cnio.es/pub/current_release/datafiles/homo_sapiens/GRCh38/appris_data.principal.txt) for annotation of principal isoform from [APPRIS](http://appris.bioinfo.cnio.es/) - File name in the source: *appris_data.principal.txt*

`subcellular_location.csv` from `The Human Protein Atlas` - Download [subcellular_location.csv](http://www.proteinatlas.org/download/subcellular_location.csv.zip) with subcellular localization information from [The Human Protein Atlas](http://www.proteinatlas.org/) - File name in the source: *subcellular_location.csv*

Recon model - Prepare a human genome-scale metabolic model with consistent TPR associations and that shows biologically reasonable simulation performance - File name in the source: *Recon2M.1_Entrez_Gene.xml*

EFICAz output file as input file - File name in the source: *20170110_EFICAz_result.txt*. - EFICAz can be run with a different set of peptide sequences (see below).

WoLF PSort output file as input file - File name in the source: *20170110_WoLFPSort_result.txt* - WoLF PSort can be run with a different set of peptide sequences (see below).

BRENDA data - Set user email address and password before implementing the GeTPRA framework. The framework programmatically fetches BRENDA data from BRENDA through its API.

##Implementation **Note**: All the arguments shown below should be provided when implementing the framework

Note: Make sure to provide own information for *-brenda_email* and *-brenda_pw*

Note: Implementation of this source code takes long (~ 8 h)

```
$ python run_GeTPRA_framework.py \
    -output_dir ./getpra_results/ \
    -ec ./input_data/getpra_inputs/20170110_EFICAz_result.txt \
    -sl ./input_data/getpra_inputs/20170110_WoLFPSort_result.txt \
    -brenda_email user_email_address \
    -brenda_pw user_password \
    -mnx_xref ./input_data/getpra_inputs/chem_xref.tsv \
    -ensembl ./input_data/getpra_inputs/Ensembl_GRCh38_EnsemblDB_v88.txt \
    -appris ./input_data/getpra_inputs/appris_data.principal.txt \
    -model ./input_data/getpra_inputs/Recon2M.1_Entrez_Gene.xml \
    -hpa ./input_data/getpra_inputs/subcellular_location.csv \
    -ncbi_id_information ./input_data/getpra_inputs/gene2ensembl.gz
```

##Output files from the GeTPRA framework - Raw output files from the GeTPRA framework, which were used for the publication, are available in: *getpra/getpra_results_publication_version/* - New output files upon implementation of the framework are generated in: *getpra/getpra_results/*. This folder is automatically created.

Implementation of EFICAz and Wolf PSort (optional) Output files of EFICAz and Wolf PSort serve as input files for the GeTPRA framework.

Peptide sequences of metabolic genes as inputs for EFICAz and Wolf PSort - Get peptide sequences of metabolic genes by implementing *./input_data/get_peptide_sequences.py*

```
$ python ./input_data/get_peptide_sequences.py \
-output_dir ./input_data/getpra_inputs/ \
-model ./input_data/getpra_inputs/Recon2M.1_Entrez_Gene.xml \
-ensembl ./input_data/getpra_inputs/Ensembl_GRCh38_EnsemblDB_v88.txt
```

- File name in the source: *Ensembl_peptide_seq_metabolic_genes.fa*

  >ENSP00000452494|ENST00000448914 TGGY >ENSP00000488240|ENST00000631435 GTGG >ENSP00000487941|ENST00000632684 GTGG >ENSP00000451515|ENST00000434970 PSY >ENSP00000451042|ENST00000415118 EI

Installation 1. Download [EFICAz2.5](http://cssb.biology.gatech.edu/skolnick/webservice/EFICAz2/index.html)

2. Set environment variable for EFICAz2.5

```
$ export EFICAz25_PATH="[insert-destination-directory]/EFICAz2.5.1/"
$ export PATH="${PATH}:${EFICAz25_PATH}"
```

3. Download [WoLF PSort](https://github.com/fmaguire/WoLFPSort)
4. Set environment variable for WoLF PSort

```
$ export WoLFPSort_PATH="[insert-destination-directory]/WoLFPSort/"
$ export PATH="${PATH}:${WoLFPSort_PATH}"
```

Implementation of EFICAz and Wolf PSort - Predict EC numbers using [EFICAz2.5](http://cssb.biology.gatech.edu/skolnick/webservice/EFICAz2/index.html)

```
$ python $EFICAz25_PATH/eficaz2.5 Ensembl_peptide_seq_metabolic_genes.fa
```

- Predict subcellular localization using [WoLF PSort](https://github.com/fmaguire/WoLFPSort)

```
$ WoLFPSort_PATH/bin/runWolfPsortSummary animal < Ensembl_peptide_seq_metabolic_
↪genes.fa
```

- **Output files from the above two implementations using peptide sequences of entire human genes are available in:**
  - EFICAz: *getpra/input_data/20170110_EFICAz_result_using_all_human_genes.txt*
  - Wolf PSort: *getpra/input_data/20170110_WoLFPSort_result_using_all_human_genes.txt*

Extract metabolic genes from EFICAz and Wolf PSort output data obtained with entire human genes - Extract metabolic genes from the EFICAz and Wolf PSort output data

```
$ python ./input_data/get_EFICAz_WolfPSort_results.py \
-output_dir ./input_data/getpra_inputs/ \
-model ./input_data/getpra_inputs/Recon2M.1_Entrez_Gene.xml \
-ec ./input_data/getpra_inputs/20170110_EFICAz_result_using_all_human_genes.
↪txt \
-sl ./input_data/getpra_inputs/20170110_WoLFPSort_result_using_all_human_genes.
↪txt \
-ensembl ./input_data/getpra_inputs/Ensembl_GRCh38_EnsemblDB_v88.txt
```

- **Resulting output files in the source:**
  - EFICAz: *getpra/input_data/Trimmed_EFICAz_result.txt*
  - Wolf PSort: *getpra/input_data/Trimmed_WoLFPSort_result.txt*

Publication Jae Yong Ryu 1, Hyun Uk Kim 1 & Sang Yup Lee. Framework and resource for more than 11,000 gene-transcript-protein-reaction associations in human metabolism., *Proc. Natl. Acad. Sci. U.S.A.*, 2017, http://www.pnas.org/content/early/2017/10/23/1713050114

CHAPTER

TWO

RECONMANAGER

Recon-manager procedure

This project is to develop a framework that systematically predicts Gene-Transcript-Protein-Reaction Associations (GeTPRA) in human metabolims and updates a human genome-scale metabolic model (GEM) accordingly. **Recon manager** is a part of the project, which is a collection of scripts to generate Recon 2M.1 and simulate Recon models.

Features Recon manager contains scripts that implement following tasks independently:

- Convert GPR to TPR associations
- Update metabolite information
- Calculate model statistics
- Evaluate functionality of metabolic model
- Reconstruct personal GEMs using tINIT

Installation *Major dependencies* - [gurobipy](http://www.gurobi.com/)

Procedure **Note**: This source code was developed in Linux, and has been tested in Ubuntu 14.04.5 LTS (i7-4770 CPU @ 3.40GHz)

1. Clone the repository
2. Create and activate virtual environment

```
$ virtualenv venv
$ source venv/bin/activate
```

3. Install packages at the root of the repository

```
$ pip install pip --upgrade
$ pip install -r requirements.txt
```

4. Install [gurobipy](http://www.gurobi.com/)

   In our case, we installed gurobipy in the root of a server, and created its symbolic link in *venv*:

```
$ ln -s /usr/local/lib/python2.7/dist-packages/gurobipy/ $HOME/recon-
↪manager/venv/lib/python2.7/site-packages/
```

Feature: Convert GPR to TPR associations Input arguments and corresponding files Following working input files can be found in: *./input_data/GPR_to_TPR_inputs*. These files were used for the data presented in the manuscript.

- *-o* : Output directory

- ***-model*** [COBRA-compliant SBML file (generic human GEM)]
  - File name in the source: *Recon2M.1_Entrez_Gene.xml*
- ***-gene_transcript_information*** [A list of gene IDs and their matching transcript IDs]
  - File name in the source: *Ensembl88_GRCh38_all_transcript_information.txt*

  File format

  NCBI gene ID Gene stable ID Transcript stable ID RefSeq mRNA ID UCSC Stable ID 2733 ENSG00000119392 ENST00000309971 NM_001003722 uc004bvj.4 2733 ENSG00000119392 ENST00000372770 NM_001499 uc004bvi.4 5690 ENSG00000126067 ENST00000373237 NM_002794 uc001bzf.4 5690 ENSG00000126067 ENST00000373237 NM_001199779 uc001bzf.4 5690 ENSG00000126067 ENST00000621781 NM_001199780 uc021olh.3

  Download procedure

  1. Go to [Ensembl BioMarts](http://www.ensembl.org/biomart/martview)
  2. Click *Dataset* on the left menu

     Select *Ensembl Genes 89* in the drop-down menu *CHOOSE DATABASE*

     Select *Human genes (GRCh38.p10)* in the drop-down menu *CHOOSE DATASET*
  3. Click *Filters* on the left menu

     Click *GENE:* in the main menu (center)

     Check *Gene type* and select *protein_coding*
  4. Click *Attributes* on the left menu

     Check *Features* in the center
  5. Click both *GENE:* and *EXTERNAL:* in the main menu (center)

     GENE: -> Ensembl -> Uncheck *Gene stable ID* and *Transcript stable ID*

     Check following items in order:

     - *EXTERNAL: -> External References (max 3) -> NCBI gene ID*
     - *GENE: -> Ensembl -> Gene stable ID*
     - *GENE: -> Ensembl -> Transcript stable ID*
     - *EXTERNAL: -> External References (max 3) -> RefSeq mRNA ID*
     - *EXTERNAL: -> External References (max 3) -> UCSC Stable ID*
  6. Click the button *Results* on the top left
  7. Click the button *Go* in the top center

Implementation **Note**: Running this script takes ~ 4 m

```
$ python model_GPR_to_TPR_converter.py \
-o ./results/GPR_to_TPR_results/ \
-gene_transcript_information ./input_data/GPR_to_TPR_inputs/Ensembl88_GRCh38_all_
↪transcript_information.txt \
-model ./input_data/GPR_to_TPR_inputs/Recon2M.1_Entrez_Gene.xml
```

Feature: Update metabolite information Input arguments and corresponding files Following working input files can be found in: *./input_data/metabolite_information_update_inputs*. These files were used for the data presented in the manuscript.

- *-o* : Output directory
- ***-model*** [COBRA-compliant SBML file (generic human GEM)]
  - File name in the source: *Recon2M.1_Entrez_Gene.xml*
- ***-mnx_xref*** [Info on chemical identifiers from [MetaNetX](http://www.metanetx.org/)]
  - File name in the source: *chem_xref.tsv*
  - Click [chem_xref.tsv](http://www.metanetx.org/cgi-bin/mnxget/mnxref/chem_xref.tsv) for downloading
- ***-mnx_prop*** [Info on chemical structures from [MetaNetX](http://www.metanetx.org/)]
  - File name in the source: *chem_prop.tsv*
  - Click [chem_prop.tsv](http://www.metanetx.org/cgi-bin/mnxget/mnxref/chem_prop.tsv) for downloading
- ***-bigg*** [Info on BiGG metabolites from [BiGG Models](http://bigg.ucsd.edu/)]
  - File name in the source: *bigg_models_metabolites.txt*
  - Click [bigg_models_metabolites.txt](http://bigg.ucsd.edu/static/namespace/bigg_models_metabolites.txt) for downloading
- ***-chebi*** [Info on ChEBI and InChI from [ChEBI](https://www.ebi.ac.uk/chebi/init.do)]
  - File name in the source: *chebiId_inchi.tsv*
  - Click [chebiId_inchi.tsv](ftp://ftp.ebi.ac.uk/pub/databases/chebi/Flat_file_tab_delimited/chebiId_inchi.tsv) for downloading

Implementation **Note**: Running this script takes ~ 5 s

```
$ python model_update_metabolite_information.py \
-o ./results/metabolite_information_update_results/ \
-model ./input_data/metabolite_information_update_inputs/Recon2M.1_Entrez_Gene.xml \
-mnx_xref ./input_data/metabolite_information_update_inputs/chem_xref.tsv \
-mnx_prop ./input_data/metabolite_information_update_inputs/chem_prop.tsv \
-bigg ./input_data/metabolite_information_update_inputs/bigg_models_metabolites.txt \
-chebi ./input_data/metabolite_information_update_inputs/chebiId_inchi.tsv
```

Feature: Calculate model statistics Input arguments and corresponding files Following working input files can be found in: *./input_data/model_function_inputs*. These files were used for the data presented in the manuscript.

- *-o* : Output directory
- ***-model*** [COBRA-compliant SBML file (generic human GEM)]
  - File name in the source: *Recon2M.1_Entrez_Gene.xml*
- ***-medium*** [A representative medium (RPMI-1640 medium)]
  - File name in the source: *RPMI1640_medium.txt*

File format

```
EX_gly_LPAREN_e_RPAREN_     -0.05   1000
EX_arg_L_LPAREN_e_RPAREN_   -0.05   1000
EX_asn_L_LPAREN_e_RPAREN_   -0.05   1000
EX_asp_L_LPAREN_e_RPAREN_   -0.05   1000
EX_cys_L_LPAREN_e_RPAREN_   -0.05   1000
EX_glu_L_LPAREN_e_RPAREN_   -0.05   1000
EX_his_L_LPAREN_e_RPAREN_   -0.05   1000
```

Implementation **Note**: Running this script takes ~ 47 m

```
$ python model_metabolic_model_statistics.py \
-o ./results/model_statistics_results/ \
-medium ./input_data/model_function_inputs/RPMI1640_medium.txt \
-model ./input_data/model_function_inputs/Recon2M.1_Entrez_Gene.xml
```

Feature: Evaluate functionality of metabolic model Input arguments and corresponding files Following working input files can be found in: *./input_data/model_function_inputs*. These files were used for the data presented in the manuscript.

- *-o* : Output directory
- ***-model*** [COBRA-compliant SBML file (generic human GEM)]
  - File name in the source: *Recon2M.1_Entrez_Gene.xml*
- ***-medium*** [A representative medium (RPMI-1640 medium)]
  - File name in the source: *RPMI1640_medium.txt*
- ***-defined_medium*** [A defined minimal medium]
  - File name in the source: *Defined_medium.txt*
- ***-es_genes*** [A list of essential genes]
  - File name in the source: *Essential_genes_from_wang_et_al.txt*
- ***-ne_genes*** [A list of non-essential genes]
  - File name in the source: *Non_essential_genes_from_wang_et_al.txt*
- ***-c_source*** [A list of carbon sources]
  - File name in the source: *atp_carbon_source.txt*
- *-biomass* : Reaction ID for biomass generation equation
- *-oxygen* : Reaction ID for oxygen uptake
- *-atp* : Reaction ID for ATP production

Implementation **Note**: Directly insert reaction IDs in terminal for *-biomass*, *-oxygen* and *-atp*

Note: Running this script takes ~ 7 m

```
$ python model_metabolic_function.py \
-o ./results/model_function_results/ \
-model ./input_data/model_function_inputs/Recon2M.1_Entrez_Gene.xml \
-medium ./input_data/model_function_inputs/RPMI1640_medium.txt \
-defined_medium ./input_data/model_function_inputs/Defined_medium.txt \
-es_genes ./input_data/model_function_inputs/Essential_genes_from_wang_et_al.txt \
-ne_genes ./input_data/model_function_inputs/Non_essential_genes_from_wang_et_al.txt \
```

(continues on next page)

(continued from previous page)

```
-c_source ./input_data/model_function_inputs/atp_carbon_source.txt \
-biomass biomass_reaction \
-oxygen EX_o2_LPAREN_eRPAREN \
-atp DM_atp_c_
```

Feature: Reconstruct personal GEMs using [tINIT](http://msb.embopress.org/content/10/3/721.long) Input arguments and corresponding files Following working input files can be found in: *./input_data/tINIT_inputs*. These files were used for the data presented in the manuscript.

- *-o* : Output directory
- ***-model*** [COBRA-compliant SBML file (generic human GEM)]
  - File name in the source: *Recon2M.1_Entrez_Gene.xml*
- ***-medium*** [A representative medium (RPMI-1640 medium)]
  - File name in the source: *RPMI1640_medium.txt*
- ***-task*** [A list of metabolic tasks]
  - File name in the source: *MetabolicTasks.csv*
- ***-present_reaction*** [A list of reactions that should be present in model]
  - File name in the source: *essential_reactions.txt*
- ***-present_metabolite*** [A list of metabolites that should be present in model]
  - File name in the source: *essential_metabolites.txt*
- ***-i*** [Omics data]
  - File name in the source: *BLCA_T_TTL.csv*
- *-biomass* : Reaction ID for biomass generation equation

Implementation **Note**: Directly insert a reaction ID in terminal for *-biomass*

Note: Running this script for a Recon model takes ~ 8 m

```
$ python personal_GEM_tINIT.py \
-o ./results/tINIT_results/ \
-medium ./input_data/tINIT_inputs/RPMI1640_medium.txt \
-model ./input_data/tINIT_inputs/Recon2M.1_Entrez_Gene.xml \
-biomass biomass_reaction \
-task ./input_data/tINIT_inputs/MetabolicTasks.csv \
-present_reaction ./input_data/tINIT_inputs/essential_reactions.txt \
-present_metabolite ./input_data/tINIT_inputs/essential_metabolites.txt \
-i ./input_data/tINIT_inputs/BLCA_N_TTL.csv
```

Feature: Predict flux using transcript-level RNA-Seq data and GeTPRA Input arguments and corresponding files Following working input files can be found in: *./input_data/Flux_prediction*. These files were used for the data presented in the manuscript.

- *-o* : Output directory
- ***-g_model*** [COBRA-compliant SBML file (generic human GEM)]
  - File name in the source: *Recon2M.2_BiGG_UCSC_Transcript.xml*
- ***-c_model*** [COBRA-compliant SBML file (context-specific human GEM)]

  - File name in the source: *LIHC_TCGA-BC-A10Q.xml*

- ***-getpra*** [GeTPRA file]
  - File name in the source: *GeTPRA.txt*

- ***-use_getpra*** [Option for flux prediction using transcript-level data]
  - Insert`yes` for flux prediction using transcript-level data
  - Insert`no` for flux prediction using gene-level data

Implementation **Note**: Running this script takes ~ 7 m

```
$ python flux_prediction.py \
-o ./results/Flux_prediction/ \
-i ./input_data/Flux_prediction/LIHC_TCGA-BC-A10Q.csv \
-getpra ./input_data/Flux_prediction/GeTPRA.txt \
-g_model ./input_data/Flux_prediction/Recon2M.2_BiGG_UCSC_Transcript.xml \
-c_model ./input_data/Flux_prediction/LIHC_TCGA-BC-A10Q.xml \
-use_getpra yes
```

Publication Jae Yong Ryu 1, Hyun Uk Kim 1 & Sang Yup Lee. Framework and resource for more than 11,000 gene-transcript-protein-reaction associations in human metabolism., Proc. Natl. Acad. Sci. U.S.A., 2017, http://www.pnas.org/content/early/2017/10/23/1713050114

CHAPTER

THREE

DEEPEC

DeepEC running procedure

Note: Size of the protein sequence input file should be adjusted according to the memory size of your computer. This source code was developed in Linux, and has been tested in Ubuntu 14.04 with Python 2.7, Python 3.4, Python 3.5 or Python 3.6. It should be noted that Python 3.7 is currently not supported.

1. Clone the repository

```
$ git clone https://bitbucket.org/kaistsystemsbiology/deepec.git
```

2. Create and activate a conda environment

```
$ conda env create -f environment.yml
$ conda activate deepec
```

3. Example

- Run DeepEC

```
$ python deepec.py -i ./example/test.fa -o ./output
```

CHAPTER

FOUR

DEEPDDI

DeepDDI running procedure This project is to develop a framework that systematically predicts drug-drug interactions (DDIs) and drug-food interactions (DFIs).

Features - DeepDDI predicts DDIs and DFIs using names of drug-drug or drug-food constituent pairs and their structural information as inputs - DeepDDI predicts 86 DDI types as outputs of human-readable sentences

Note: This source code was developed in Linux, and has been tested in Ubuntu 14.04

1. Create and activate virtual environment

```
$ virtualenv venv
$ source venv/bin/activate
```

2. Install packages at the root of the repository

```
$ pip install pip --upgrade
$ pip install -r requirements.txt
```

3. Install [rdkit](http://www.rdkit.org/)

   In our case, we installed rdkit in the root of a server using apt-get (i.e., *apt-get install rdkit*), and created its symbolic link in *venv*:

```
$ ln -s /usr/lib/python2.7/dist-packages/rdkit/ ./venv/lib/python2.7/site-packages/
```

Input files for the DeepDDI

Following working input files can be found in: *./data/*. These files were used for the data presented in the manuscript.

Example

```
$ python run_DeepDDI.py -i ./examples/input_structures.txt -o ./output_dir/
```

CHAPTER

FIVE

IBRIDGE

iBrdige running procedure

CHAPTER

SIX

MELI-3D

MELI-3D running procedure

CHAPTER

SEVEN

13C-MFA

13C-MFA running procedure

Note:

- This is a Matlab based simulator that helps to analyze fluxes in isotopically steady state using a 13C-labeled carbon source*
- It was developed and tested in Windows 10 (64 bit), Matlab R2019b*
- By running the provided p-files using Matlab, flux analysis using 13C can be performed*
- For more detailed instructions, please refer to the provided ppt-file*

CHAPTER

EIGHT

DEEPTFACTOR

DeepTFactor running procedure

Note: This source code was developed in Linux, and has been tested in Ubuntu 16.04 with Python 3.6.

1. Clone the repository

```
$ git clone https://bitbucket.org/kaistsystemsbiology/deeptfactor.git
```

2. Create and activate virtual environment

```
$ conda env create -f environment.yml
$ conda activate deeptfactor
```

3. **To use GPU for the computation, install an appropriate version of pytorch (and cuda).**

    - Please refer to <https://pytorch.org>

Example

- Run DeepTFactor

```
$ python tf_running.py -i ./Dataset/example_tf.fasta -o ./result -g cpu
$ python tf_running.py -i ./Dataset/example_tf.fasta -o ./result -g cuda:1
```

CHAPTER

NINE

RETROPRECURSORSELECTION

Retro-precursor-selection running procedure

Note This source code works on Linux and has been tested on Ubuntu 16.04 with Python 3.6.

1. Clone the repository:

```
$ git clone https://wdjang@bitbucket.org/kaistsystemsbiology/retro-precursor-selection.
↪git
```

2. Create and activate a conda environment (It takes less than 30 seconds):

```
$ conda env create -f environment.yml
$ conda activate SSA
```

Example

```
$ python run_ssa.py -i 'CCCCN'[SMILES of target product] -o output[Name of output␣
↪directory]
```

Three output files are generated in a folder that is newly created after computation: molecule_type.txt, predicted_precursors.txt, reaction_center.txt

Run time of this source code is usually ~60 seconds.

CHAPTER

TEN

DEEPRFC

DeepRFC running procedure

Note: This source code was developed in Linux, and has been tested in Ubuntu 16.04 with Python 3.6. Anaconda was used as a package manager

1. Clone the repository

```
$ git clone https://bitbucket.org/kaistsystemsbiology/deeprfc.git
```

2. Create and activate a conda environment

```
$ conda env create -f environment.yml
$ conda activate deeprfc
```

3. If pip error occurs due to Pytorch version,

   Remove followings from *environment.yml*

   ```
   Torch==1.2.0+cu92
   Torchvision==0.4.0+cu02
   ```

   -Install an appropriate version of Pytorch, according to your computing envrionment. Please refer to https://pytorch.org/get-started/locally/

Example

- Run DeepRFC

```
$ python deeprfc.py -i ./example/test.txt -o ./output
```

CHAPTER

ELEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX