simrel Documentation

Release 0.5.5

Raju Rimal

Sep 01, 2022

CONTENTS

PySimrel package provides a way to simuilate data from a linear model. It provides few sets of parameters that users can specify to simulate the data. With these few sets of parameters, user can get wide range of properties in the simulated data.

CHAPTER

ONE

CONTENTS:

1.1 Modules

1.1.1 Examples

```
>>> sobj = Simrel(n_pred = 10, n_relpred = '4, 5', pos_relcomp = '0, 1; 2, 3, 4',
    gamma = 0.7, rsq = '0.7, 0.8', n_resp = 4, eta = 0.7, pos_resp = '0, 2; 1, 3')
```

```
>>> print(sobj.properties)
Numpy Arrays:
---------------------------------------------
eigen_x:                 Shape: (10,)
eigen_y:                 Shape: (4,)
rotation_x:              Shape: (10, 10)
rotation_y:              Shape: (4, 4)
sigma_latent:            Shape: (14, 14)
sigma:                   Shape: (14, 14)
rsq:                     Shape: (4, 4)
rsq_w:                   Shape: (4, 4)
minerror:                Shape: (4, 4)
beta_z:                  Shape: (10, 4)
beta:                    Shape: (10, 4)
beta0:                   Shape: (4,)
Dictionaries:
---------------------------------------------
relevant_predictors:     Keys: rel, irrel
```

```
>>> print(sobj.covariances)
    Numpy Arrays:
    +-------------+-------------------+
    |             |                   |
    |    cov_ww   |      cov_wz       |
    |    cov_yy   |      cov_xy       |
    |    (4, 4)   |     (4, 10)       |
    |             |                   |
    +-------------+-------------------+
    |             |                   |
    |             |                   |
    |    cov_zw   |      cov_zz       |
```

(continues on next page)

(continued from previous page)

```
    |    cov_xy   |       cov_xx       |
    |   (10, 4)   |      (10, 10)      |
    |             |                    |
    |             |                    |
    +-------------+--------------------+
```

Simrel is a package for simulating linear model data

1.2 Classes

1.2.1 Simrel Class

class Simrel(*n_pred: Union[str, int] = 10*, *n_relpred: Union[str, int] = '4, 5'*, *pos_relcomp: Union[str, int] = '0, 1; 2, 3, 4'*, *gamma: float = 0.7*, *rsq: Union[str, int] = '0.7, 0.8'*, *n_resp: Union[str, int] = 4*, *eta: float = 0.7*, *pos_resp: Union[str, int] = '0, 2; 1, 3'*, *mu_x: Optional[Union[str, int]] = None*, *mu_y: Optional[Union[str, int]] = None*, *parameter_parsed: bool = False*, *properties_computed: bool = False*)

Main Class for simulated objects

The class contains all the definitions of *simrel* objects. The class will also provide necessary methods to compute various population properties.

```
>>> sobj = Simrel(n_pred = 10, n_relpred = '4, 5', pos_relcomp = '0, 1; 2, 3, 4',
    gamma = 0.7, rsq = '0.7, 0.8', n_resp = 4, eta = 0.7, pos_resp = '0, 2; 1, 3')
```

```
>>> print(sobj.properties)
Numpy Arrays:
---------------------------------------------
eigen_x:                 Shape: (10,)
eigen_y:                 Shape: (4,)
rotation_x:              Shape: (10, 10)
rotation_y:              Shape: (4, 4)
sigma_latent:            Shape: (14, 14)
sigma:                   Shape: (14, 14)
rsq:                     Shape: (4, 4)
rsq_w:                   Shape: (4, 4)
minerror:                Shape: (4, 4)
beta_z:                  Shape: (10, 4)
beta:                    Shape: (10, 4)
beta0:                   Shape: (4,)
Dictionaries:
---------------------------------------------
relevant_predictors:     Keys: rel, irrel
```

```
>>> print(sobj.covariances)
    Numpy Arrays:
    +-------------+--------------------+
    |             |                    |
    |    cov_ww   |       cov_wz       |
    |    cov_yy   |       cov_xy       |
```

(continues on next page)

(continued from previous page)

```
    |    (4, 4)   |      (4, 10)      |
    |             |                   |
    +-------------+-------------------+
    |             |                   |
    |             |                   |
    |    cov_zw   |       cov_zz      |
    |    cov_xy   |       cov_xx      |
    |   (10, 4)   |      (10, 10)     |
    |             |                   |
    |             |                   |
    +-------------+-------------------+
```

`n_pred`

Number of predictor variables. Ex: *n_pred: 10*

Type
Either integer or string

`n_relpred`

Number of relevant predictor variables for each response components In the case of single response model, the parameters refers to the number of predictors relevant for that single response

Type
Either integer or string

`parse_parameters()`

Parse the parameters passed during initialization This method parse the parameters which are passed as string into a nested list. It uses `parse_parm()` function where further documentation can be found.

1.2.2 Helper Classes

`class Covariances`

Class defining various covariances of the simulated data

This provides a nice graphical output of covariances.

Parameters

- **cov_ww** (*np.ndarray*) – Covariance matrix of latent components of response
- **cov_zz** (*np.ndarray*) – Covariance matrix of latent components of predictors
- **cov_zw** (*np.ndarray*) – Covariance matrix containing covariances between latent components of predictors and response
- **cov_yy** (*np.ndarray*) – Covariance matrix of response
- **cov_xx** (*np.ndarray*) – Covariance matrix of response
- **cov_xy** (*np.ndarray*) – Covariance matrix containing covariances between predictors and response

`class Properties`

A data class for different properties of simulated object

Parameters

- **eigen_x** (*np.ndarray*) – Eigenvalues corresponding to predictors

- **eigen_y** (*np.ndarray*) – Eigenvalues corresponding to responses
- **relevant_predictors** (*np.ndarray*) – Position index of relevant predictors for each responses
- **sigma_latent** (*np.ndarray*) – Variance-Covariance matrix of latent components of predictors and Responses
- **sigma** (*np.ndarray*) – Variance-Covariance matrix of predictors and Responses
- **beta_z** (*np.ndarray*) – Regression coefficient corresponding to the principal components of predictors
- **beta** (*np.ndarray*) – Regression coefficient corresponding to the predictor variables
- **beta0** (*np.ndarray*) – Regression Intercept
- **rsq_w** (*np.ndarray*) – Coefficient of determination for latent component of responses (Variation explained by latent components of predictors on latent components of response)
- **rsq** (*np.ndarray*) – Coefficient of determination for responses (Variation explained by predictors on response)
- **minerror** (*np.ndarray*) – True minimum model error
- **rotation_x** (*np.ndarray*) – Rotation Matrix (eigenvector matrix) corresponding to predictors
- **rotation_y** (*np.ndarray = None*) – Rotation Matrix (eigenvector matrix) corresponding to response

`class Data`(*X: numpy.ndarray*, *Y: numpy.ndarray*)

1.3 Utilities Functions

`get_cov`(*rel_pos*, *rsq*, *kappa*, *lmd*, *random_seed=None*)

Compute Covariances

Compute covariances at the position specified in `rel_pos` recursively using the function `sample_cov` satisfying the `rsq` and the eigen values in `kappa` and `lmd`.

Parameters

- **rel_pos** (*list*) – position of relevant components
- **rsq** (*list*) – A list of coefficient of determination
- **kappa** (*list*) – A list of eigenvalues related to response variables
- **lmd** (*list*) – A list of eigenvalues related to predictor variables
- **random_seed** (*int*) – An integer for random state

Returns

A matrix of dimension equals to the length of `kappa` by length of `lmd` with computed covariances at the position specified in `rel_pos`.

Return type

np.array

`get_eigen`(*rate*, *nvar*, *min_value=0.0001*)

Compute eigen values using exponential decay function.

$$\lambda_i = \exp^{-\gamma(i-1)}$$

Parameters

- **`rate`** – rate of exponential decay factor
- **`nvar`** – Number of variables (number of eigenvalues to compute)
- **`min_value`** – Lower limit for smallest eigenvalue

Returns

A list of eigenvalues

`get_relpred`(*n_pred*, *n_relpred*, *pos_relcomp*, *random_state=None*)

Identify relevant predictors through sampling

Get relevant and irrelevant position of predictor variables. The irrelevant components index are the one which are not in `pos_relcomp`. The number of extra components are defined in `n_relpred`.

Parameters

- **`n_pred`** (*`int`*) – Number of predictor variables
- **`n_relpred`** (*`list`*) – List of number of predictors relevant for each response
- **`pos_relcomp`** (*`list`*) – List of List containing the position index of relevant components
- **`random_state`** (*`int`*) – An integer for random state

Returns

A dictionary with relevant and irrelevant position index of predictors

Return type

dict

`get_rotate`(*mat*, *pred_pos*, *random_state=None*)

Fill up a block of matrix `mat` based on position index in `pred_pos`. The block will be an orthogonal rotation matrix.

Parameters

- **`mat`** (*`np.array`*) – A matrix possibly a square matrix as covariance
- **`pred_pos`** (*`list`*) – A list of position index for the block rotation
- **`random_state`** (*`int`*) – An integer for random state to control randomness

Returns

A matrix of same size as `mat` but filled with an orthogonal block

Return type

np.array

`get_rotation`(*rel_irrel_pred*, *random_state=None*)

Create orthogonal rotation matrix

Creates an orthogonal rotation matrix from dictionary of relevant and irrelevant positions using *get_rotate* function.

Parameters

`rel_irrel_pred` (*`dict`*) – A dictionary of relevant and irrelevant position (possibly obtained from the function *get_relpred*.

Returns
An orthogonal rotation matrix

Return type
np.array

`parse_param`(*parm: Optional[Union[str, int]]*)

Parse the parameters from string to a nested list

Parameters
`parm` (*`str, int`*) – Either integer, float (in some cases) or mostly string

Returns
A nested list of parsed parameters

Return type
list

`sample_cov`(*lmd*, *rsq*, *pos*, *kappa*, *alpha_*)

Compute covariance satisfying given parameters

Compute covariance from a sample of uniform distribution satisfying *rsq*, a set of *lmd* and *kappa*

Parameters

- **`lmd`** (*`set or list`*) – A set of eigenvalue of predictors at position specified by `pos`.
- **`rsq`** (*`float`*) – Coefficient of determination
- **`pos`** (*`list`*) – Position index of in which covariance need to be non-zero
- **`kappa`** (*`list`*) – Eigenvalue corresponding to response (univariate) or response component (multivariate)
- **`alpha`** – A sample from univariate distribution between -1 and 1

Returns
An array of computed covariances of length equals to `lmd`.

Return type
np.array

`sample_extra_pos`(*rs*, *n_extra_pos*, *extra_pos*, *irrel_pos*)

Sample Extra Position Required

Sample position index of extra relevant predictors from irrelevant predictors in `irrel_pos`.

Parameters

- **`rs`** (*`np.random.mtrand.RandomState`*) – A numpy RandomSeed object
- **`n_extra_pos`** (*`int`*) – An integer for number of extra position index to sample
- **`extra_pos`** (*`list`*) – A list container for collecting extra relevant components
- **`irrel_pos`** (*`list`*) – A list or set of irrelevant position indices

Returns
a list of relevant and irrelevant position indices

Return type
list

CHAPTER

TWO

INDICES AND TABLES

PYTHON MODULE INDEX

INDEX