

---

# **simrel Documentation**

***Release 0.6.1***

**Raju Rimal**

**Sep 01, 2022**



# CONTENTS

<b>1</b>	<b>Contents:</b>	<b>3</b>
1.1	Modules . . . . .	3
1.2	Classes . . . . .	4
1.3	Utilities Functions . . . . .	6
<b>2</b>	<b>Indices and tables</b>	<b>9</b>
	<b>Python Module Index</b>	<b>11</b>
	<b>Index</b>	<b>13</b>



PySimrel package provides a way to simulate 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.



## 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 (10, 4)	cov_xx (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.

```
>>> subj = 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_VY		COV_XV

(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

## INDICES AND TABLES



## PYTHON MODULE INDEX

### p

`pysimrel`, [4](#)

`pysimrel.utilities`, [6](#)





## C

Covariances (*class in pysimrel*), 5

## D

Data (*class in pysimrel*), 6

## G

get\_cov() (*in module pysimrel.utilities*), 6

get\_eigen() (*in module pysimrel.utilities*), 6

get\_relpred() (*in module pysimrel.utilities*), 7

get\_rotate() (*in module pysimrel.utilities*), 7

get\_rotation() (*in module pysimrel.utilities*), 7

## M

module

    pysimrel, 4

    pysimrel.utilities, 6

## N

n\_pred (*Simrel attribute*), 5

n\_relpred (*Simrel attribute*), 5

## P

parse\_param() (*in module pysimrel.utilities*), 8

parse\_parameters() (*Simrel method*), 5

Properties (*class in pysimrel*), 5

pysimrel

    module, 4

pysimrel.utilities

    module, 6

## S

sample\_cov() (*in module pysimrel.utilities*), 8

sample\_extra\_pos() (*in module pysimrel.utilities*), 8

Simrel (*class in pysimrel*), 4