A python package to model Circular Economy policy and technological interventions in Environmentally Extended Input-Output Analysis starting from SUTs (EXIOBASE V3.3)

Documentation: https://cmlplatform.github.io/pycirk/

10.5281/zenodo.1492957

1.1 1. Installation

1.1.1 1.1. Stable release

Run in your terminal:

$ pip install pycirk

1.1.2 1.2. From source

Clone repository:

$ git clone https://fdonati@bitbucket.org/CML-IE/pycirk.git

Once you have a copy of the source, you can install it with:

$ python setup.py install
1.2 2. Usage

1.2.1 2.1. Import package

    import pycirk

1.2.2 2.2. Initialize

    s = pycirk.Launch(method, directory, aggregation)

1.2.3 2.3. set your scenarios and analysis

    1. Open scenarios.xls in the directory that was specified
    2. From there you can specify interventions and parameters for the analysis
    3. save and continue to the following steps

1.2.4 2.4. Run scenarios

Run one specific scenario

    s.scenario_results(scen_no, output_dataset) (0 = baseline)

Run all scenarios

    s.all_results()

1.2.5 2.5. save scenarios

Save your results

    s.save_results()

2.6. Use from command line

2.6.1. pycirk --help

Usage: pycirk [OPTIONS]


Options:

<table>
<thead>
<tr>
<th>Command</th>
<th>Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>-tm, --transf_method TEXT</td>
<td>0 = PXP ITA_TC; 1 = PXP ITA_MSC</td>
</tr>
<tr>
<td>-dr, --directory TEXT</td>
<td>if left black it will be default</td>
</tr>
<tr>
<td>-ag, --aggregation</td>
<td>1 = bi-regional (EU-ROW) 0 = None (49 regions)</td>
</tr>
<tr>
<td>-sc, --scenario TEXT</td>
<td>all, 1, 2,... accepted - 0=baseline</td>
</tr>
<tr>
<td>-s, --save TEXT</td>
<td>False=no, True=yes</td>
</tr>
<tr>
<td>-od, --output_dataset</td>
<td>False=no, True=yes</td>
</tr>
<tr>
<td>--help</td>
<td>Show this message and exit.</td>
</tr>
</tbody>
</table>

2.6.2. Command example
1.3 3. Features

Examples of policies that can be modelled through the software:

- sharing
- recycling
- life extension
- rebound effects
- substitution
- market and value added changes
- efficiency

The tables in which it is possible to apply changes:

- total requirement matrix (A)
- intermediate transactions (Z)
- final demand (Y)
- primary inputs (W)
- emission intermediate extensions (E)
- material intermediate extensions (M)
- resource intermediate extensions (R)
- emission final demand extension (EY)
- material final demand extension (MY)
- resource final demand extensions (RY)
- primary inputs coefficients (w)
- emission intermediate extensions coefficients (e)
- material intermediate extensions coefficients (m)
- resource intermediate extensions coefficients (r)
- emission final demand extension coefficients (eY)
- material final demand extension coefficients (mY)
- resource final demand extensions coefficients (rY)

It is possible to specify:

- region of the intervention
- whether the intervention affects domestic, import transactions or both
1.4 4. Important modules

1.4.1 4.1. scenarios.xls

From this .xls file it is possible to set different types of interventions and the analysis to perform:

- matrix = specifies in which matrix of IOT the changes are applied
- change_type = Primary and ancillary are only used to specify the type of intervention from a conceptual level
- reg_o or reg_d = Regional coordinates (o=origin or row, d=destination or column)
- cat_o or cat_d = category (e.g. products or extensions ) coordinates (o=origin or row, d=destination or column)
- kt = technical coefficient (max achievable technically); a negative value means reduction; unit = %
- ka = absolute values for addition
- kp = penetration coefficient (level of market penetration of the policy); unit = %
- copy = allows you to copy a specific transaction to a different point in the matrices (useful for proxy creation)
- substitution = tells the software whether it needs to substitute values among specified categories
- sk = which intervention should be substituted
- swk = Substitution weighing factor (how much of the original transaction should be substituted); allows to simulate difference in prices and physical properties between categories; unit = %

These can be set for:
- product category e.g. C_STEL (basic iron), C_PULP (pulp), etc.
- final demand category e.g. F_HOUS (households), F_GOVE (government), etc.
- primary input category e.g. E_HRHS (employment highly skilled), T_TLSA (taxes less subsidies), etc.
- emissions extensions e.g. E_CO2_c (CO2 - combustion)
- material extensions e.g. NI.02 (Nature Inputs: Coking Coal)
- resource extension e.g. L_1.1 (Land use - Arable Land - Rice)

Furthermore, from the analysis sheet you can set the following variables to be compared in the analysis:

- product categories
- primary input categories
- emissions extensions
- material extensions
- resource extensions
- region of interest
- impact categories # Please see the data_validation_list sheet in the scenarios.xls file for the comprehensive list

1.5 6. Credits

Thanks to dr. Arnold Tukker, dr. Joao Dias Rodriguez for the supervision dr. Arjan de Koning for knowledge support in exiobase MSc. Glenn Auguilar Hernandez for testing

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.
2.1 Stable release

To install pycirk, run this command in your terminal:

```
$ pip install pycirk
```

This is the preferred method to install pycirk, as it will always install the most recent stable release. If you don’t have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for pycirk can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/FDonati/pycirk
```

Or download the tarball:

```
$ curl -OL https://github.com/FDonati/pycirk/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```
Usage

To use pycirk in a project:

```python
import pycirk
```
4.1 pycirk package

4.1.1 Submodules

4.1.2 pycirk.cli module

4.1.3 pycirk.fundamental_operations module

Created on Wed Nov 2 12:05:08 2016

Description: Class to perform SUT and IOT transformations and balancing

Scope: Modelling circular economy policies in EEIOA

@author: Franco Donati @institution: Leiden University CML

class pycirk.fundamental_operations.Operations
    Bases: object

    Contains all basic operations to transform SUTs into IOTs and verify them. It contains two subclasses defining two different transformation methods: PxP ITA Market Share and Technical Coefficient methods. Note: It should be expanded to other transformation methods in the future.

class IOT
    Bases: object

    General IOT operations subclass. Some methods repeat from other subclasses, but it's good to have them divided for clarity.

    A (inv_diag_x)
    Technical coefficient matrix $A = Z \times \text{inv}(	ext{diag}(x))$

    B (inv_diag_x)
    Calculates extensions intensities
**IOT** \((Y, W, E, R, M)\)

- \(L()\)
  - Leontief inverse \(L = (I-A)^{-1}\)

- \(R(diag_x)\)
  - Calculates absolute extensions

- \(RY(diag_yj)\)
  - Calculates absolute extensions in final demand

- \(Z(diag_x)\)
  - Total product output \(Z = A \times diag_x\)

- \(bY(inv_diag_yj)\)
  - Calculates intensities of extensions in final demand
    Method for transformation matrix of \(bY\) (e.g. final demand emissions)

- \(x(Y)\)
  - Total product output is the sum of \(S_i\) and \(y\)

- \(x_IAy(y)\)
  - Total product output \(x = \text{inv}(I - A) \times yi\)

**class PxP_ITA_MSC**

*Model with Market Share Coef. Prod x Prod Industry Technology assumption*

- \(A(D)\)
  - Total requirement multipliers \(A = Z \times D\)

- \(B(D, inv_diag_g)\)
  - Calculates extensions intensities

- \(D(inv_diag_q)\)
  - Market share coefficients \(D = V \times \text{inv}(\text{diag}(q))\)

- \(L()\)
  - Leontief inverse \(L = (I-A)^{-1}\)

- \(R(diag_q)\)
  - Calculates absolute extensions

- \(S(inv_diag_g)\)
  - Intermediate coefficients
    Input requirements \(Z = U \times \text{inv}(\text{diag}(g))\)

- \(Z(D, diag_q)\)
  - Intermediates \(Z = \text{inter}_\text{coef} \times D \times \text{diag}(q)\)

**class PxP_ITA_TC**

*Model with Transformation Coefficients ProdxProd Industry Technology assumption*

- \(B(T, inv_diag_q)\)
  - Calculates extension intensity

- \(L(T, inv_diag_q)\)
  - Input coefficients intermediates \(A = U \times T \times \text{inv}([\text{diag}(q)])\)

  - Multiplier matrix \(L = (I-A)^{-1}\)

- \(R(diag_q)\)
  - Calculates absolute extensions
\[ T = (\text{inv\_diag\_g}) \]
Transformation matrix \( T = \text{inv(diag(g))} \times V \)

\[ Z = (U) \]
Intemediates \( Z = U \times T \)

calculate_characterized()

delta_Y (Yalt)
method to calculate difference in \( Y \) \( Y = \) final demand baseline \( Yalt = \) final demand scenario

delta_x (Lalt, y)
method to calculate difference in \( q \) \( L = \) Leontief of baseline \( Lalt = \) Leontief of scenario

inv()
Returns inverse by dividing by 1 and eliminating inf and nan values

verifyIOT (Y, W)

4.1.4 pycirk.labels module

Created on sat Jan 28 2017
Description: Labelling elements for SUTs and IOTs
Scope: Modelling the Circular Economy in EEIO
@author: Franco Donati @institution: Leiden University CML

class pycirk.labels.Labels
Bases: object

apply_labels (matrix, labels, axis=0)
Applies labels to a dataframe \( \text{axis} = 0 \Rightarrow \text{Index} \) \( \text{axis} = 1 \Rightarrow \text{columns} \)

calc_no_of_something (labels)

get_labels (matrix)
Collects labels from a dataframe

get_unique_labels (list_of_labels, for_units=True)

identify_labels (M_name)

list_of_something (labels)

load_labels (directory)

organize_unique_labels (directory)

relabel (M, column_labels, index_labels)
Processes apply_labels and apply _names together

relabel_to_save (data, trans_method, labels_directory)
This function makes sure that everything is labeled in IOT tables
trans_method = 0 is prod x prod, 1 is ind x ind

save_labels (data, directory)
4.1.5 pycirk.make_scenarios module


Description: Reading policy values and modifying matrices for scenarios

Scope: Modelling the Circular Economy in EEIO

@author: Franco Donati @institution: Leiden University CML

pycirk.make_scenarios.balancing_operation(V, U, Y, W)

Re-balancing of supply-use tables after data changes

V (supply) : numpy.array
U (use) : numpy.array
Y (final_demand) : numpy.array
W (primary_inputs) : numpy.array
output : dict

It outputs a dictionary containing a re-balanced supply-use tables system where:

V = supply table
U = use table
Y = final demand
W = primary inputs

pycirk.make_scenarios.basic_add(a, at)

Adds values together

a : numpy.array
at : numpy array

pycirk.make_scenarios.basic_mult(id, a, kt, kp)

Policy intervention

It may be a primary intervention or an acillary action.

a: numpy.array a supply chain or a point in it subject to a change
kt: float technical coefficient (max achievable technically)
kp: float penetration coefficient (level of market penet. of the policy)
id: int identification number of the intervention in case of missing information

A a numpy.array of the same order/shape of a

pycirk.make_scenarios.counterfactual(scen_file, scen_no, M, M_name, labels)

Separates changes by matrix subject to intervention and apply them on a specific matrix

scen_file: str directory of the file in which the scenarios are specified
scen_no [int] specific scenario e.g “1” or “scenario_1”
M [numpy.array] matrix affected by the policies
M_name [str] matrix name as diplayed under sheet_name[“matrix”]
labels [obj] matrix labels

A numpy array modified according to the specified changes in the scenario file
pycirk.make_scenarios.counterfactual_engine (M, inter_sets, subs=False, copy=False)
This function allows for the processing of the specified interventions onto a selected matrix. It calls various
functions to modify the values in a specified matrix.

M [numpy.array] matrix of reference

inter_sets: dict contains all specification concerning the changes to be applied (intervention sets)

subs [bool] If True it will call the substitution function according to specifications in scenarios.xlsx

copy [bool] if True it will copy value from one part of the matrix to another according to specifications in
scenarios.xlsx

i : index coordinate
g : column coordinate
ide : intervention identification number
kt : technical change coefficient
kp : market penetration coefficient
fx_kp : market penetration coeffient applicable to substitution
expan : expansion coef. (used only for simple transaction changes)

A numpy.array of the modified matrix

pycirk.make_scenarios.make_counterfactuals (data, scen_no, scen_file, labels)
Calculate all the counterfactual IO matrices

data [obj] An object containing all necessary matrices of the IO system

scen_no [int] the identification number of the scenario to reference in scen_file

scen_file [str] the directory where the scenarios.xlsx file is store

labels [obj] an object containing all labels for the IO matrices

An object containing a modified IO system

pycirk.make_scenarios.make_counterfactuals_SUT (data, scen_no, scen_file, labels)
Calculate all the counterfactual SUT matrices

data [obj] An object containing all necessary matrices of the SUT system

scen_no [int] the identification number of the scenario to reference in scen_file

scen_file [str] the directory where the scenarios.xlsx file is store

labels [obj] an object containing all labels for the SUT matrices

An object containing a modified SUT system

pycirk.make_scenarios.make_new (filtered_changes, M, M_name, labels)
Organizes the data concerning the changes and calls the functions to modified matrices based on specied sce-
narios

filtered_changes: pandas.DataFrame A table filtered by matrix name containing all changes to be applied

M [numpy.array] matrix on which to implement the changes

M_name [str] nomenclature referring to the matrix to be changed

labels: obj object containing all matrix labels
g is any column in the matrix
i is any row row in the matrix
A numpy.array of the processed matrix

pycirk.make_scenarios.substitution(d, s, fx_kp)
Moves the value from one or multiple cells (in the same row or column)
Substitution: Material substitution or certain types of rebound effects
If the size of the array of the original value is different from that of the destination (substituted), we obtain the total of the value to be substituted and the substitution is implemented by dividing the tot by the number of elements on the destination array and then added to the destination array (equally distributed)

d  [numpy.array] transaction with which we are substituting
s  [numpy.array] original transaction that was subject to changes (the transactions from which the value is coming from)
fx_kp  [float] relative size of c that is added on the transaction to expand d
A numpy.array of modified d

4.1.6 pycirk.make_secondary_flows module
Created on Mon Jul 16 15:10:19 2018 Description: Modifying SUT to ensure appearance of secondary material flows in IOT
Scope: Modelling the Circular Economy in EEIO
@author: Franco Donati @contributor: Arjan de Koning @institution: Leiden University CML

pycirk.make_secondary_flows.allocate_sec_mat(V, U, Y, prod_or, ind_or)
This function allows to move the primary material output from the secondary material industries to the secondary material output. This allows for the presence of secondary materials in the IOT once they are transformed from SUTS.

prod_or = row position of the primary supplied material ind_or = colum pos. of the primary industry supplying primary material

pycirk.make_secondary_flows.make_secondary(data)
This allows to allign secondary flow in such a way that they then appear in the IOT

Primary Products’ positions

Primary Sectors’ positions:

4.1.7 pycirk.organize_io module
Created on Tue Feb 7 16:29:23 2017
Description: Organize essential tables for saving
Scope: Modelling the Circular economy in EEIO
@author: Franco Donati @institution: Leiden University CML
pycirk.organize_io.organizer(data)

### 4.1.8 pycirk.positions module

Created on Wed Jan 23 10:58:46 2019

Description: Finding the position of labels

Scope: Modelling the Circular Economy in EEIO

@author: Franco Donati @institution: Leiden University CML

```python
class pycirk.positions.make_coord_array(cat_coord, reg_coord, no_countries, no_categories):
    It creates an array of coordinates based on the specification of the users.
    cat_coord [int, numpy.array, bool] the numerical coordinate of a specific category belonging to a matrix in the
    IO or SUT system. If None is passed then it will return an array of all coordinates in range no_categories.
    reg_coord [int, numpy.array, bool] the numerical coordinate of a specific region in the IO or SUT system. If
    None is passed then it will return an array of all coordinates in range no_countries.
    no_countries [int] the total number of countries or regions in the dataset
    no_categories [int] the total number of categories referring one axis in the chosen matrix
    A numpy.array referring to each coordinate point specified by the user
```

```python
class pycirk.positions.make_coord_array_for_make_sec(coordinates, no_countries, no_categories):
    It creates an array of coordinates based on the total location of secondary materials and processing categories
    coordinates [int, numpy.array] the numerical coordinate of secondary categories belonging to the SUT system
    no_countries [int] the total number of countries or regions in the dataset
    no_categories [int] the total number of categories referring one axis in the chosen matrix
    A numpy.array referring to each coordinate point
```

```python
class pycirk.positions.single_position(item, labels):
    Takes a dataframe of the multiindex and identifies the position of the specified values
    item [str] The label the user is looking for
    labels [obj] An object containing a set of labels (as specified in the labels.py module)
    An numpy.array containing the coordinate of a specific label or None in case of there is no specified label
```

### 4.1.9 pycirk.pycirk module

Created on Tue Nov 15 16:29:23 2016

Description: Outputting scenarios

Scope: Modelling the Circular Economy in EEIO

@author: Franco Donati @institution: Leiden University CML

```python
class pycirk.pycirk.Launch(method=0, make_secondary=False, save_directory=None, aggregation=1, file=None, test=False):
    Bases: object
    Pycirk’s main class and methods
```

4.1. pycirk package
Initialize the pycirk programme to make EEIO scenarios and analysis. From here, you can launch all the analysis specifications listed under scenarios.xlsx

**method**  [int]  SUTs to IO transformation methods

- 0 = Prod X Prod Ind-Tech Assumption Technical Coeff method
- 1 = Prod X Prod Ind-Tech Assumption Market Share Coeff method

**make_secondary**  [bool]  modifies SUT so that secondary technologies which process scrap materials into primary materials are also available in the IO tables

- False = Don’t modify
- True = Modify

**save_directory**  [str]  directory in which you want to work and save your results

**aggregation**  :  int, bool

- 0 = None (multi-regional 49 regions)
- 1 = bi-regional (EU- ROW)

**file**  [bool, str]  allows you to specify where the dataset is placed. None will use the default location within the installed package

**test**  [bool]  if set to true it will run the test settings under under pycirk/tests

**scenario_results**  [int]  Allows to calculate the results for a given specified scenario

- 0 = baseline data

**all_results**  :  Retrieves all results for all specified scenarios and baseline

**save_results**  [int and bool]  save all specified analytical results from all scenario and baseline

**analysis.xlsx**  [excel file]  to be found under the default folder on the specified directory it allows to specify the parameters for your scenario and analysis

**IO tables**  [pkl]  IO tables of the specified scenarios, these are located in the output folder in the save directory

**results**  [DataFrame]  results gathered from the processed scenarios and baseline

**all_results** ()  
Process all scenarios and collects their results together with Baseline analysis results

It outputs a pandas.DataFrame with all results

**save_results** (scen_no=None, output_dataset=False)  
Saves all results in excel format for info and results or in pickle format for the dataset

**scen_no** :  int

- 0 = baseline
- 1-n = specified scenarios

**output_dataset**  :  bool  If true it will output a dictionary containing all IOT tables in pd.DataFrames

Default values will save all results from the all_results method and they will output only scenario.xlsx and info_and_results.xlsx

Output_dataset is only possible when scen_no is specified in which case it would save also a data.pkl file

**scenarios.xlsx**  [excel file]  scenario settings excel file used for the analysis in the same output directory with the results
**info_and_results.xlsx**  [excel file] excel file containing general info about project plus the results from the analysis

**data.pkl**  [pickle file] new modified IO dataset in pickle format This is only possible if outputting single scenario (scen_no != None)

**scenario_results**(scen_no, output_dataset=False)
Run to output results of a specified scenario

**scen_no**: int  0 = baseline 1-n = specified scenarios

**output_dataset**: bool  If true it will output a dictionary containing all IOT tables in pd.DataFrames

specified results in DataFrame form or a dictionary containing results and a dictionary of dataframes containing IO matrices

### 4.1.10 pycirk.pycirk_settings module

Created on Wed Feb 20 16:29:23 2019

Description: Setting parameters for pycirk

Scope: Modelling the Circular Economy in EEIO

@author: Franco Donati  @institution: Leiden University CML

```python
class pycirk.pycirk_settings.Settings (method=0, make_secondary=False, 
save_directory=", aggregation=1, file=None, 
test=False)
```

Bases: object

This class allows for to specify the settings for pycirk.

**method**  [int] SUTs to IO transformation methods

0 = Prod X Prod Ind-Tech Assumption Technical Coeff method

1 = Prod X Prod Ind-Tech Assumption Market Share Coeff method

**make_secondary**  [bool] modifies SUT so that secondary technologies which process scrap materials into primary materials are also available in the IO tables

False = Don’t modify

True = Modify

**save_directory**  [str] directory in which you want to work and save your results

aggregation : int, bool

0 = None (multi-regional 49 regions)

1 = bi-regional (EU- ROW)

**file**  [bool, str] allows you to specify where the dataset is placed. None will use the default location within the installed package

**test**  [bool] if set to true it will run the test settings under under pycirk//tests

**assign_labels_to_class()**

**check_dataset_location()**

It identifies where the baseline dataset is located and whether it is present in the directory.

A dictionary containing location “loc” and type of format (SUT or IO)
check_expand_directory(directory)
    Checking that we are using the right directory for the right OS

create_output_folder()
    It creates an output folder where to save analytical results This is placed in the user’s working directory

create_scenario_file()
    It creates a new scenario file by copying the original from pycirk directory to the new working directory
    specified by the user

load_dataset(data)
    It loads the baseline dataset

load_results_params()

number_scenarios()

project_specs(test=False)
    General labelling of the project

scenario_file()
    It returns where the working scenarios.xlsx file is located

set_IO_scenario(data, scen_no)

set_SUTs()

set_SUTs_scenario(data, scen_no)

set_save_directory()
    It specifies where the scenario file for input is

transform_to_io()
    Transforms the SUT dataset into an IO system
    If the user specified to make secondary material processing apparent then it will launch the function to
    modify the database
    If an IO from the same transformation method exists, then it will load that one instead
    If a pre-existing IO is not present then it will save the transformed dataset in the directory. This is done to
    speed up processing time.
    An object containing all the fundamental IO matrices to begin the analysis and scenarios.

4.1.11 pycirk.results module

Created on Mon Feb 27 09:26:43 2017

Description: module to perform results analysis

Scope: Modelling the Circular Economy in EEIO

@author: Franco Donati @institution: Leiden University CML, TU Delft TPM

pycirk.results.iter_thru_for_results(data, analysis_specs, scen_no, labels)
    filter policy interventions from scenario file according to specified matrix and return the respective matrix with
    it Output only specified (see scenario.xls) results to be analysed

pycirk.results.retrieve_specified_data(data, spec_row, labels)
    Separate, collect and rename results for base and scenarios according to specifications under th sheet “analysis”
    in scenarios.xls
    data = any IOT table spec_row = row in the scenarios sheet specifying settings
4.1.12 pycirk.save_utils module

Created on Sat Mar 4 11:02:52 2017

Description: Save data to xls

Scope: Modelling circular economy policies in EEIOA

@author: Franco Donati @institution: Leiden University CML

pycirk.save_utils.add_date_to_gen_specs(specs)
   Adds timemark to general specifications e.g. authors, institution etc

pycirk.save_utils.save_outputs(results, directory, specs, scen_no, data=None)
   It saves results into a previously specified directory
   
   results [pandas.DataFrame] a dataframe containing only selected results
   
   directory [str] the general location specified for your scenarios.xlsx file
   
   specs [dict] a dictionary containing general info about the project (e.g. author, institution, etc)
   
   scen_no [int or None] an integer specifying the scenario number or enter None to save all results
   
   data [dict of pd.DataFrame] a completely new dataset to be pickled. Default value is None otherwise pass the dataset

   scenarios.xlsx [excel file] scenario settings excel file used for the analysis in the same output directory with the results

   info_and_results.xlsx [excel file] excel file containing general info about project plus the results from the analysis

   data.pkl [pickle file] new modified IO dataset in pickle format

4.1.13 pycirk.transformation_methods module

Created on Mon Feb 6 12: 29: 47 2017

Description: Uses methods within SUTops to calculate IOT and Extensions

Scope: Modelling the Circular economy in EEIO

@author: Franco Donati @institution: Leiden University CML

class pycirk.transformation_methods.Transform(SUTs)
   Bases: object
   Transforms SUTs to IOT an calculuate extensions

   IOTpxpSTA_MSCm()
      IOT prod x prod Single tech Industry-technology assumption Market share coef method

   IOTpxpSTA_TCM()
      IOT prod x prod Single tech Industry-technology assumption Technical coef method

4.1.14 Module contents

Top-level package for pycirk.
Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given. You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs


If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.
5.1.4 Write Documentation

pycirk could always use more documentation, whether as part of the official pycirk docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/FDonati/pycirk/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome ;)

5.2 Get Started!

Ready to contribute? Here’s how to set up pycirk for local development.

1. Fork the pycirk repo on GitHub.
2. Clone your fork locally:
   
   ```
   $ git clone git@github.com:your_name_here/pycirk.git
   ```
3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

   ```
   $ mkvirtualenv pycirk
   $ cd pycirk/
   $ python setup.py develop
   ```
4. Create a branch for local development:

   ```
   $ git checkout -b name-of-your-bugfix-or-feature
   ```
   Now you can make your changes locally.
5. When you’re done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

   ```
   $ flake8 pycirk tests
   $ python setup.py test or py.test
   $ tox
   ```
   To get flake8 and tox, just pip install them into your virtualenv.
6. Commit your changes and push your branch to GitHub:

   ```
   $ git add .
   $ git commit -m "Your detailed description of your changes."
   $ git push origin name-of-your-bugfix-or-feature
   ```
7. Submit a pull request through the GitHub website.
5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/FDonati/pycirk/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

$ py.test tests.test_pycirk

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags

Travis will then deploy to PyPI if tests pass.
6.1 Development Lead

- Franco Donati <f.donati@cml.leidenuniv.nl>

6.2 Contributors

None yet. Why not be the first?
7.1 0.1.0 (2018-05-11)

- First release on PyPI.
Indices and tables

- genindex
- modindex
- search
<table>
<thead>
<tr>
<th>Module</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>pycirk</td>
<td>19</td>
</tr>
<tr>
<td>pycirk.cli</td>
<td>9</td>
</tr>
<tr>
<td>pycirk.fundamental_operations</td>
<td>9</td>
</tr>
<tr>
<td>pycirk.labels</td>
<td>11</td>
</tr>
<tr>
<td>pycirk.make_scenarios</td>
<td>12</td>
</tr>
<tr>
<td>pycirk.make_secondary_flows</td>
<td>14</td>
</tr>
<tr>
<td>pycirk.organize_io</td>
<td>14</td>
</tr>
<tr>
<td>pycirk.positions</td>
<td>15</td>
</tr>
<tr>
<td>pycirk.pycirk</td>
<td>15</td>
</tr>
<tr>
<td>pycirk.pycirk_settings</td>
<td>17</td>
</tr>
<tr>
<td>pycirk.results</td>
<td>18</td>
</tr>
<tr>
<td>pycirk.save_utils</td>
<td>19</td>
</tr>
<tr>
<td>pycirk.transformation_methods</td>
<td>19</td>
</tr>
</tbody>
</table>
Index

A
A() (pycirk.fundamental_operations.Operations.IOT method), 9
A() (pycirk.fundamental_operations.Operations.PxP_ITA_MSC method), 10
add_date_to_gen_specs() (in module pycirk.save_utils), 19
all_results() (pycirk.pycirk.Launch method), 16
allocate_sec_mat() (in module pycirk.make_secondary_flows), 14
apply_labels() (pycirk.labels.Labels method), 11
assign_labels_to_class() (pycirk.pycirk_settings.Settings method), 17

B
B() (pycirk.fundamental_operations.Operations.IOT method), 9
B() (pycirk.fundamental_operations.Operations.PxP_ITA_MSC method), 10
B() (pycirk.fundamental_operations.Operations.PxP_ITA_TC method), 10
balancing_operation() (in module pycirk.make_scenarios), 12
basic_add() (in module pycirk.make_scenarios), 12
basic_mult() (in module pycirk.make_scenarios), 12
bY() (pycirk.fundamental_operations.Operations.IOT method), 10

c
calc_no_of_something() (pycirk.labels.Labels method), 11
calculate_characterized() (pycirk.fundamental_operations.Operations method), 11
check_dataset_location() (pycirk.pycirk_settings.Settings method), 17
check_expand_directory() (pycirk.pycirk_settings.Settings method), 17
counterfactual() (in module pycirk.make_scenarios), 12
counterfactual_engine() (in module pycirk.make_scenarios), 12
create_output_folder() (pycirk.pycirk_settings.Settings method), 18
create_scenario_file() (pycirk.pycirk_settings.Settings method), 18

D
D() (pycirk.fundamental_operations.Operations.PxP_ITA_MSC method), 10
delta_x() (pycirk.fundamental_operations.Operations method), 11
delta_y() (pycirk.fundamental_operations.Operations method), 11

g
get_labels() (pycirk.labels.Labels method), 11
get_unique_labels() (pycirk.labels.Labels method), 11

I
identify_labels() (pycirk.labels.Labels method), 11
inv() (pycirk.fundamental_operations.Operations method), 11
IOT() (pycirk.fundamental_operations.Operations.IOT method), 9
IOTpxpSTA_MSCm() (pycirk.transformation_methods.Transform method), 19
IOTpxpSTA_TCm() (pycirk.transformation_methods.Transform method), 19
iter_thru_for_results() (in module pycirk.results), 18

L
L() (pycirk.fundamental_operations.Operations.IOT method), 9
V

verifyIOT() (pycirk.fundamental_operations.Operations-method), 11

X

×() (pycirk.fundamental_operations.Operations.IOT-method), 10

×_IAY() (pycirk.fundamental_operations.Operations.IOT-method), 10

Z

Z() (pycirk.fundamental_operations.Operations.IOT-method), 10

Z() (pycirk.fundamental_operations.Operations.PxP_ITA_MSC-method), 10

Z() (pycirk.fundamental_operations.Operations.PxP_ITA_TC-method), 11