Hypermodern Screening

Tobias Stenzel

Apr 06, 2020

CONTENTS

CHAPTER

ONE

LICENSE

MIT License

Copyright (c) 2020 Tobias

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

CHAPTER

TWO

REFERENCE

2.1 hypermodern_screening.sampling_schemes

Sampling parameter vectors tailored to EE computations.

hypermodern_screening.sampling_schemes.**morris_trajectory**(*n_inputs*, *n_levels*, *seed=123*, *normal=False*, *numeric_zero=0.01*, *step_function=<function stepsize>*, *stairs=True*)

Create random sample in trajectory design.

This function creates a random sample for a number of function parameters (columns). The sample itself consists of the number plus one vectors of parameter draws (rows). It also computes the steps taken by each element.

Parameters

- **n_inputs** (int) – Number of input paramters / columns / rows - 1.
- **n_levels** (int) – Number of distict grid points.
- **seed** (int) – Random seed.
- **normal** (bool) – Indicates whether to transform points by *scipy.normal.ppt*
- **numeric_zero** (float) – *if normal is True*: Prevents *scipy.normal.ppt* to return *-Inf* and *Inf* for 0 and 1.
- **step_function** (Callable) – Constant step as function of *n_levels* added to lower half of point grid.
- **stairs** (bool) – if False: Randomly shuffle columns, dissolves stairs shape.

Return type Tuple[ndarray, ndarray]

Returns

- *B_random* – Random sample in trajectory design. Dimension *n_inputs* x *n_inputs + 1*.
- *trans_steps* – Column vector of steps added to base value point. Sorted by parameter/column. Dimension *n_inputs* x *1*.

See also:

stepsize() See parameter *step_function*.

transform_uniform_stnormal_uncorr() See parameter *numeric_zero*.

Notes

The method is described in [1]. This function follows the notation therein. The idea is tailored to compute a random sample of function arguments to compute local derivates. First, a random row of paramters is drawn. Then, one parameter is changed by a fixed step in each row. The local derivatives can be computed by subtracting the function evaluations of each row from its upper row, thereby obtaining one local derivative for each parameter. The order of rows and columns may be shuffled. Shuffling rows creates a negative stepsize. By default, the shuffling of columns is turned off to facilitate post-processing. Importantly, an additional option is to evaluate the points by the inverse normal cdf to account for normally distributed input paramters vice versa uniformly distributed ones. For this purpose, zeros and ones are slighly shifted towards the centre of [0,1], so that no infinite values arise. Given the shape of the inverse cdf, the specific transformation choice has large influences on the stepsize and therefore the Elementary Effects. To account for transformations, the step is recomputed for each parameter by subtracting the last first row from the last row.

References

[1] Morris, M. D. (1991). Factorial sampling plans for preliminary computational experiments. Technometrics 33 (2), 161–174.

hypermodern_screening.sampling_schemes.**radial_sample**(*n_rad*, *n_inputs*, *normal=False*, *sequence='S'*)

Generate sample in radial design as described in [1].

For each subsample, there are *n_inputs + 1* rows and *n_inputs* colums. Each row is identical except of the diagonal of the sample w/o the first row.

Parameters

- **n_rad** (int) – Number of subsamples.
- **n_inputs** (int) – Number of input paramters / columns / rows - 1.
- **seed** – Random seed.
- **normal** (bool) – Indicates whether to transform points by *scipy.normal.ppt*
- **numeric_zero** – *if normal is True*: Prevents *scipy.normal.ppt* to return *-Inf* and *Inf* for 0 and 1.
- **sequence** (str) – Type of quasi-random sequence.

Return type Tuple[List[ndarray], List[ndarray]]

Returns

- *sample* – Random sample in radial design. Dimension *n_inputs* x *n_inputs + 1*.

- *trans_steps* – Column vector of steps added to base value point. Sorted by parameter/column. Dimension *n_inputs* x *1*.

Notes

See [2] for abbreviations of the different sequence types. In contrary to the trajectory design, the stepsize differs right from the start by design and only one element changes in each row compared to the first row. All distict elements in the whole sample are drawn at once because the default Sobol' sequence can not be reseeded.

References

[1] Ge, Q. and M. Menendez (2017). Extending morris method for qualitative global sensitivityanalysis of models with dependent inputs. Reliability Engineering & System Safety 100 (162), 28–39. [2] <https://github.com/jonathf/chaospy/blob/master/chaospy/distributions/sampler/ generator.py#L62>

`hypermodern_screening.sampling_schemes.`**`stepsize`**(*n_levels*)

Compute stepsize to create equiprobable sample points for the traj. design.

Parameters **`n_levels`** (`int`) – Number of points in a trajectory sample.

Returns Step added to each lower half point of the point grid.

Return type step

Raises **`AssertionError`** – If the number of levels is not an even integer.

Notes

This function, published in [1], assumes that the number of sample points called "levels" is an even integer. The first row in the trajectory is initialized with the lower half of the desired equispaced points between 0 and 1. Given the below formula, the step added to the lowest, second lowest, . . . , highest point in the lower half creates the lowest, second lowest, . . . , highest point in the upper half of the point grid.

References

[1] Morris, M. D. (1991). Factorial sampling plans for preliminary computational experiments. Technometrics 33 (2), 161–174.

`hypermodern_screening.sampling_schemes.`**`trajectory_sample`**(*n_traj*, *n_inputs*, *n_levels*, *seed=123*, *normal=False*, *numeric_zero=0.01*, *step_function=<function stepsize>*, *stairs=True*)

Loops over *morris_sample*.

Parameters

- **`n_inputs`** (`int`) – Number if input paramters.
- **`n_levels`** (`int`) – Number of distict grid points.
- **`seed`** (`int`) – Random seed.
- **`normal`** (`bool`) – Indicates whether to transform points by *scipy.normal.ppt*

- **numeric_zero** (float) – *if normal is True*: Prevents *scipy.normal.ppt* to return *-Inf* and *Inf* for 0 and 1.
- **step_function** (Callable) – Constant step as function of *n_levels* added to lower half of point grid.
- **stairs** (bool) – if False: Randomly shuffle columns, dissolves stairs shape.

Return type Tuple[List[ndarray], List[ndarray]]

Returns

- *sample_traj_list* – Set of trajectories.
- *steps_list* – Set of steps taken by each base row.

2.2 hypermodern_python.select_sample_set

Decrease a set of sample sets in order to increase its representativeness.

These approaches are developed in the context of the trajectory design because it can not cover the space very densely. The methods are taken from the effective screening design in [1] and the efficient screening design in [2].

References

[1] Campolongo, F., J. Cariboni, and A. Saltelli (2007). An effective screening design for sensitivity analysis of large models. Environmental modelling & software 22 (10), 1509–1518. [2] Ge, Q. and M. Menendez (2014). An efficient sensitivity analysis approach for computationally expensive microscopic traffic simulation models. International Journal of Transportation 2 (2), 49–64.

hypermodern_screening.select_sample_set.**campolongo_2007**(*sample_traj_list*, *n_traj*)

Implement the post-selected sample set in [1].

Takes a list of Morris trajectories and selects the *n_traj* trajectories with the largest distance between them. Returns the selection as array with n_inputs at the verical and n_traj at the horizontal axis and as a list. It also returns the diagonal matrix that contains the pair distance between each trajectory pair.

Parameters

- **sample_traj_list** (*list of ndarrays*) – Set of samples.
- **n_traj** (*int*) – Number of samples to choose from *sample_traj_list*.

Return type Tuple[List, ndarray, List]

Returns

- **sample_traj_list** (*list of ndarrays*) – Set of trajectories.
- **select_dist_matrix** (*ndarray*) – Symmetric *distance_matrix* of selection.
- **select_indices** (*list*) – Indices of selected samples.

hypermodern_screening.select_sample_set.**combi_wrapper**(*iterable*, *r*)

Wrap *itertools.combinations*, written in C, see [1].

Parameters

- **iterable** (*iterable object*) – Hashable container like a list of distinct elements to combine.
- **r** (*int*) – Number to draw from *iterable* with putting back and regarding the order.

Returns All possible combinations in ascending order.

Return type list_list

Example

```
>>> combi_wrapper([0, 1, 2, 3], 2)
[[0, 1], [0, 2], [0, 3], [1, 2], [1, 3], [2, 3]]
```

References

[1] https://docs.python.org/2/library/itertools.html#itertools.combinations.

hypermodern_screening.select_sample_set.**compute_pair_distance**(*sample_0*, *sample_1*)

Compute the distance measure between a pair of samples.

The aggregate distance between sum of the root of the square distance between each parameter vector of one sample to each vector of the other sample.

Parameters

- **sample_0** (ndarray) – Sample with paramters in cols and draws as rows.
- **sample_1** (ndarray) – Sample with paramters in cols and draws as rows.

Returns Pair distance.

Return type distance

Raises

- **AssertionError** – If sample is not in trajectory or radial design shape.
- **AssertionError** – If the sample shapes differ.

Notes

The distance between two samples is sum of the root of the square distance between each parameter vector of one sample to each vector of the other sample.

hypermodern_screening.select_sample_set.**distance_matrix**(*sample_list*)

Compute symmetric matrix of pair distances for a list of samples.

Parameters **sample_list** (List[ndarray]) – Set of samples.

Returns Symmatric matrix of pair distances.

Return type distance_matrix

hypermodern_screening.select_sample_set.**final_ge_menendez_2014**(*sample_traj_list*, *n_traj*)

Implement both "improvements" in [2] vis-a-vis [1].

Parameters

- **sample_traj_list** (List[ndarray]) – Set of samples.
- **n_traj** (int) – Number of samples to choose from *sample_traj_list*.

Return type Tuple[List[ndarray], ndarray, List[int]]

Returns

- *sample_traj_list* – Set of trajectories.
- *select_dist_matrix* – Symmetric *distance_matrix* of selection.
- *select_indices* – Indices of selected samples.

See also:

next_combi_total_distance_gm14()

Notes

This function, is in fact much slower than *intermediate_ge_menendez_2014* because it uses more for loops to get the pair distances from the right combinations that must be subtracted from the total distances. This function selects n_traj trajectories from n_traj_sample trajectories by iteratively selecting n_traj_sample - i for i = 1,. . . ,n_traj_sample - n-traj. For this purpose, next_combi_total_distance_gm14 computes the total distance of each trajectory combination by using the total distance of each combination in the previous step and subtracting each pair distance with the dropped trajectory, that yielded the lowest total distance combinations in the previous step.

hypermodern_screening.select_sample_set.**intermediate_ge_menendez_2014**(*sample_traj_list*, *n_traj*)

Implement the essential of the two "improvements" in[2] vis-a-vis [1].

This is basically a wrapper around *select_trajectories_wrapper_iteration*.

Parameters

- **sample_traj_list** (*list of ndarrays*) – Set of samples.
- **n_traj** (*int*) – Number of samples to choose from *sample_traj_list*.

Return type Tuple[List, ndarray, List]

Returns

- **sample_traj_list** (*list of ndarrays*) – Set of trajectories.
- **select_dist_matrix** (*ndarray*) – Symmetric *distance_matrix* of selection.
- **select_indices** (*list*) – Indices of selected samples.

See also:

select_trajectories_wrapper_iteration()

Notes

Oftentimes this function leads to diffent combinations than *select_trajectories*. However, their total distance is very close to the optimal solution.

hypermodern_screening.select_sample_set.**next_combi_total_distance_gm14**(*combi_total_distance*, *pair_dist_matrix*, *lost_index*)

Select the set of samples minus one sample.

Based on the algorithmic computation of the *total_distance* proposed by [2]. I.e. by re-using and adjusting the first *combi_total_distance* matrix each iteration. Used for selecting iteratively rather than by brute force.

Parameters

- **combi_total_distance_next** – Matrix with n_traj + 1 rows. The first n_traj cols are filled with indices of samples and the last column is the *total_distance* of the combinations of samples marked by indices in the same row and the columns before.
- **pair_dist_matrix** (ndarray) – Distance matrix of all combinations and their total_distance.
- **lost_index** (int) – index of the sample that will be dropped from the samples in the above objects.

Return type Tuple[ndarray, ndarray, ndarray]

Returns

- *combi_total_distance_next* – *combi_total_distance* without the dropped sample.
- *pair_dist_matrix_next* – *pair_dist_matrix* without the dropped sample.
- *lost_index* – *lost_index* without the dropped sample one iteration before.

Notes

The function computes the total distance of each trajectory combination by using the total distance of each combination in the previous step and subtracting each pair distance with the dropped trajectory, that yielded the lowest total distance combinations in the previous step. This function, is in fact much slower than *select_trajectories_wrapper_iteration* because it uses more for loops to get the pair distances from the right combinations that must be subtracted from the total distances.

hypermodern_screening.select_sample_set.**select_sample_set_normal**(*samp_list*, *n_select*, *numeric_zero*)

Post-select set of samples based on [0,1] and transform it to stnormal space.

Parameters

- **samp_list** (List[ndarray]) – Sub-samples.
- **n_select** (int) – Number of sub-samples to select from *samp_list*.
- **numeric_zero** (float) – *if normal is True*: Prevents *scipy.normal.ppt* to return *-Inf* and *Inf* for 0 and 1.

Return type Tuple[List[ndarray], List[ndarray]]

Returns

- *samp_list*
- *steps_list*

Notes

Function for post-selection is *intermediate_ge_menendez_2014* because it is the fastest.

See also:

intermediate_ge_menendez_2014()

hypermodern_screening.select_sample_set.**select_trajectories**(*pair_dist_matrix*, *n_traj*)

Compute *total distance* for each *n_traj* combinations of a set of samples.

Parameters

- **pair_dist_matrix** (ndarray) – *distance_matrix* for a sample set.
- **n_traj** (int) – Number of sample combinations for which the *total_distance* is computed.

Return type Tuple[List, ndarray]

Returns

- **max_dist_indices** (*list of ints*) – Indices of samples in *pair_dist_matrix* that are part of the combination with the largest *total_distance*.
- **combi_total_distance** (*ndarray*) – Matrix with n_traj + 1 rows. The first n_traj cols are filled with indices of samples and the last column is the *total_distance* of the combinations of samples marked by indices in the same row and the columns before.

Raises

- **AssertionError** – If *pair_dist_matrix* is not symmetric.
- **AssertionError** – If the number of combinations does not correspong to the combinations indicated by the size of *pair_dist_matrix*.

Notes

This function can be very slow because it computes distances between np.binomial(len(pair_dist_matrix, n_traj) pairs of trajectories. Example: *np.biomial(30,15)* = 155117520. This selection function yields precise results because each total distance for each possible combination of trajectories is computed directly. The faster, iterative methods can yield different results that are, however, close in the total distance. The total distances tend to differentiate clearly. Therefore, the optimal combination is precisely determined.

hypermodern_screening.select_sample_set.**select_trajectories_wrapper_iteration**(*pair_dist_matrix*, *n_traj*)

Select the set of samples minus one sample.

Used for selecting iteratively rather than by brute force. Implements the main step of the essential of the two "improvements" from [2] to [1].

Parameters

- **pair_dist_matrix** (*ndarray*) – Distance matrix of all combinations and their total_distance.
- **n_traj** (*int*) – number of samples to choose from a set of samples based on their *total_distance*.

Return type Tuple[List, ndarray]

Returns

- **tracker_keep_indices** (*list*) – Indices of samples part of the selection.
- **combi_total_distance** (*ndarray*) – Matrix with n_traj + 1 rows. The first n_traj cols are filled with indices of samples and the last column is the *total_distance* of the combinations of samples marked by indices in the same row and the columns before.

See also:

select_trajectories()

Notes

Oftentimes this function leads to diffent combinations than *select_trajectories*. The reason seems to be that this function deviates from the optimal path due to numerical reasons as different combinations may be very close (see [2]). However, the total sum of the returned combinations are close. Therefore, the *total_distance* loss is negligible compared to the speed gain for large numbers of trajectory combinations. This implies that, *combi_total_distance* always differs from the one in *select_trajectories* because it only contains the combination indices from the last iteration if n_traj is smaller than the sample set minus 1. The trick using *tracker_keep_indices* is an elegant solution.

hypermodern_screening.select_sample_set.**total_distance**(*distance_matrix*)

Compute the total distance measure of all pairs of samples in a set.

The equation corresponds to Equation (10) in [2].

Parameters **distance_matrix** (*ndarray*) – diagonal matrix of distances for sample pairs.

Returns **total_distance** – total distance measure of all pairs of samples in a set.

Return type float

2.3 hypermodern_python.transform_distributions

Functions for the inverse Rosenblatt / inverse Nataf transformation (u to z_c).

hypermodern_screening.transform_distributions.**covariance_to_correlation**(*cov*)

Convert covariance matrix to correlation matrix.

Parameters **cov** (ndarray) – Covariance matrix.

Returns Correlation matrix.

Return type corr

hypermodern_screening.transform_distributions.**transform_stnormal_normal_corr**(*z_row*, *cov*, *mu*)

Transform u to z_c.

Transformation from standard normal to multivariate normal space with given correlations following [1], page 77-102.

Step 1) Compute correlation matrix. Step 2) Introduce dependencies to standard normal sample. Step 3) Destandardize sample to normal space.

Parameters

- **z_row** (ndarray) – Row of uncorrelated standard normal deviates.
- **cov** (ndarray) – Covariance matrix of correlated normal deviates.
- **mu** (ndarray) – Expectation values of correlated normal deviates

Return type Tuple[ndarray, float]

Returns

- *x_norm_row* – Row of correlated normal deviates.
- *correlate_step* – Lower right corner element of the lower Cholesky matrix.

Notes

Importantly, the step in the numerator of the uncorrelated Elementary Effect is multiplied by *correlate_step*. Therefore, this factor has to multiply the step in the denominator as well to not violate the definition of the function derivation. This method is equivalent to the one in [2], page 199 which uses the Cholesky decomposition of the covariance matrix directly. This saves the scaling by SD and expectation. This method is simpler and slightly more precise than the one in [3], page 33, for normally distributed paramters. [1] explains how Rosenblatt and Nataf transformation are equal for normally distributed deviates.

References

[1] Lemaire, M. (2013). Structural reliability. John Wiley & Sons. [2] Gentle, J. E. (2006). Random number generation and Monte Carlo methods. Springer Science & Business Media. [3] Ge, Q. and M. Menendez (2017). Extending morris method for qualitative global sensitivity analysis of models with dependent inputs. Reliability Engineering & System Safety 100 (162), 28–39.

hypermodern_screening.transform_distributions.**transform_uniform_stnormal_uncorr**(*uniform_deviat*
nu-
meric_zero=0.0

Transorm u to z_u.

Converts sample from uniform distribution to standard normal space without regarding correlations.

Parameters

- **uniform_deviates** (ndarray) – Draws from Uniform[0,1].
- **numeric_zero** (float) – Used to substitute zeros and ones before applying *scipy.stats.norm* to not obtain *-Inf* and *Inf*.

Returns *uniform deviates* converted to standard normal space without correlations.

Return type stnormal_deviates

See also:

morris_trajectory()

Notes

This transformation is already applied as option in *morris_trajectory*. The reason is that *scipy.stats.norm* transforms the random draws from the unit cube non-linearily including the addition of the step. To obtain non-distorted screening measures, it is important to also account for this transformation of the step in the denominator to not violate the definition of the function derivation. The parameter *numeric_zero* can be highly influential. I prefer it to be relatively large to put more proportional, i.e. less weight on the extremes.

2.4 hypermodern_python.transform_reorder

Functions for reordering the sample rows following [1].

The intuition behind the reordering in general is the following: To compute the uncorrelated Elementary Effects, one moves the sampled elements that have been changed by *step* to the back of the row. For the correlated EE, one leaves the newly changed element in front, but moves the elements that were changed in rows above to the end. These compose the left parts of the numerator in the EE definition. One then subtracts the same row, except that the changed element is unchanged. The reason for these reorderings is that the correlation technique works hierarchically, like Dominoes. The element before is unaffected by the correlation of the elements thereafter. This

implies that the first element is unchanged, as for the correlated EE. Therefore, the step is involved in correlating the other elements without becoming changed itself. The opposite is true for the uncorrelated EE. The above procedure is dervied from the ordering in trajectory samples. It also works for the radial design. Other functions order the expectations and covariance matrix accordingly. They are also used to initialize the correlating loops in the two functions in *transform_ee.py* in the right order.

References

[1] Ge, Q. and M. Menendez (2017). Extending morris method for qualitative global sensitivityanalysis of models with dependent inputs. Reliability Engineering & System Safety 100 (162), 28–39.

hypermodern_screening.transform_reorder.**ee_corr_reorder_sample**(*sample*)

For each row i (non-pythonic), move the first i-1 elements to the back.

Parameters **sample** (ndarray) – sample.

Returns Reordered sample.

Return type sample_reordered

Notes

There is no *row_plus_one=False* option because this is equivalent with *uncorr_reorder_sample(sample, row_plus_one=True)*.

hypermodern_screening.transform_reorder.**ee_uncorr_reorder_sample**(*sample*, *row_plus_one=True*)

For each row i (non-pythonic), move the first i elements to the back.

Parameters

- **sample** (ndarray) – sample.
- **row_plus_one** (bool) – Add 1 to row index, i.e. start with second row.

Returns Reordered sample.

Return type sample_reordered

hypermodern_screening.transform_reorder.**reorder_cov**(*cov*)

Arrange covariance matrix according to the expectation vector.

(When the first element is moved to the end.)

Parameters **cov** (ndarray) – Covariance matrix of row.

Returns Reordered covariance matrix of row.

Return type cov_reordered

hypermodern_screening.transform_reorder.**reorder_mu**(*mu*)

Move the first element of the expectation vector to the end.

Parameters **mu** (ndarray) – Expectation values of row.

Returns Reordered expectation values of row.

Return type mu_reordered

hypermodern_screening.transform_reorder.**reverse_ee_corr_reorder_sample**(*sample_reordered*)

Reverse of function *corr_reorder_sample*.

Parameters **sample_reordered** (ndarray) – Reordered sample.

Returns Trjectory in original order.

Return type sample

hypermodern_screening.transform_reorder.**reverse_ee_uncorr_reorder_sample**(*sample_reordered*, *row_plus_one=True*)

Reverse of function *uncorr_reorder_sample*.

Parameters **sample_reordered** (*ndarray*) – Reordered sample.

Returns **sample** – Trjectory in original order.

Return type ndarray

hypermodern_screening.transform_reorder.**reverse_reorder_cov**(*cov_reordered*)

Reverse of function *reorder_cov*.

Parameters **cov_reordered** (*ndarray*) – Reordered covariance matrix.

Returns **cov** – Covarince matrix in original order.

Return type ndarray

hypermodern_screening.transform_reorder.**reverse_reorder_mu**(*mu_reordered*)

Reverse of function *reorder_mu*.

Parameters **mu_reordered** (ndarray) – Reordered expectation values of row.

Returns Expectation values of row in original order.

Return type mu

2.5 hypermodern_python.transform_ee

Compute the component for the redesigned EE expressions.

Functions to compute the arguments for the function evaluations in the numerator of the individual uncorrelated and correlated Elementary Effects following [1], page 33 and 34, and coefficients that scale the step. These functions can handle samples in both, trajectory and radial, designs.

References

[1] Ge, Q. and M. Menendez (2017). Extending morris method for qualitative global sensitivity analysis of models with dependent inputs. Reliability Engineering & System Safety 100 (162), 28–39.

hypermodern_screening.transform_ee.**trans_ee_corr**(*sample_list*, *cov*, *mu*, *radial=False*)

Transform list of samples to two lists of transformed samples.

(For the computation of the correlated Elementary Effects.)

Parameters

- **sample_list** (List) – Set of untransformed samples.
- **cov** (ndarray) – Covariance matrix.
- **mu** (ndarray) – Expectation value.
- **radial** (bool) – Sample is in trajectory or radial design.

Returns samples containing the rows that are the arguments for the LHS function evaluation for the correlated Elementary Effect.

Return type trans_piplusone_iminusone

Raises **AssertionError** – If the dimension of *mu*, *cov* and the elements in *sample_list* do not fit together.

Notes

For the trajectory design, the transformation for the rows on the RHS of the correlated Elementary Effects is equal to the one on the LHS of the uncorrelated Elementary Effects. Therefore, if *radial is False*, this transformation is skipped and left to *trans_ee_uncorr_samples*. To compute the EEs from radial samples, the arguments of the subtracted function are the first row of the sample. Yet, they must be reordered and transformed according to their order, too.

See also:

`trans_ee_uncorr_samples()`

hypermodern_screening.transform_ee.**trans_ee_uncorr**(*sample_list*, *cov*, *mu*, *radial=False*)

Transform list of samples to two lists of transformed samples.

(For the computation of the uncorrelated Elementary Effects.)

Parameters

- **sample_list** (List[ndarray]) – Set of untransformed samples.
- **cov** (*np.ndarray*) – Covariance matrix.
- **mu** (ndarray) – Expectation value.
- **radial** (bool) – Sample is in trajectory or radial design.

Return type Tuple[List[ndarray], List[ndarray], List[ndarray]]

Returns

- *trans_piplusone_i* – samples containing the rows that are the arguments for the LHS function evaluation for the uncorrelated Elementary Effect.
- *trans_pi_i* – samples containing the rows that are the arguments for the RHS function evaluation for the uncorrelated Elementary Effect.
- *coeff_step* – Factors in the denumerator of the uncorrelated Elementary Effect. Accounts for the decorrelation of the Step.

Raises **AssertionError** – If the dimension of *mu*, *cov* and the elements in *sample_list* do not fit together.

Notes

The rows in the two different transformed samples equal to T(p_{i+1}, i) and T(p_{i}, i). Understanding the transformations may require to write up the first transformation from p_i and p_{i+1} to T_1(p_{i}, i) and T_1(p_{i+1}, i). T_1 shifts the first i elements to the end for each row p_{i}. This function creates list of transformations of whole samples. The rows in the samples for T(p_{i}, i) that are to be subtracted from T(p_{i+1}, i), are still positioned one below compared to the samples for T(p_{i}, i). Therefore, importantly, one needs to compare each row in a sample from *trans_pi_i* with the respective row one below in *trans_piplusone_i*. To compute the EEs from radial samples, the arguments of the subtracted function are the first row of the sample. Yet, they must be reordered and transformed according to their order, too.

2.6 hypermodern_python.screening_measures

Compute Elementary Effects from transformed samples and derived measures.

Computes the screening measures for correlated inputs that I improved upon [1] by adjusting the step in the denumeroter to the transformed step in the nominator in order to not violate the definition of the function derivative.

References

[1] Ge, Q. and M. Menendez (2017). Extending morris method for qualitative global sensitivityanalysis of models with dependent inputs. Reliability Engineering & System Safety 100 (162), 28–39.

hypermodern_screening.screening_measures.**compute_measures**(*ee_i*, *sd_x=array([1])*, *sd_y=array([1])*, *sigma_norm=False*, *ub=False*)

Compute aggregate measures based on (individual) Elementary Effects.

ee_i (individual) Elementary Effects of input paramters (cols).

sd_x Parameters' SD.

sd_y QoI's SD.

sigma_norm Indicates wether to compute measures normalized by *sd_x / sd_y*.

ub Indicates wether to compute squared EEs and measures normalized by *var_x / var_y*.

Returns

contains:

ee_mean Mean Elementary Effect for each parameter.

ee_abs_mean Mean absolute correlated Elementary Effect for each parameter.

ee_sd SD of correlated Elementary Effects for each parameter.

Return type measures_list

Notes

ub follows http://www.andreasaltelli.eu/file/repository/DGSA_MATCOM_2009.pdf.

hypermodern_screening.screening_measures.**screening_measures**(*function*, *traj_list*, *step_list*, *cov*, *mu*, *radial=False*)

Compute screening measures for a set of paramters.

Parameters

- **function** (Callable) – Function or Model of which its parameters are subject to screening.
- **traj_list** (List[ndarray]) – List of transformed trajectories according to [1].
- **step_list** (List[ndarray]) – List of steps that each parameter takes in each trajectory.
- **cov** (ndarray) – Covariance matrix of the input parameters.
- **mu** (ndarray) – Expectation values of the input parameters.

- **radial** (bool) – Sample is in trajectory or radial design.

Return type Tuple[List[ndarray], List[ndarray]]

Returns

- *measures_list* –

  contains:

  ee_uncorr Mean uncorrelated Elementary Effect for each parameter.

  ee_corr Mean correlated Elementary Effect for each parameter.

  abs_ee_uncorr Mean absolute uncorrelated Elementary Effect for each parameter.

  abs_ee_corr Mean absolute correlated Elementary Effect for each parameter.

  sd_ee_uncorr SD of uncorrelated Elementary Effects for each parameter.

  sd_ee_corr SD of correlated Elementary Effects for each parameter.

- *obs_list* –

  contains:

  ee_uncorr_i Observations of uncorrelated Elementary Effects.

  ee_corr_i Observations of correlated Elementary Effects.

Notes

The samples can be in trajectory or in radial design and the deviates can be from an arbitrary (correlated) normal distribution or an uncorrelated Uniform[0,1] distribution. Unorrelated uniform paramters require different interpretion of *mu* as a scaling summand rather than the expectation value. It might be necessary to multiply the SDs by *(n_trajs/(n_trajs - 1))* for the precise formula. However, this leads to problems for the case of only one trajectory - which is used in *test_screening_measures_uncorrelated_g_function*.

Fill out.

CHAPTER

THREE

INSTALLATION

To install the Hypermodern Python project, run this command in your terminal:

```
$ pip install hypermodern-screening
```

PYTHON MODULE INDEX

h

INDEX

T