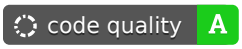
econsa Documentation

Release 0.01

Linda Maokomatanda

Jul 24, 2020

CONTENTS



I prefer true but imperfect knowledge, even if it leaves much undetermined and unpredictable, to a pretence of exact knowledge that is likely to be false.

– Friedrich von Hayek, Nobel Prize Lecture

`econsa` is an open-source package for economists that facilitates the sound analysis of computational economic models. It offers suitable methods for uncertainty propagation and global sensitivity analysis.

With `conda` available on your path, installing and testing `econsa` is as simple as typing

```
$ conda install -c opensourceeconomics econsa
$ python -c "import econsa; econsa.test()"
```

CHAPTER

ONE

SUPPORTED BY

1.1 Motivation

Computational economic models clearly specify an individual's objective and the institutional and informational constraints of their economic environment under which they operate. Fully-parameterized computational implementations of the economic model are estimated on observed data as to reproduce the observed individual decisions and experiences. Based on the results, researchers can quantify the importance of competing economic mechanisms in determining economic outcomes and forecast the effects of alternative policies before their implementation ([12]).

The uncertainties involved in such an analysis are ubiquitous. Any such model is subject to misspecification, its numerical implementation introduces approximation error, the data is subject to measurement error, and the estimated parameters remain partly uncertain.

A proper accounting of the uncertainty is a prerequisite for the use of computational models in most disciplines ([1][10]) and has long been recognized in economics as well ([5][3][8]). However, in practice economists analyze the implications of the estimated model, economists display incredible certitude ([9]) as all uncertainty is disregarded. As a result, flawed findings are accepted as truth and contradictory results are competing. Both have the potential to undermine the public trust in research in the long run.

Any computational economic model $\boldsymbol{M}$ provides a mapping between its input parameters $\boldsymbol{x}$ and the quantities of interest y.

$$\boldsymbol{M} : \boldsymbol{x} \in \mathcal{D}_{\boldsymbol{X} \mapsto y = \boldsymbol{M}(\boldsymbol{x})}$$

We follow [2] and use the **Economic Order Quantity (EOQ)** model ([6]) as a running example throughout our documentation. We thus start by explaining its basic setup first and then discuss uncertainty propagation and sensitivity analysis.

1.1.1 EOQ model

The **EOQ** inventory management model provides a stylized representation of the decision problem faced by a firm that needs to determine the order quantity of a product that minimizes the unit cost per piece. The unit cost T depends on the price of the product C, the size of the order X as each comes with a fixed cost S, and an annual capital cost R expressed as a percentage of the value of the inventory. Core simplifications of the model include a constant monthly demand M over the year and the delivery of each order in full when inventory reaches zero.

We can then derive the unit cost as follows:

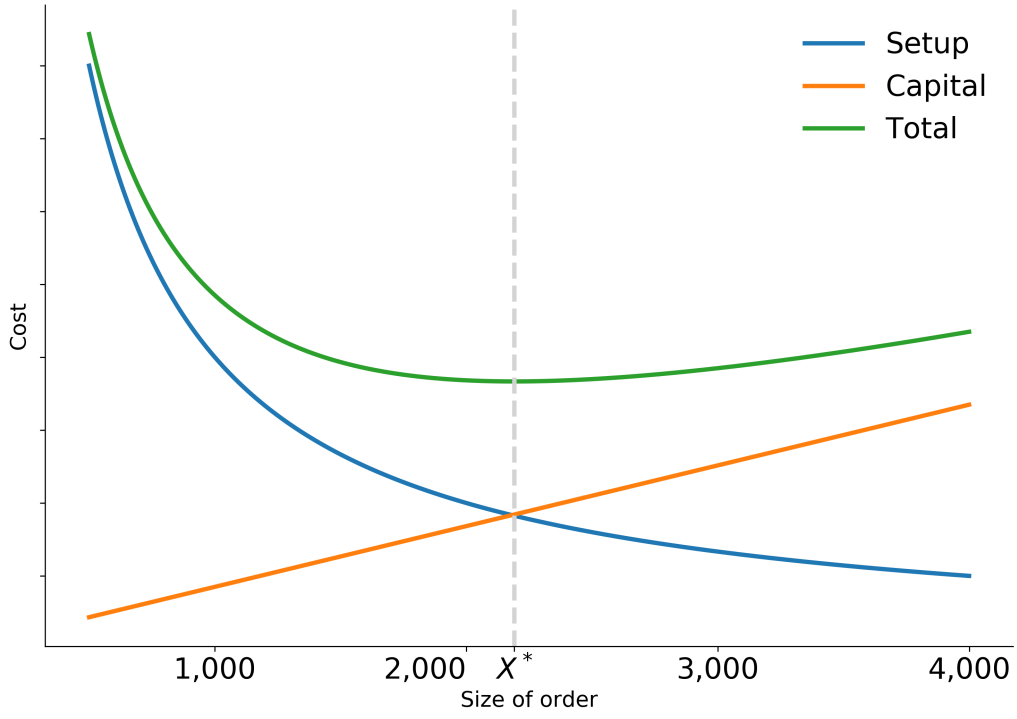
$$T = \underbrace{\frac{1}{12 \times M} \times R \times \frac{C \times X + S}{2}}_{\text{Part I}} + \underbrace{\frac{S}{X} + C}_{\text{Part II}}.$$

The first part of the equation denotes the capital cost of one unit in storage. It is computed based on the ratio of the value of the average stock and the total number of ordered units during the year. The second part captures each unit's cost as part of an order of size X.

The economic order quantity X^* is determined as:

$$X^* = \sqrt{\frac{24 \times M \times S}{R \times C}}.$$

The figure below reproduces the fundamental economic trade-offs of the model for a fixed parameterization of M, C, S, and R. An increase in the size of order X results in a decrease in the setup cost per unit, but at the same time, capital cost increases as the stock of inventory increase.



Going forward, we treat the annual interest and depreciation rate R as an exogenous parameter and set it to 10%. We can map the example to our more general notation by denoting the optimal order quantity as y and collecting the three

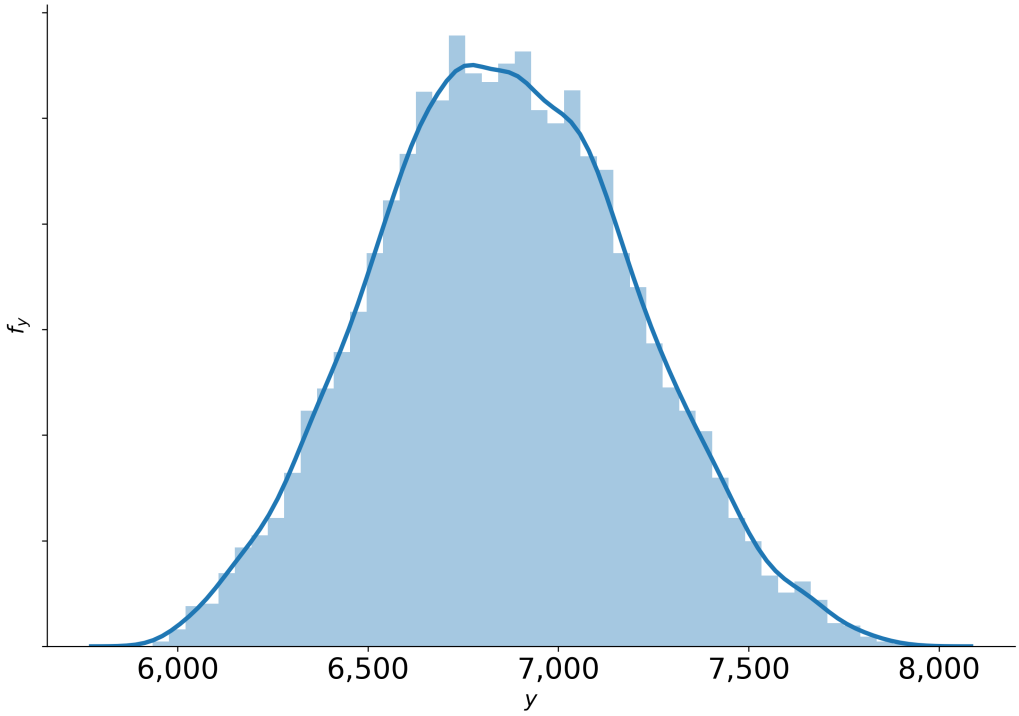
remaining input parameters in $\mathbf{x}$ as follows:

$$\mathbf{x} = (x_1, x_2, x_3)^T = (M, C, S)^T.$$

1.1.2 Uncertainty propagation

We start from a probabilistic model for the input parameters that is informed by, for example, expert knowledge or the outcome of a calibration. We treat the model parameters $\mathbf{X}$ as a simple random vector with a joint probability density function $f_{\mathbf{X}}$. We are not particularly interested in the uncertainty of each individual parameter of the model. Instead we seek to learn about the induced distribution of the model output Y as the uncertainty about the model parameters $\mathbf{X}$ propagates through the computational model $\mathbf{M}$. We want to study the statistical properties of Y.

We now return to the example of the **EOQ** model. We specify a uniform distribution centered around $\mathbf{x^0} = (M, C, S) = (1230, 0.0135, 2.15)$ and spread the support 10% above and below the center. We solve for the optimal economic order quantity Y for $1,000$ random input parameters and end up with the distribution f_Y below.



1.1.3 Sensitivity analysis

sensitivity analysis, hierarchization of input parameter with respect to the resulting uncertainty.

Quantitative sensitivity analysis

Elementary effects

Qualitative sensitivity analysis

Sobol indices

This part will be written by timmens as part of his thesis.

Shapely values

This part will be written by lindamaok899 as part of her thesis.

1.2 Methods

Here we explain and document in detail, the methods we implement in the `econsa` package to perform sensitivity analysis and uncertainty quantification. An insight into how the calculations are performed is not a prerequisite for using `econsa`, in most cases, the default settings works fine. Global Sensitivity Analysis can be divided into two categories: quali- and quantitative methods. `econsa` implements both methods as a comprehensive to ensure flexibility depending on your model requirements, features and specifications.

1.2.1 Qualitative sensitivity analysis

`econsa` applies the methods in [4] to calculate morris indices for models with dependent parameters. The Elementary Effects (EE), also known as the Morris method, is a qualitative way to screen inputs and helps to determine the set of influential and non-influential inputs. Shapely values on the other hand, ...

Constributor: Janos Gabler (janosg)

Calculate morris indices for models with dependent parameters.

We convert frequently between iid uniform, iid standard normal and multivariate normal variables. To not get confused, we use the following naming conventions:

-u refers to to uniform variables -z refers to standard normal variables -x refers to multivariate normal variables.

`econsa.morris.`**`elementary_effects`**(*func*, *params*, *cov*, *n_draws*, *sampling_scheme='sobol'*, *n_cores=1*)

Calculate Morris Indices of a model described by func.

The distribution of the parameters is assumed to be multivariate normal, with mean `params["value"]` and covariance matrix `cov`.

The algorithm is based on Ge and Menendez, 2017, (GM17): Extending Morris method for qualitative global sensitivity analysis of models with dependent inputs.

Parameters

- **`func`** (*`function`*) – Function that maps parameters into a quantity of interest.
- **`params`** (*`pd.DataFrame`*) – DataFrame with arbitrary index. There must be a column called value that contains the mean of the parameter distribution.

- **cov** (*pd.DataFrame*) – Both the index and the columns are the same as the index of params. The covariance matrix of the parameter distribution.
- **n_draws** (*int*) – Number of draws
- **sampling_scheme** (*str*) – One of ["sobol", "random"]. Default: "sobol"

Returns

- **mu_ind** (*float*) – Absolute mean of independent part of elementary effects
- **sigma_ind** (*float*) – Standard deviation of independent part of elementary effects

1.2.2 Quantitative sensitivity analysis

`econsa` provides several algorithms for quantitative sensitivity analysis.

Sobol indices

We implement the methods outlined in [7].

Constributor: Tim Mensinger (timmens)

Shapley values

We implement the methods outlined in [11].

Constributor: Linda Maokomatanda (lindamaok899)

1.2.3 Sampling methods

Capabilities for sampling of random input parameters.

This module contains functions to sample random input parameters.

`econsa.sampling.`**cond_mvn**(*mean*, *cov*, *dependent_ind*, *given_ind=None*, *given_value=None*, *check_cov=True*)

Conditional mean and variance function.

This function provides the conditional mean and variance-covariance matrix of [Y given X], where $Z = (X, Y)$ is the fully-joint multivariate normal distribution with mean equal to `mean` and covariance matrix `cov`.

This is a translation of the main function of R package condMVNorm.

Parameters

- **mean** (*array_like*) – The mean vector of the multivariate normal distribution.
- **cov** (*array_like*) – Symmetric and positive-definite covariance matrix of the multivariate normal distribution.
- **dependent_ind** (*int or array_like*) – The indices of dependent variables.
- **given_ind** (*array_like, optional*) – The indices of independent variables (default value is *None*). If not specified return unconditional values.
- **given_value** (*array_like, optional*) – The conditioning values (default value is *None*). Should be the same length as `given_ind`, otherwise throw an error.

- **check_cov** (*bool, optional*) – Check that `cov` is symmetric, and all eigenvalue is positive (default value is *True*).

Returns

- **cond_mean** (*array_like*) – The conditional mean of dependent variables.
- **cond_cov** (*arrray_like*) – The conditional covariance matrix of dependent variables.

Examples

```
>>> mean = np.array([1, 1, 1])
>>> cov = np.array([[4.0677098, -0.9620331, 0.9897267],
...                 [-0.9620331, 2.2775449, 0.7475968],
...                 [0.9897267, 0.7475968, 0.7336631]])
>>> dependent_ind = [0, ]
>>> given_ind = [1, 2]
>>> given_value = [1, -1]
>>> cond_mean, cond_cov = cond_mvn(mean, cov, dependent_ind, given_ind, given_
↪value)
>>> np.testing.assert_almost_equal(cond_mean, -4.347531, decimal=6)
>>> np.testing.assert_almost_equal(cond_cov, 0.170718, decimal=6)
```

Conditional sampling from Gaussian copula.

This module contains functions to sample random input parameters from a Gaussian copula.

`econsa.copula.`**cond_gaussian_copula**(*cov*, *dependent_ind*, *given_ind*, *given_value_u*, *size=1*)

Conditional sampling from Gaussian copula.

This function provides the probability distribution of conditional sample drawn from a Gaussian copula, given covariance matrix and a uniform random vector.

Parameters

- **cov** (*array_like*) – Covariance matrix of the desired sample.
- **dependent_ind** (*int or array_like*) – The indices of dependent variables.
- **given_ind** (*array_like*) – The indices of independent variables.
- **given_value_u** (*array_like*) – The given random vector (u) that is uniformly distributed between 0 and 1.
- **size** (*int*) – Number of draws from the conditional distribution. (default value is *1*)

Returns **cond_quan** – The conditional sample ($G(u)$) that is between 0 and 1, and has the same length as `dependent_ind`.

Return type numpy.ndarray

Examples

```
>>> np.random.seed(123)
>>> cov = np.array([[ 3.290887,  0.465004, -3.411841],
...                 [ 0.465004,  3.962172, -0.574745],
...                 [-3.411841, -0.574745,  4.063252]])
>>> dependent_ind = 2
>>> given_ind = [0, 1]
>>> given_value_u = [0.0596779, 0.39804426]
>>> condi_value_u = cond_gaussian_copula(cov, dependent_ind, given_ind, given_
↪value_u)
>>> np.testing.assert_almost_equal(condi_value_u[0], 0.856504, decimal=6)
```

1.3 Tutorials

We provide several tutorials that showcase the use case for `econsa`.

1.3.1 The EOQ model and uncertainty propagation

We show how to conduct uncertainty propagation for the **EOQ** model. We can simply import the core function from `temfpy`.

[139]:
```
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
import chaospy as cp

from temfpy.uncertainty_quantification import eoq_model
```

Setup

We specify a uniform distribution centered around $\mathbf{x^0} = (M, C, S) = (1230, 0.0135, 2.15)$ and spread the support 10% above and below the center.

[140]:
```
marginals = list()
for center in [1230, 0.0135, 2.15]:
    lower, upper = 0.9 * center, 1.1 * center
    marginals.append(cp.Uniform(lower, upper))
```

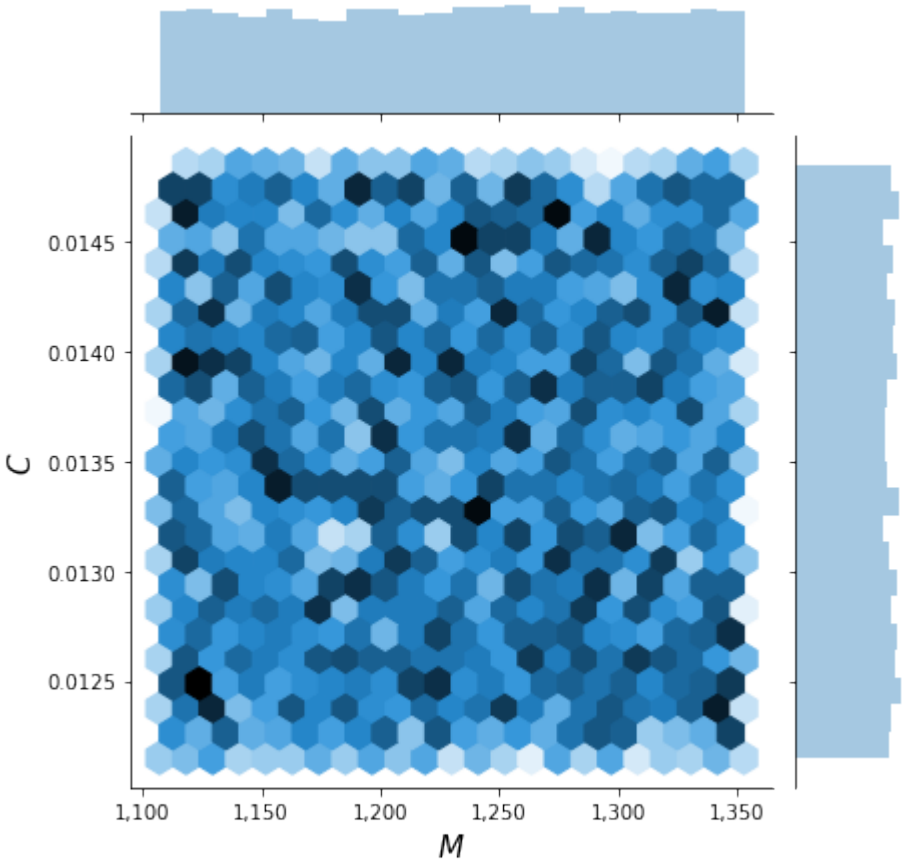
Independent parameters

We now construct a joint distribution for the the independent input parameters and draw a sample of $1,000$ random samples.

[147]:
```
distribution = cp.J(*marginals)
sample = distribution.sample(10000, rule="random")
```

The briefly inspect the joint distribution of M and C.

[149]:
```
plot_joint(sample)
```

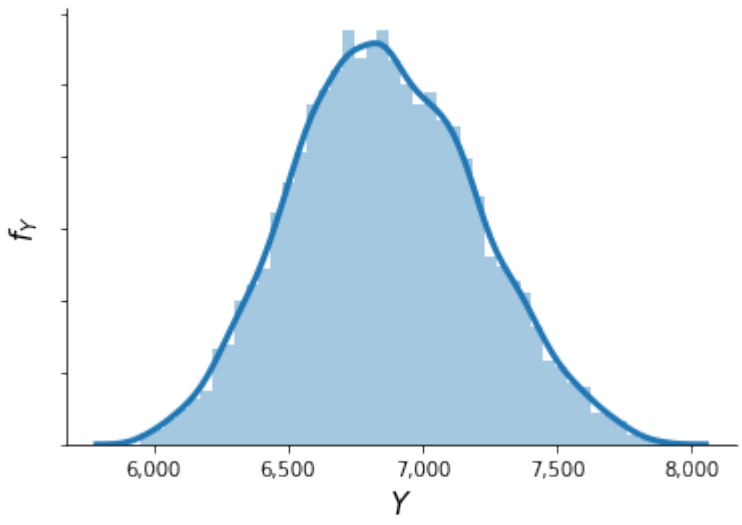


We are now ready to compute the optimal economic order quantity for each draw.

[150]:
```
y = eoq_model(sample)
```

This results in the following distribution f_Y.

[152]:
```
plot_quantity(y)
```



Depdendent paramters

We now consider dependent parameters and construct their joint distribution using a Gaussian copula.

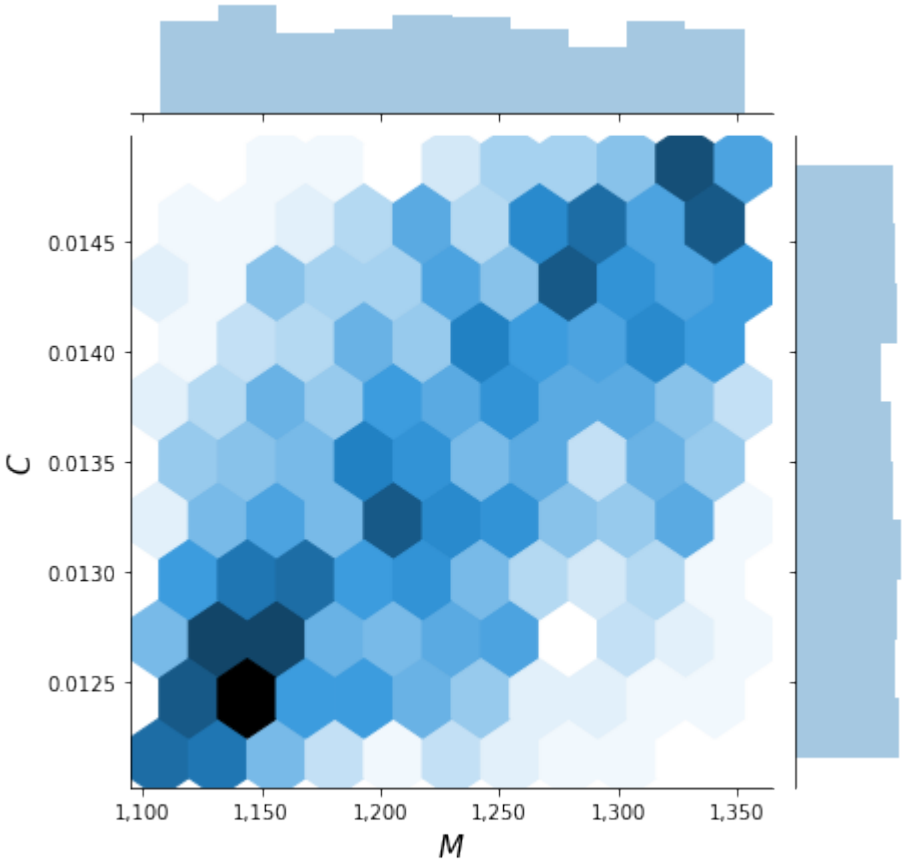
[153]:
```
corr = [[1.0, 0.6, 0.2], [0.6, 1.0, 0.0], [0.2, 0.0, 1.0]]
copula = cp.Nataf(distribution, corr)
```

We are ready to sample from the distribution.

[154]:
```
sample = copula.sample(1000, rule="random")
```

Again, we briefly inspect the joint distribution which now clearly shows a dependence pattern.

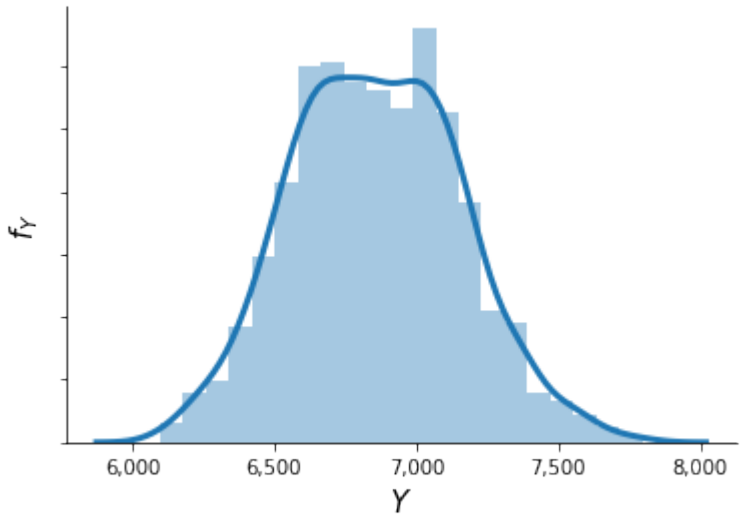
[155]:
```
plot_joint(sample)
```



[156]:
```
y = eoq_model(sample)
```

This now results in a distribution of f_Y where the peak is flattened out.

[160]:
```
plot_quantity(y)
```



1.3.2 Qualitative sensitivity analysis

We showcase the use of `econsa` for qualitative sensitivity analysis.

[7]:
```
# from temfpy.uncertainty_quantification import simple_linear_function,
from econsa.morris import elementary_effects  # noqa: F401
import numpy as np

# This is an example where we extracted the sampling from the
# function to compute the elementary effects.
input_parameters = np.random.normal(size=(100, 4))
n_draws, cov = 100, np.identity(3)

# TODO: This requires some more work to set up.
# rslt = elementary_effects(simple_linear_function, input_parameters, cov, n_draws)
```

1.4 Projects

econsa is under active development in support of several research and educational projects.

1.4.1 Research

- Eisenhauer, P., Gabler, J., Janys, L., & Mensinger, T. (2020). Uncertainty quantification for structural econometric models. Unpublished Working Paper .

1.4.2 Thesis

- Maokomatanda, L. (2020). ...
- Mensinger, T. (2020). ...
- Wan, L. (2020). ...

1.5 Published versions

1.6 Acknowledgements

econsa is developed and maintained as part of the OpenSourceEconomics initiative.

Project Manager

- Philipp Eisenhauer (peisenha)

Developers

- Linda Maokomatanda (lindamaok899)
- Janos Gabler (janosg)
- Tim Mensinger (timmens)
- Leiqiong Wan (loikein)

1.7 Related work

We are drawing on related work throughout.

1.7.1 Software

- Feinberg, J., and Langtangen, H. P. (2015). *Chaospy: an open source tool for designing methods of uncertainty quantification*. *Journal of Computational Science, 11*, 46-57.
- Herman, J., and Usher, W. (2017). *SALib: an open-source Python library for sensitivity analysis*. *Journal of Open Source Software, 2* (9).
- Tennoe S., Halnes G., and Einevoll G.T. (2018). *Uncertainpy: a Python toolbox for uncertainty quantification and sensitivity analysis in computational neuroscience*. *Frontiers in Neuroinformatics, 12*, 49.

1.7.2 Books

- Ghanem, R., Higdon, D., and Owhadi, H. (2017). *Handbook of uncertainty quantification*. Cham, Switzerland: Springer International Publishing.
- Saltelli et al. (2008). *Global Sensitivity Analysis. The Primer*. Chichester, UK: John Wiley & Sons Ltd.
- Saltelli, A., Tarantola, S., Campolongo, F., and Ratto, M. (2004). *Sensitivity analysis in practice. A guide to assessing scientific models*. Chichester, UK: John Wiley & Sons Ltd.
- Smith, R.C. (2014). *Uncertainty quantification: theory, implementation, and applications*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

- Sullivan, T.J. (2015). *Introduction to uncertainty quantification*. Cham, Switzerland: Springer International Publishing.

1.7.3 Popular science

- King, M., and Kay, J. (2020). *Radical uncertainty: decision-making for an unknowable future*. London, UK: The Bridge Street Press.

1.8 Bibliography

BIBLIOGRAPHY

[1] Marvin L. Adams, editor. *Assessing the reliability of complex models: mathematical and statistical foundations of verification, validation, and uncertainty quantification*. National Academies Press, Washington, D.C., 2012.

[2] Emanuele Borgonovo and Elmar Plischke. Sensitivity analysis: a review of recent advances. *European Journal of Operational Research*, 248(3):869–887, 2016.

[3] Fabio Canova. Statistical inference in calibrated models. *Journal of Applied Econometrics*, 9(1):123–144, 1994.

[4] Qiao Ge and Monica Menendez. An efficient sensitivity analysis approach for computationally expensive microscopic traffic simulation models. *International Journal of Transportation*, 2(2):49–64, 2014.

[5] Lars Peter Hansen and James J. Heckman. The empirical foundations of calibration. *Journal of economic perspectives*, 10(1):87–104, 1996.

[6] Ford W. Harris. How many parts to make at once. *Operations Research*, 38(6):947–950, 1990.

[7] Sergei Kucherenko, Stefano Tarantola, and Paola Annoni. Estimation of global sensitivity indices for models with dependent variables. *Computer Physics Communications*, 183(4):937–946, 2012.

[8] Finn E. Kydland. On the econometrics of world business cycles. *European Economic Review*, 36(2-3):476–482, 1992.

[9] Charles F. Manski. Communicating uncertainty in policy analysis. *Proceedings of the National Academy of Sciences*, 116(16):7634–7641, 2019.

[10] William L. Oberkampf and Christopher J. Roy. *Verification and validation in scientific computing*. Cambridge University Press, Cambridge, UK, 2010.

[11] Art B. Owen. Sobol' indices and shapley value. *SIAM/ASA Journal on Uncertainty Quantification*, 2(1):245–251, 2014.

[12] Kenneth I. Wolpin. *The limits to inference without theory*. MIT University Press, Cambridge, MA, 2013.

PYTHON MODULE INDEX

e

INDEX