CUBmods

Release 0.0.1

Massimo Pierini

Aug 16, 2024

CONTENTS:

CHAPTER

ONE

INTRODUCTION

The class of CUB (Combination of Uniform and Binomial) models, proposed by Professor Domenico Piccolo in 2003 (Piccolo and others, 2003) within the cotext of rating and preference data analysis, hypothesizes that the ordinal responses provided by the raters are not the simple result of a reasoned choice, but rather the complex combination of a multitude of factors, both internal and external (Piccolo and Simone, 2019).

Simplifying, two main components can be distinguished: *feeling* and *uncertainty*.

The primary component of feeling is due to sufficient awareness and understanding of the topic based on knowledge and experience (Piccolo and Simone, 2019). The secondary component of uncertainty is instead generated by an *intrinsic fuzziness*, due to various circumstances: limited knowledge, lack of interest, timing of the survey, method of administration, boredom, etc. (Piccolo and Simone, 2019).

The simplest way to consider these two aspects is a distribution resulting from a mixture of a shifted Binomial component for the first and Uniform Discrete for the second (Piccolo and Simone, 2019) which takes the form of the CUB family models, subsequently extended to consider further factors such as the overdispersion of Binomial component, the effect of shelter choice, etc. (Piccolo and Simone, 2019).

This package is the first implementation of CUB class models in Python and is mainly based upon the work of Domenico Piccolo and Rosaria Simone, and the `CUB` package in R (Iannario *et al.*, 2022).

It has been implemented by Massimo Pierini as a Bachelor's thesis (Pierini, 2024).

1.1 Motivation

Currently the class of CUB models has been implemented in statistical and econometric programming languages such as R (Iannario *et al.*, 2022), Stata (Cerulli *et al.*, 2021), Gretl (Piccolo and Simone, 2019, Simone *et al.*, 2019) and GAUSS (Piccolo and others, 2006). However, given the recent increase in the development of the Python programming language also in the statistical field (Pittard and Li, 2020), their implementation in this environment could be useful to the scientific community.

1.2 References

CHAPTER

TWO

MANUAL

The package `cubmods` can be used to apply inferential methods to an observed sample in order to estimate the parameters and the covariance matrix of a model within the CUB class. Also, for each family, random samples can be drawn from a specified model.

Currently, six families have been defined and implemented:

- CUB (Combination of Uniform and Binomial)
- CUBSH (CUB + a SHelter choice)
- CUSH (Combination of Uniform and a SHelter choice)
- CUSH2 (Combination of Uniform and 2 SHelter choices)
- CUBE (Combination of Uniform and BEta-binomial)
- IHG (Inverse HyperGeometric)

For each family, a model can be defined with or without covariates for one or more parameters.

Details about each family and examples are provided in the following chapters.

Even if each family has got its own *Maximum Likelihood Estimation* function `mle()` that could be called directly, for example `cub.mle()`, the function `gem.estimate()` provides a simplified and generalised procedure for MLE.

Similarly, even if each family has got its own *Random Sample Drawing* function `draw()` that could be called directly, for example `cub.draw()`, the function `gem.draw()` provides a simplified and generalised procedure to draw a random sample.

In this manual `gem` functions will be used for the examples.

The last chapter, shows the basic usage for the tool `multicub`.

2.1 GeM usage

GeM (Generalized Mixture) is the main module of `cubmods` package, which provides simplified and generalized functions to both estimate a model from an observed sample and draw a random sample from a specified model.

The function `gem.estimate()` is the main function for the estimation and validation of a model from an observed sample, calling for the corresponding `.mle()` function of the specified family, with or without covariates.

The function `gem.draw()` is the main function for drawing a random sample from a specified model, calling for the corresponding `.draw()` function of the corresponding family, with or without covariates.

2.1.1 The *formula* syntax

Both functions need a `formula` that is a **string** specifying the name of the ordinal variable (before the tilde ~ symbol) and of the covariates of the components (after the tilde symbol ~). Covariates for each component are separated by the pipeline symbol `|`. The *zero* symbol `0` indicates no covariates for a certain component. The *one* symbol `1` indicates that we want to estimate the parameter of the constant term only. If more covariates explain a single component, the symbol + concatenates the names. Qualitative variables names, must be placed between brackets `()` leaded by a `C`, for example `C(varname)`.

Warning: No columns in the DataFrame should be named `constant`, `1` or `0`. In the column names, only letters, numbers, and underscores _ are allowed. Spaces **SHOULD NOT BE** used in the column names, but replaced with _.

For example, let's suppose we have a DataFrame where `response` is the ordinal variable, `age` and `sex` are a quantitative and a qualitative variable to explain the *feeling* component only in a `cub` family model. The formula will be `formula = "response ~ 0 | age + C(sex)"`.

Notice that spaces are allowed between symbols and variable names in the formula but they aren't needed: a formula `"ord ~ X | Y1 + Y2 | Z"` is the same as `"ord~X|Y1+Y2|Z"`.

Warning: The number of fields separated by the pipeline `|` in a formula **MUST BE** equal to the number of parameters specifying the model family. Therefore: two for `cub` and `cush2`, three for `cube` and `cub` with shelter effect, one for `cush` and `ihg`.

2.1.2 Arguments of `estimate` and `draw`

Within the function `estimate` the number of ordinal categories `m` is internally retrieved if not specified (taking the maximum observed category) but it is advisable to pass it as an argument to the call if some category has zero frequency. Within the function `draw` instead, the number of ordinal categories `m` must always be specified.

A `pandas` DataFrame must always be passed to the function `estimate`, with the *kwarg* `df`. It should contain, at least, a column of the observed sample and the columns of the covariates (if any). If no `df` is passed to the function `draw` for a model without covariates instead, an empty DataFrame will be created.

The number `n` of ordinal responses to be drawn should always be specified in the function `draw` for models without covariates. For model with covariates instead, `n` is not effective because the number of drawn ordinal responsed will be equal to the passed DataFrame rows.

A `seed` could be specified for the function `draw` to ensure reproducibility. Notice that, for models with covariates, `seed` cannot be `0` (in case, it will be automatically set to `1`).

If no `model` is declared, the function takes `"cub"` as default. Currently implemented models are: `"cub"` (default), `"cush"`, `"cube"`, and `"ihg"`. CUB models with shelter effect are automatically implemented using `model="cub"` and specifying a shelter choice with the *kwarg* `sh`. CUSH2 models are automatically implemented using `model="cush"` and passing a list of two categories to the *kwarg* `sh` instead of an integer.

To `draw` must be passed the parameters' values with the *kwargs* of the corresponding family: for example, `pi` and `xi` for CUB models without covariates, `beta` and `gamma` for CUB models with covariates for both feeling and uncertainty, etc. See the `.draw()` function reference of the corresponding family module for details.

If `model="cub"` (or nothing), then a CUB mixture model is fitted to the data to explain uncertainty, feeling (`ordinal~Y|W`) and possible shelter effect by further passing the extra argument `sh` for the corresponding category. Subjects' covariates can be included by specifying covariates matrices in the formula as `ordinal~Y|W|X`, to explain uncertainty (Y), feeling (W) or shelter (X). Notice that covariates for shelter effect can be included only if specified

for both feeling and uncertainty (GeCUB models). Nevertheless, the symbol `1` could be used to specify a different combination of components with covariates. For example, if we want to specify a CUB model with covariate `cov` for uncertainty only, we could pass the formula `ordinal ~ cov | 1 | 1`: in this case, for feeling and shelter effect, the constant terms only (γ_0 and ω_0) will be estimated and the values of the estimated ξ and δ could be computed as $\hat{\xi} = \text{expit}(\hat{\gamma}_0)$ and $\hat{\delta} = \text{expit}(\hat{\omega}_0)$.

If `family="cube"`, then a CUBE mixture model (Combination of Uniform and Beta-Binomial) is fitted to the data to explain uncertainty, feeling and overdispersion. Subjects' covariates can be also included to explain the feeling component or all the three components by specifying covariates matrices in the Formula as `ordinal~Y|W|Z` to explain uncertainty (Y), feeling (W) or overdispersion (Z). For different combinations of components with covariates, the symbol 1 can be used. Notice that $\hat{\phi} = e^{\hat{\alpha}_0}$.

If `family="ihg"`, then an IHG model is fitted to the data. IHG models (Inverse Hypergeometric) are nested into CUBE models. The parameter θ gives the probability of observing the first category and is therefore a direct measure of preference, attraction, pleasantness toward the investigated item. This is the reason why θ is customarily referred to as the preference parameter of the IHG model. Covariates for the preference parameter θ have to be specified in matrix form in the Formula as `ordinal~V`.

If `family="cush"`, then a CUSH model is fitted to the data (Combination of Uniform and SHelter effect). If a category corresponding to the inflation should be passed via argument `sh` a CUSH model is called and covariates for the shelter parameter δ are specified in matrix form Formula as `ordinal~X`. If two category corresponding to the inflation should be passed via argument `sh` (as a *list* or *array*) a CUSH2 model is called and covariates for the shelters' parameters (δ_1, δ_2) are specified in matrix form Formula as `ordinal~X1|X2`. Notice that, to specify covariates for a single shelter choice in a CUSH2 model, the formula should be `ordinal~X1|0` and not `ordinal~0|X2`.

Extra arguments include the maximum number of iterations `maxiter` for the optimization algorithm, the required error tolerance `tol`, and a dictionary of parameters of a known model `gen_pars` to be compared with the estimates.

2.1.3 Methods of `estimate` and `draw`

For both functions, the methods `.summary()` and `.plot()` are always available calling the main functions to print a summary and plot the results, respectively. For `.plot()` arguments and options, see the `CUBsample` Class (for object returned by `draw`) and the extended `CUBres` Classes of the corresponding family (for objects returned by `estimate`).

Calling `.as_dataframe()` will return a DataFrame of parameters' names and values for objects of the Class `CUBsample` returned by `draw`. For objects of the Base Class `CUBres` returned by `estimate` instead, will return a DataFrame with parameters' component, name, estimated value, standard error, Wald test statistics and p-value.

Calling the method `.save(fname)` the object can be saved on a file called `fname.cub.sample` (for `draw`) or `fname.cub.fit` (for `estimate`).

Saved objects can then be loaded using the function `general.load(fname)`.

2.1.4 Attributes of `estimate` and `draw`

For both objects returned by `estimate` and `draw`, the attributes `.formula` and `.df` are always available. The function `draw` will return the original DataFrame (if provided) with an extra column of the drawn ordinal response called as specified in the formula.

Many other attributes can be called from objects of the Base Class `CUBres` returned by `estimate`, such as the computed loglikelihood, the AIC and BIC, ectcetera. For details, see the Base Class `CUBres` reference guide.

2.2 CUB family

Basic family of the class CUB. See the references for details.

2.2.1 References

1. Angela D'Elia and Domenico Piccolo. A mixture model for preferences data analysis. *Computational Statistics & Data Analysis*, 49(3):917–934, 2005.
2. Maria Iannario, Domenico Piccolo, and others. Inference for cub models: a program in r. *Statistica & Applicazioni*, 12(2):177–204, 2014.
3. Domenico Piccolo and others. On the moments of a mixture of uniform and shifted binomial random variables. *Quaderni di Statistica*, 5(1):85–104, 2003.
4. Domenico Piccolo and others. Observed information matrix for mub models. *Quaderni di Statistica*, 8(1):33–78, 2006.
5. Domenico Piccolo and Rosaria Simone. The class of cub models: statistical foundations, inferential issues and empirical evidence. *Statistical Methods & Applications*, 28:389–435, 2019.

2.2.2 Without covariates

A model of the CUB family for responses with m ordinal categories, without covariates is specified as

$$\Pr(R = r|\boldsymbol{\theta}) = \pi \binom{m-1}{r-1} (1-\xi)^{r-1} \xi^{m-r} + \frac{1-\pi}{m}$$

where π and ξ are the parameters for respectively the *uncertainty* and the *feeling* components.

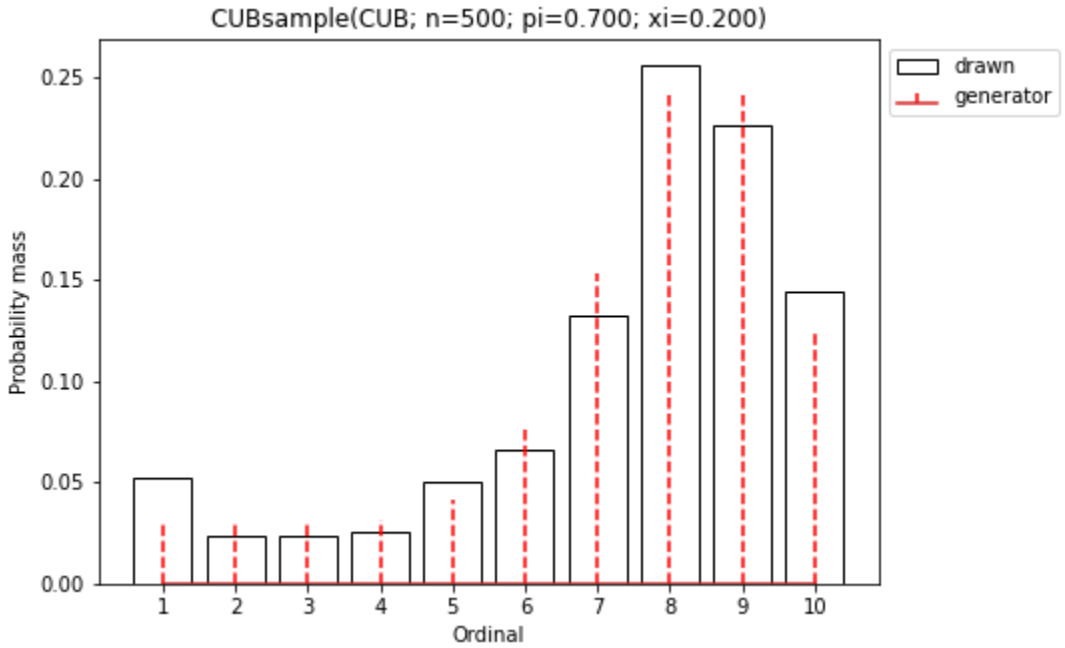
Note that $(1-\pi)$ is the weight of the Uncertainty component and $(1-\xi)$ is the Feeling component for usual *positive wording*.

In the following example, a sample will be drawn from a CUB model of $n = 500$ observations of an ordinal variable with $m = 10$ ordinal categories and parameters $(\pi = .7, \xi = .2)$. A `seed=1` will be set to ensure reproducibility.

Listing 1: Script

```
# import libraries
import matplotlib.pyplot as plt
from cubmods.gem import draw

# draw a sample
drawn = draw(
    formula="ord ~ 0 | 0",
    m=10, pi=.7, xi=.2,
    n=500, seed=1)
# print the summary of the drawn sample
print(drawn.summary())
# show the plot of the drawn sample
drawn.plot()
plt.show()
```

```
==========================================================================
=====>>> CUB model <<<====== Drawn random sample
==========================================================================
m=10  Sample size=500  seed=1
formula: ord~0|0
--------------------------------------------------------------------------
pi=0.700
xi=0.200
==========================================================================
Sample metrics
Mean     = 7.368000
Variance = 5.687952
Std.Dev. = 2.384943
--------------------------------------------------------------------------
Dissimilarity = 0.0650938
==========================================================================
```



Notice that, since the default value of the kwarg `model` is `"cub"` we do not need to specify it.

Calling `drawn.as_dataframe()` will return a DataFrame with the parameters

```
  parameter  value
0        pi    0.7
1        xi    0.2
```

Using the previously drawn sample, in the next example the parameters $(\hat{\pi}, \hat{\xi})$ will be estimated.

Note that in the function `gem.estimate`:

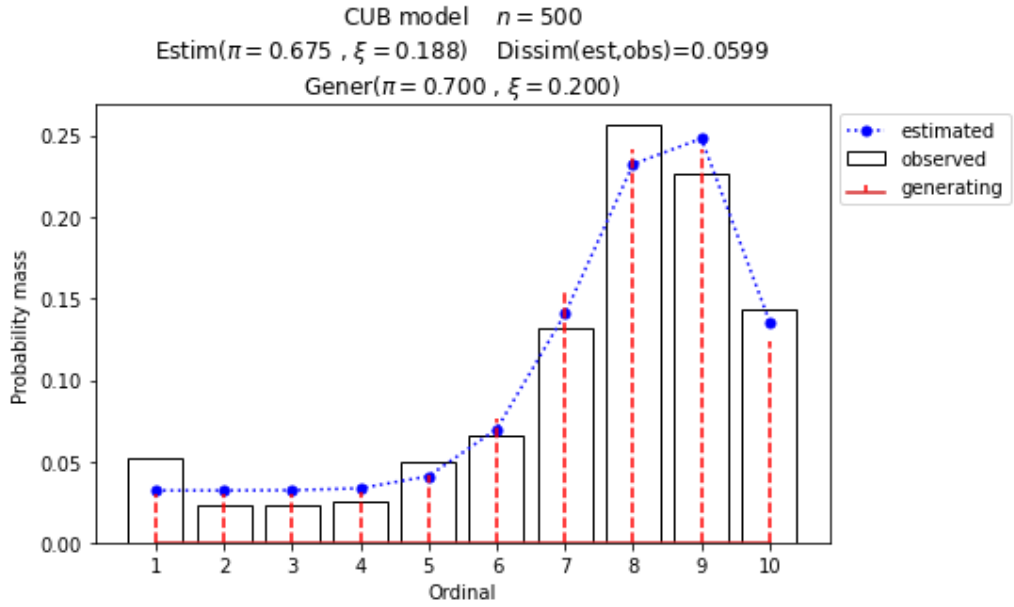
- `df` needs to be a `pandas` DataFrame; the attribute `drawn.df` will return a DataFrame with `ord` as column name of the drawn ordinal response (as previuosly speficied in the formula)
- `formula` needs the ordinal variable name (`ord` in this case) and the covariates for each component (none in this case, so `"0|0"`)

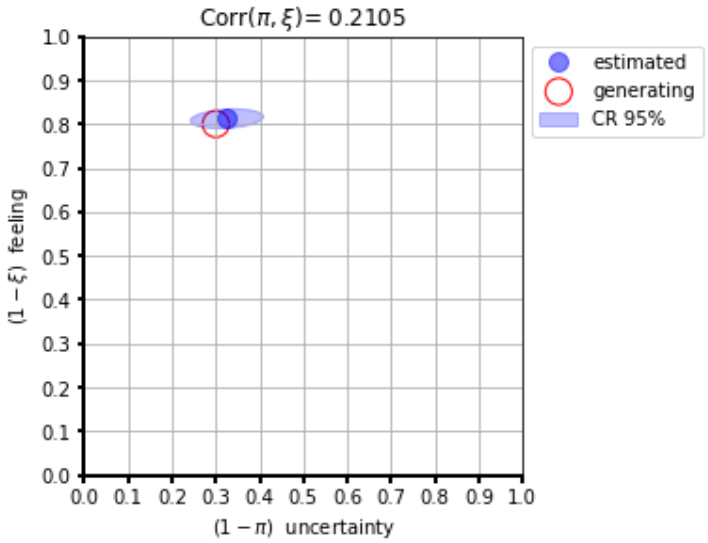
- if `m` is not provided, the maximum observed ordinal value will be assumed and a warning will be raised
- with `gen_pars` dictionary, the parameters of a known model (if any) can be specified; in this case, we'll specify the known parameters used to draw the sample

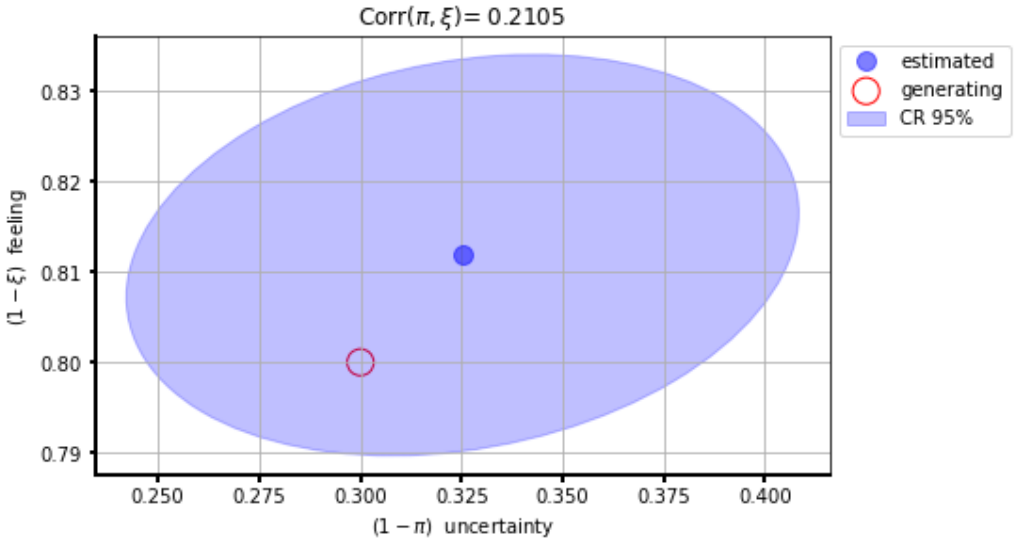
Listing 2: Script

```
# inferential method on drawn sample
fit = estimate(
    df=drawn.df,
    formula="ord~0|0",
    gen_pars={
        "pi": drawn.pars[0],
        "xi": drawn.pars[1]
    }
)
# print the summary of MLE
print(fit.summary())
# show the plot of MLE
fit.plot()
plt.show()
```

```
warnings.warn("No m given, max(ordinal) has been taken")

================================================================================
=====>>> CUB00 model <<<===== ML-estimates
================================================================================
m=10  Size=500  Iterations=13  Maxiter=500  Tol=1E-04
--------------------------------------------------------------------------------
Uncertainty
    Estimates  StdErr    Wald  p-value
pi     +0.675   0.034  19.872   0.0000
--------------------------------------------------------------------------------
Feeling
    Estimates  StdErr    Wald  p-value
xi     +0.188   0.009  20.808   0.0000
--------------------------------------------------------------------------------
Correlation   = 0.2105
================================================================================
Dissimilarity = 0.0599
Loglik(sat)   = -994.063
Loglik(MOD)   = -1000.111
Loglik(uni)   = -1151.293
Mean-loglik   = -2.000
Deviance      = 12.096
--------------------------------------------------------------------------------
AIC = 2004.22
BIC = 2012.65
================================================================================
Elapsed time=0.00187 seconds =====>>> Wed Apr 24 11:27:35 2024
================================================================================
```


CUB model $n = 500$
Estim($\pi = 0.675$, $\xi = 0.188$) Dissim(est,obs)=0.0599
Gener($\pi = 0.700$, $\xi = 0.200$)
Probability mass
Ordinal
estimated
observed
generating


Corr(π, ξ)= 0.2105
$(1-\xi)$ feeling
$(1-\pi)$ uncertainty
estimated
generating
CR 95%


Corr(π, ξ)= 0.2105
$(1-\xi)$ feeling
$(1-\pi)$ uncertainty
estimated
generating
CR 95%

Calling `fit.as_dataframe()` will return a DataFrame with parameters' estimated values and standard errors

```
     component parameter  estimate    stderr       wald        pvalue
0  Uncertainty        pi   0.67476  0.033954  19.872485  7.042905e-88
1      Feeling        xi   0.18817  0.009043  20.807551  3.697579e-96
```

2.2.3 With covariates

$$\Pr(R_i = r|\boldsymbol{\theta}, \boldsymbol{y}_i, \boldsymbol{w}_i) = \pi_i \binom{m-1}{r-1} (1-\xi_i)^{r-1} \xi_i^{m-r} + \frac{1-\pi_i}{m}$$

$$\begin{cases} \pi_i = \dfrac{1}{1+\exp\{-\boldsymbol{y}_i\boldsymbol{\beta}\}} \\ \xi_i = \dfrac{1}{1+\exp\{-\boldsymbol{w}_i\boldsymbol{\gamma}\}} \end{cases}$$

All three combinations of covariates has been implemented for CUB family in both Python and R: for *uncertainty* only, for *feeling* only, and for *both*.

Here we'll show an example with covariates for *feeling* only.

First of all, we'll draw a random sample with two covariates for the *feeling* component: `W1` and `W2`. Note that, having two covariates, we'll need three γ parameters, to consider the constant term too.

Listing 3: Script

```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from cubmods.gem import draw, estimate

# Draw a random sample
n = 1000
np.random.seed(1)
W1 = np.random.randint(1, 10, n)
np.random.seed(42)
W2 = np.random.random(n)
df = pd.DataFrame({
    "W1": W1, "W2": W2
})
drawn = draw(
    formula="res ~ 0 | W1 + W2",
    df=df,
    m=10, n=n,
    pi=0.8,
    gamma=[2.3, 0.2, -5],
)
# print the summary
print(drawn.summary())
```

```
=========================================================================
=====>>> CUB(0W) model <<<===== Drawn random sample
=========================================================================
```

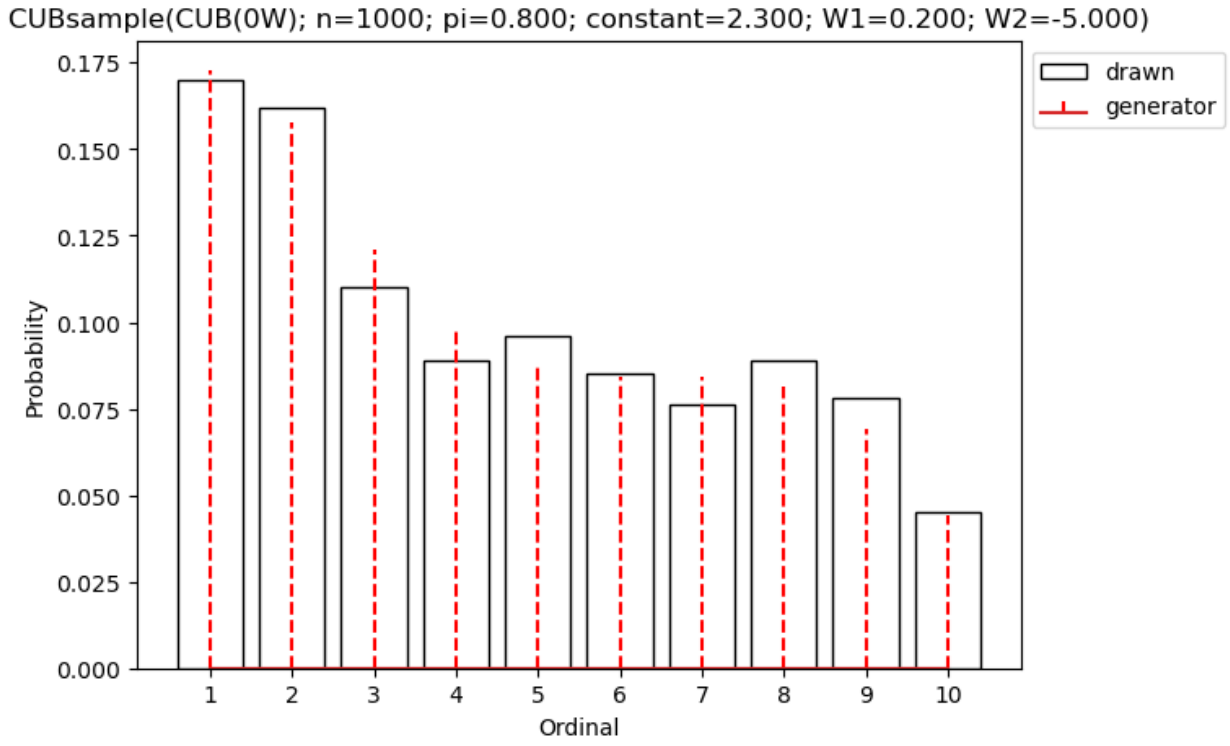
(continues on next page)

(continued from previous page)

```
m=10  Sample size=1000  seed=None
formula: res~0|W1+W2
--------------------------------------------------------------------------------
pi=0.800
constant=2.300
W1=0.200
W2=-5.000
================================================================================
Sample metrics
Mean     = 4.566000
Variance = 8.089734
Std.Dev. = 2.844246
--------------------------------------------------------------------------------
Dissimilarity = 0.0307673
================================================================================
```

Listing 4: Script

```
1 # plot the drawn sample
2 drawn.plot()
3 plt.show()
```



Listing 5: Script

```
# print the parameters' values
print(drawn.as_dataframe())
```

```
  parameter  value
0        pi    0.8
1  constant    2.3
2        W1    0.2
3        W2   -5.0
```

Listing 6: Script

```
# print the updated DataFrame
print(drawn.df)
```

```
     W1        W2  res
0     6  0.374540    2
1     9  0.950714    7
2     6  0.731994    8
3     1  0.598658    8
4     1  0.156019    4
..   ..       ...  ...
995   3  0.091582    2
996   9  0.917314    9
997   4  0.136819    1
998   7  0.950237    3
999   8  0.446006    2

[1000 rows x 3 columns]
```

Finally, we'll call `estimate` to estimate the parameters given the observed (actually, drawn) sample.

Listing 7: Script

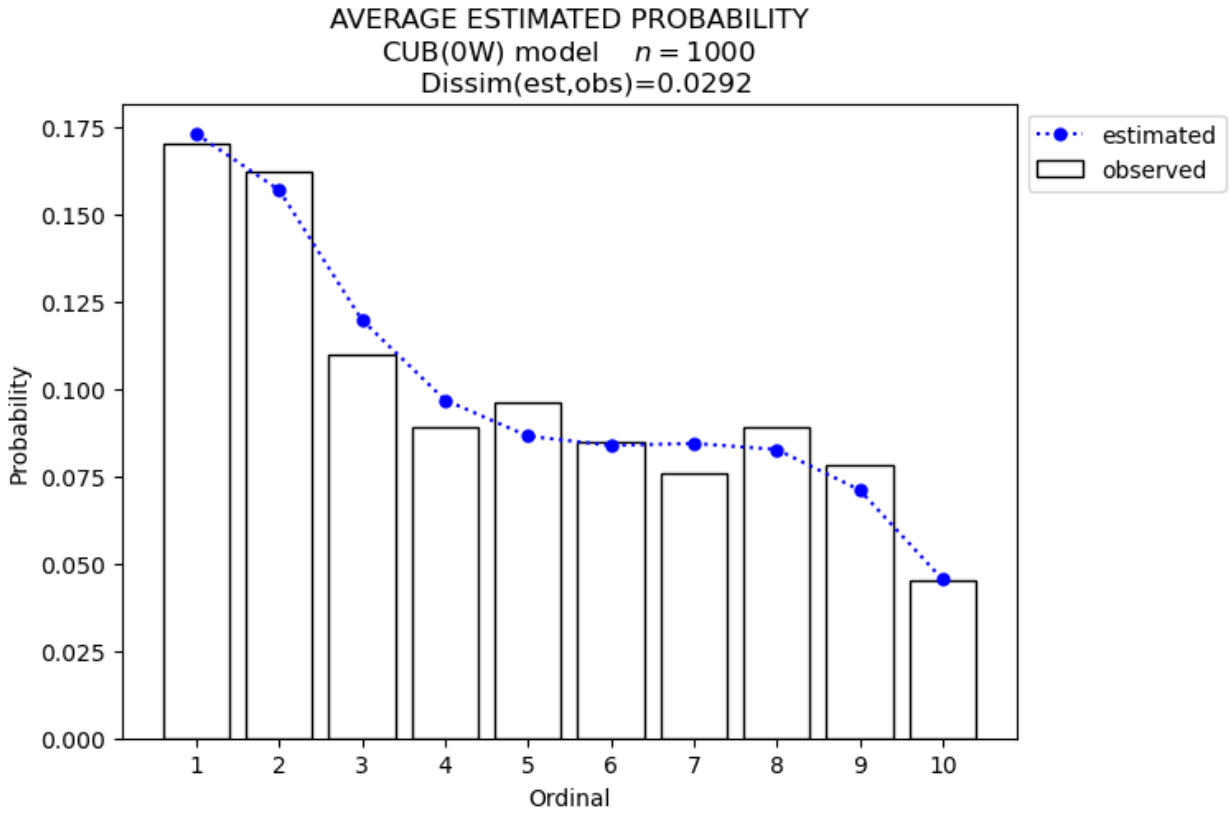
```
# MLE estimation
fit = estimate(
    formula="res ~ 0 | W1+W2",
    df=drawn.df,
)
# Print MLE summary
print(fit.summary())
# plot the results
fit.plot()
plt.show()
```

```
warnings.warn("No m given, max(ordinal) has been taken")
=============================================================================
=====>>> CUB(0W) model <<<===== ML-estimates
=============================================================================
m=10  Size=1000  Iterations=18  Maxiter=500  Tol=1E-04
-----------------------------------------------------------------------------
```

(continues on next page)

(continued from previous page)

```
Uncertainty
          Estimates  StdErr     Wald  p-value
pi            0.800  0.0198   40.499   0.0000
------------------------------------------------------------------------------
Feeling
          Estimates  StdErr     Wald  p-value
constant      2.353  0.1001   23.514   0.0000
W1            0.194  0.0138   14.034   0.0000
W2           -5.076  0.1454  -34.909   0.0000
==============================================================================
Dissimilarity = 0.0292
Loglik(MOD)   = -1807.052
Loglik(uni)   = -2302.585
Mean-loglik   = -1.807
------------------------------------------------------------------------------
AIC = 3622.10
BIC = 3641.74
==============================================================================
Elapsed time=0.09656 seconds =====>>> Thu Aug 15 18:31:21 2024
==============================================================================
```



2.3 CUBSH family

Basic family of the class CUB with shelter effect. See the references for details.

2.3.1 References

1. Maria Iannario. Modelling shelter choices in a class of mixture models for ordinal responses. *Statistical Methods & Applications*, 21:1–22, 2012.
2. Maria Iannario and Domenico Piccolo. A new statistical model for the analysis of customer satisfaction. *Quality Technology & Quantitative Management*, 7(2):149–168, 2010.
3. Maria Iannario, Domenico Piccolo, and others. Inference for cub models: a program in r. *Statistica & Applicazioni*, 12(2):177–204, 2014.
4. Domenico Piccolo and Rosaria Simone. The class of cub models: statistical foundations, inferential issues and empirical evidence. *Statistical Methods & Applications*, 28:389–435, 2019.

2.3.2 Without covariates

A model of the CUB family with shelter effect for responses with m ordinal categories, without covariates is specified as

$$\Pr(R = r|\boldsymbol{\theta}) = \delta D_r^{(c)} + (1 - \delta)\left(\pi b_r(\xi) + \frac{1-\pi}{m}\right)$$

where π and ξ are the parameters for respectively the *uncertainty* and the *feeling* components, and δ is the weight of the shelter effect.

In the next example, we'll draw an ordinal response and then estimate the parameters given the sample.

Listing 8: Script

```
# import libraries
import matplotlib.pyplot as plt
from cubmods.gem import draw, estimate

# draw a sample
drawn = draw(
    formula="ord ~ 0 | 0 | 0",
    m=7, sh=1,
    pi=.8, xi=.4, delta=.15,
    n=1500, seed=42)

print(drawn.as_dataframe())
```

```
  parameter  value
0       pi1   0.68
1       pi2   0.17
2        xi   0.40
3       *pi   0.80
4    *delta   0.15
```

Notice that:

- since "cub" is default value of the *kwarg* model, we do not need to specify it
- we'll pass to estimate *kwargs* values taken from the object drawn

Listing 9: Script

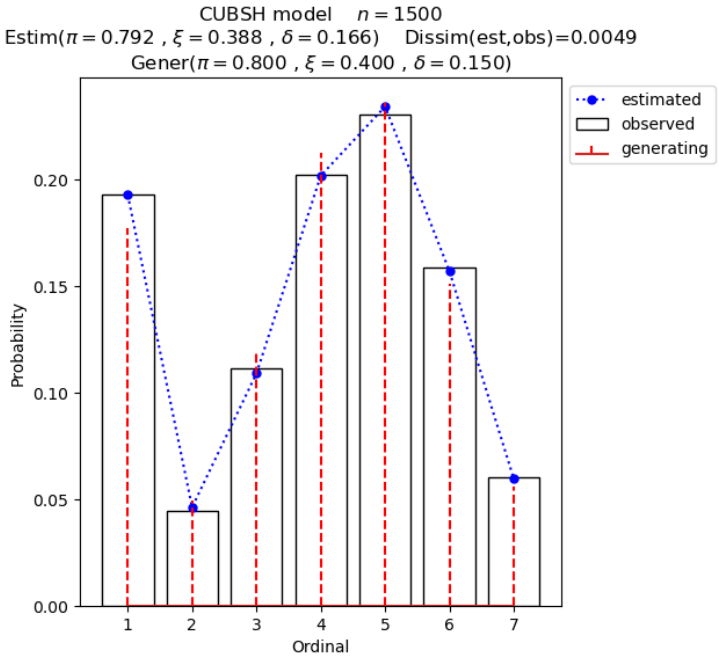
```
# inferential method on drawn sample
fit = estimate(
    df=drawn.df, sh=drawn.sh,
    formula=drawn.formula,
    gen_pars={
        "pi1": drawn.pars[0],
        "pi2": drawn.pars[1],
        "xi": drawn.pars[2],
    }
)
# print the summary of MLE
print(fit.summary())
# show the plot of MLE
fit.plot()
plt.show()
```

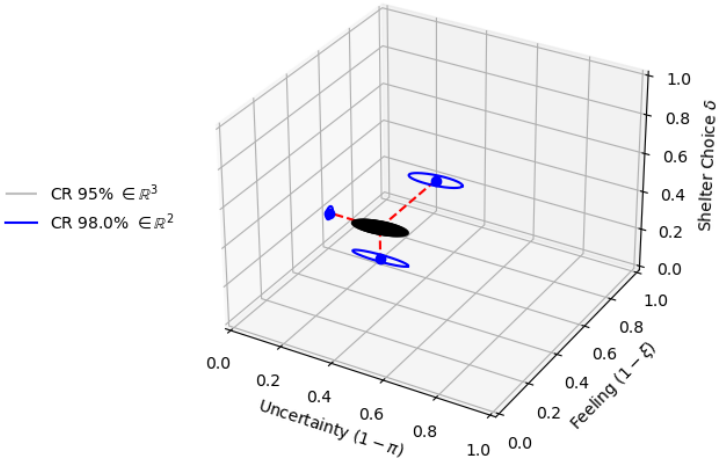
```
warnings.warn("No m given, max(ordinal) has been taken")
================================================================================
=====>>> CUBSH model <<<===== ML-estimates
================================================================================
m=7  Shelter=1  Size=1500  Iterations=59  Maxiter=500  Tol=1E-04
--------------------------------------------------------------------------------
Alternative parametrization
       Estimates  StdErr    Wald  p-value
pi1        0.661  0.0307  21.508   0.0000
pi2        0.174  0.0344   5.041   0.0000
xi         0.388  0.0077  50.592   0.0000
--------------------------------------------------------------------------------
Uncertainty
       Estimates  StdErr    Wald  p-value
pi         0.792  0.0400  19.813   0.0000
--------------------------------------------------------------------------------
Feeling
       Estimates  StdErr    Wald  p-value
xi         0.388  0.0077  50.592   0.0000
--------------------------------------------------------------------------------
Shelter effect
       Estimates  StdErr    Wald  p-value
delta      0.166  0.0116  14.327   0.0000
================================================================================
Dissimilarity = 0.0049
Loglik(sat)   = -2734.302
Loglik(MOD)   = -2734.433
Loglik(uni)   = -2918.865
Mean-loglik   = -1.823
Deviance      = 0.263
--------------------------------------------------------------------------------
AIC = 5474.87
```

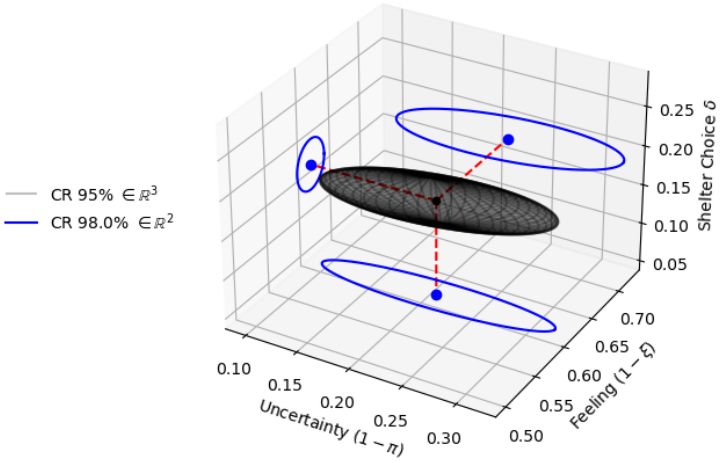
(continues on next page)

(continued from previous page)

```
BIC = 5490.81
================================================================================
```


CUBSH model $n = 1500$
Estim($\pi = 0.792$, $\xi = 0.388$, $\delta = 0.166$) Dissim(est,obs)=0.0049
Gener($\pi = 0.800$, $\xi = 0.400$, $\delta = 0.150$)
estimated
observed
generating
Probability
Ordinal


CR 95% $\in \mathbb{R}^3$
CR 98.0% $\in \mathbb{R}^2$
Uncertainty $(1-\pi)$
Feeling $(1-\xi)$
Shelter Choice δ


CR 95% $\in \mathbb{R}^3$
CR 98.0% $\in \mathbb{R}^2$
Uncertainty $(1-\pi)$
Feeling $(1-\xi)$
Shelter Choice δ

2.3.3 With covariates

$$\Pr(R_i = r|\boldsymbol{\theta}, \boldsymbol{y}_i, \boldsymbol{w}_i, \boldsymbol{x}_i) = \delta_i D_r^{(c)} + (1-\delta_i)\left(\pi_i b_r(\xi_i) + \frac{1-\pi_i}{m}\right)$$

$$\begin{cases} \pi_i = \dfrac{1}{1+\exp\{-\boldsymbol{y}_i\boldsymbol{\beta}\}} \\ \xi_i = \dfrac{1}{1+\exp\{-\boldsymbol{w}_i\boldsymbol{\gamma}\}} \\ \delta_i = \dfrac{1}{1+\exp\{-\boldsymbol{x}_i\boldsymbol{\omega}\}} \end{cases}$$

Only the model with covariates for all components has been currently defined and implemented.

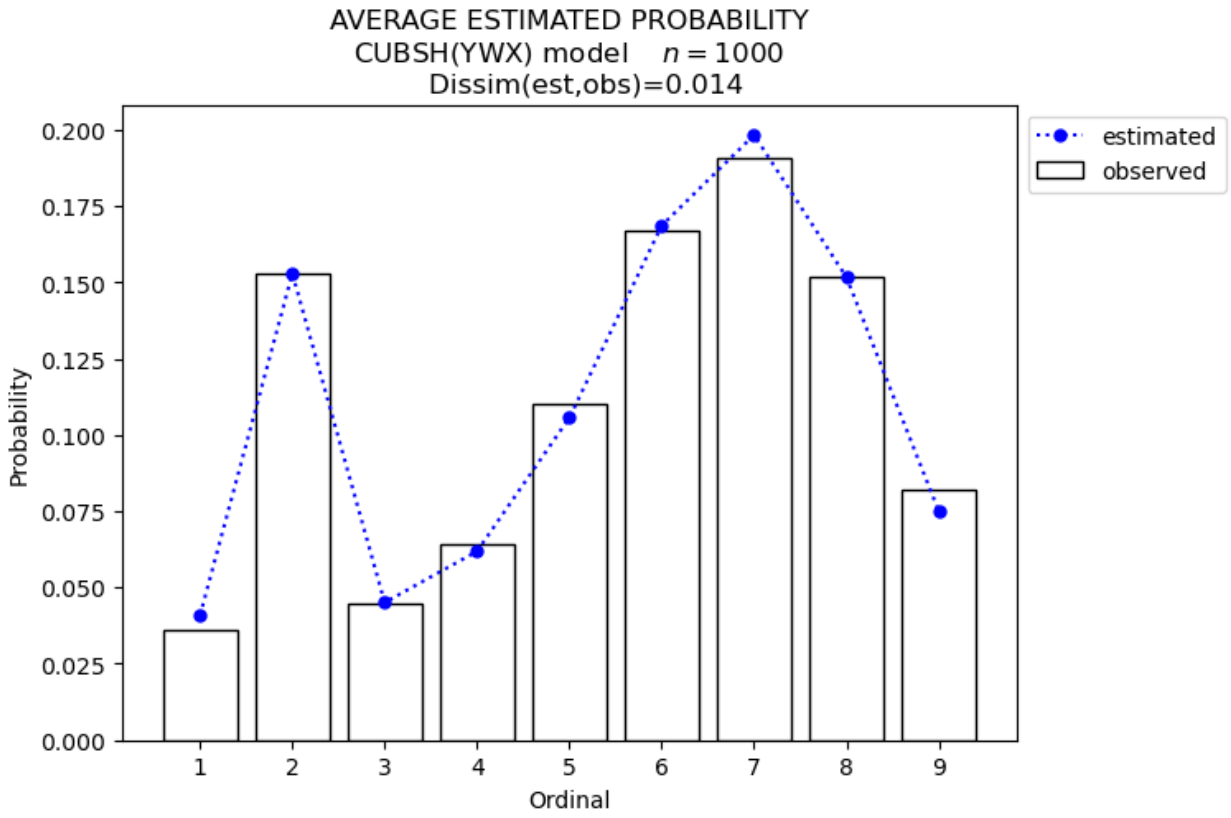
Nevertheless, thanks to the symbol 1 provided by the *formula*, we can specify a different combination of covariates.

For example, we'll specifiy a model CUB with shelter effect, with covariates for uncertainty only. We'll use the function `logit` to have better 'control' of the parameters values, because $\gamma_0 = \text{logit}(\xi)$ and similarly for π and δ.

Listing 10: Script

```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from cubmods.general import expit, logit
from cubmods.gem import draw, estimate

# Draw a random sample
n = 1000
np.random.seed(1)
W1 = np.random.randint(1, 10, n)
df = pd.DataFrame({
    "W1": W1,
})
drawn = draw(
    formula="fee ~ W1 | 1 | 1",
    df=df,
    m=9, sh=2,
    beta=[logit(.8), -.2],
    gamma=[logit(.3)],
    omega=[logit(.12)],
)

# MLE estimation
fit = estimate(
    formula="fee ~ W1 | 1 | 1",
    df=drawn.df, sh=2,
)
# Print MLE summary
print(fit.summary())
# plot the results
fit.plot()
plt.show()
```

```
warnings.warn("No m given, max(ordinal) has been taken")
================================================================================
=====>>> CUBSH(YWX) model <<<===== ML-estimates
================================================================================
m=9  Shelter=2  Size=1000  Iterations=25  Maxiter=500  Tol=1E-04
--------------------------------------------------------------------------------
Uncertainty
          Estimates  StdErr     Wald  p-value
constant      0.992  0.3314    2.994   0.0028
W1           -0.127  0.0569   -2.228   0.0259
--------------------------------------------------------------------------------
Feeling
          Estimates  StdErr     Wald  p-value
constant     -0.902  0.0381  -23.662   0.0000
--------------------------------------------------------------------------------
Shelter effect
          Estimates  StdErr     Wald  p-value
constant     -2.074  0.1260  -16.462   0.0000
================================================================================
Dissimilarity = 0.0139
Loglik(MOD)   = -2069.978
Loglik(uni)   = -2197.225
Mean-loglik   = -2.070
--------------------------------------------------------------------------------
AIC = 4147.96
BIC = 4167.59
================================================================================
Elapsed time=1.43850 seconds =====>>> Thu Aug 15 19:39:49 2024
================================================================================
```



To get the estimated values of $\hat{\xi}$ and $\hat{\delta}$ we can use the function `expit` because $\hat{\xi} = \text{expit}(\hat{\gamma}_0)$ and similarly for $\hat{\delta}$. Then, since $\widehat{es}(\xi) = \text{expit}[\hat{\gamma}_0 + \widehat{es}(\gamma_0)] - \hat{\xi}$ we can compute the standard errors of both $\hat{\xi}$ and $\hat{\delta}$.

Listing 11: Script

```
est_xi = expit(fit.estimates[2])
est_de = expit(fit.estimates[3])
est_xi_se = expit(fit.estimates[2]+fit.stderrs[2]) - est_xi
est_de_se = expit(fit.estimates[3]+fit.stderrs[3]) - est_de
print(
    "      estimates  stderr\n"
    f"xi       {est_xi:.4f}  {est_xi_se:.4f}"
    "\n"
    f"delta    {est_de:.4f}  {est_de_se:.4f}"
)
```

```
     estimates  stderr
xi      0.2886  0.0079
delta   0.1116  0.0131
```

which, in fact, match the values used to draw the sample.

2.4 CUSH family

Basic family of the class CUSH with a single shelter effect. See the references for details.

2.4.1 References

1. Stefania Capecchi and Domenico Piccolo. Dealing with heterogeneity in ordinal responses. *Quality & Quantity*, 51:2375–2393, 2017.
2. Domenico Piccolo and Rosaria Simone. The class of cub models: statistical foundations, inferential issues and empirical evidence. *Statistical Methods & Applications*, 28:389–435, 2019.

2.4.2 Without covariates

$$\Pr(R = r|\boldsymbol{\theta}) = \delta D_r^{(c)} + (1 - \delta)/m$$

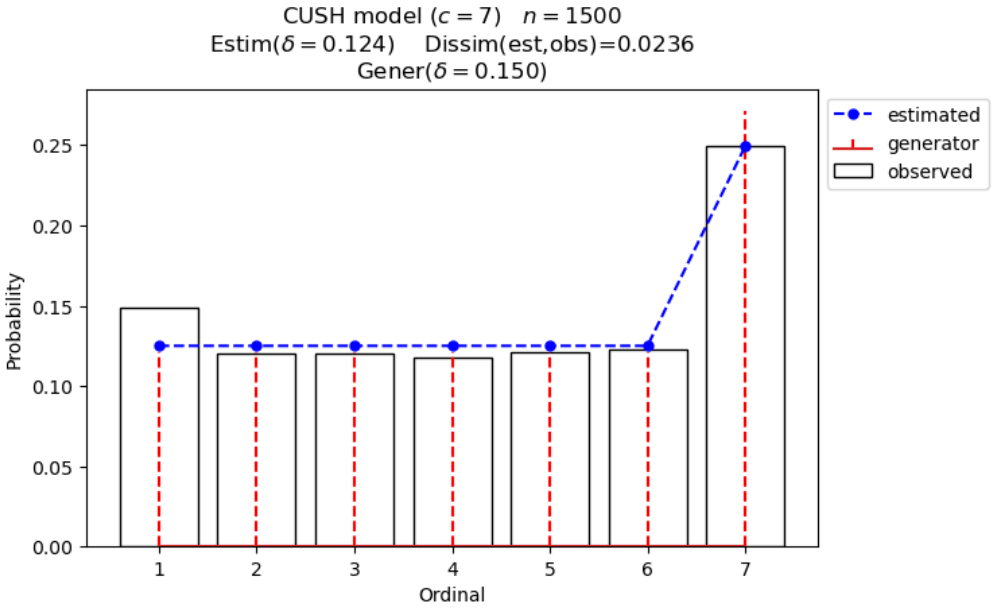
In the example, we'll draw a sample from a CUSH model without covariates and then estimate the parameter given the observed sample.

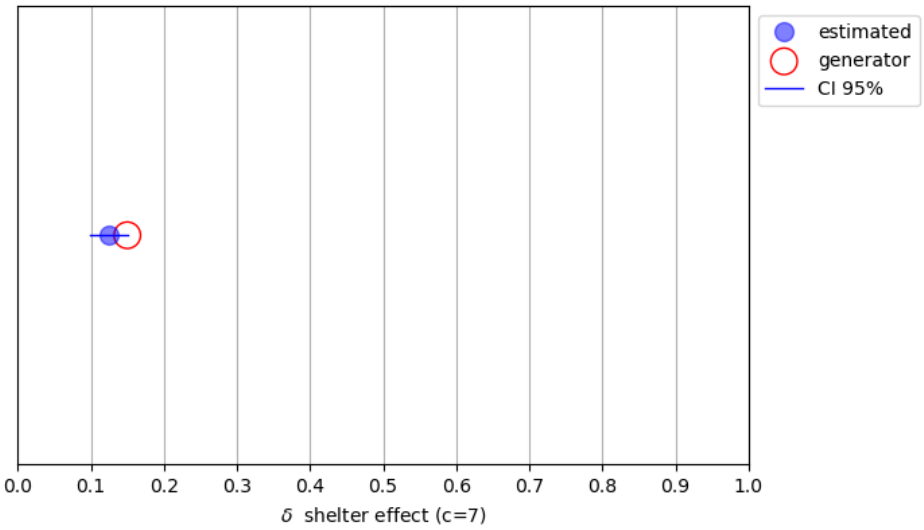
Notice that, since the `model` is not the default `"cub"`, we need to specify it.

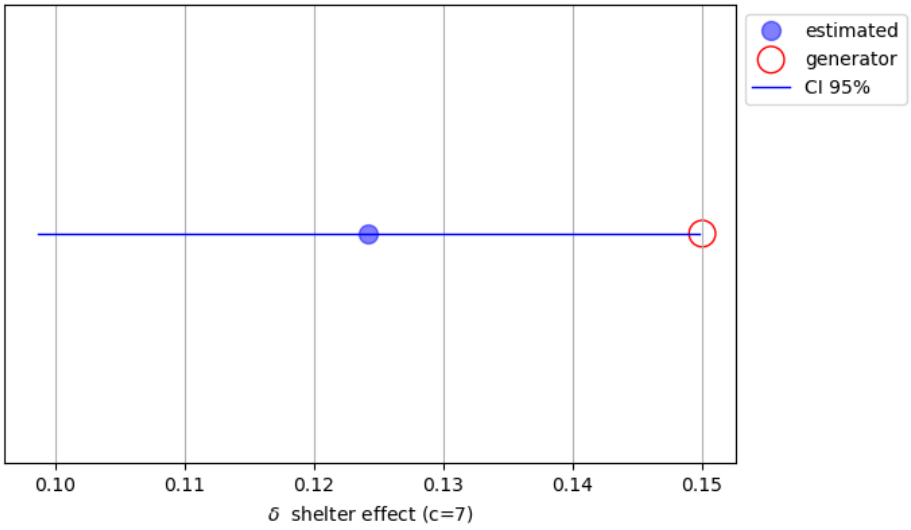
Listing 12: Script

```
# import libraries
import matplotlib.pyplot as plt
from cubmods.gem import draw, estimate

# draw a sample
drawn = draw(
    formula="ord ~ 0",
    model="cush",
    sh=7,
    m=7, delta=.15,
    n=1500, seed=76)

# inferential method on drawn sample
fit = estimate(
    df=drawn.df,
    model="cush",
    formula="ord~0",
    sh=7,
    gen_pars={
        "delta": drawn.pars[0],
    }
)
# print the summary of MLE
print(fit.summary())
# show the plot of MLE
fit.plot()
plt.show()
```

```
warnings.warn("No m given, max(ordinal) has been taken")
================================================================================
=====>>> CUSH model <<<===== ML-estimates
================================================================================
m=7  Shelter=7  Size=1500
--------------------------------------------------------------------------------
Shelter effect
       Estimates  StdErr   Wald  p-value
delta      0.124  0.0130  9.532   0.0000
================================================================================
Dissimilarity = 0.0236
Loglik(sat)   = -2856.039
Loglik(MOD)   = -2859.923
Loglik(uni)   = -2918.865
Mean-loglik   = -1.907
Deviance      = 7.768
--------------------------------------------------------------------------------
AIC = 5721.85
BIC = 5727.16
================================================================================
Elapsed time=0.00113 seconds =====>>> Fri Aug 16 10:44:07 2024
================================================================================
```


CUSH model ($c = 7$) $n = 1500$
Estim($\delta = 0.124$) Dissim(est,obs)=0.0236
Gener($\delta = 0.150$)
estimated
generator
observed
Probability
0.25
0.20
0.15
0.10
0.05
0.00
1
2
3
4
5
6
7
Ordinal


estimated
generator
CI 95%
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
δ shelter effect (c=7)


estimated
generator
CI 95%
0.10
0.11
0.12
0.13
0.14
0.15
δ shelter effect (c=7)

2.4.3 With covariates

$$\Pr(R_i = r|\boldsymbol{\theta}, \boldsymbol{x}_i) = \delta_i D_r^{(c)} + (1 - \delta_i)/m$$

$$\delta_i = \frac{1}{1 + \exp\{-\boldsymbol{x}_i\boldsymbol{\omega}\}}$$

In the example, we'll draw a sample from a CUSH model with covariates and then estimate the parameter given the observed sample.

Notice that, since the `model` is not the default `"cub"`, we need to specify it.

Listing 13: Script

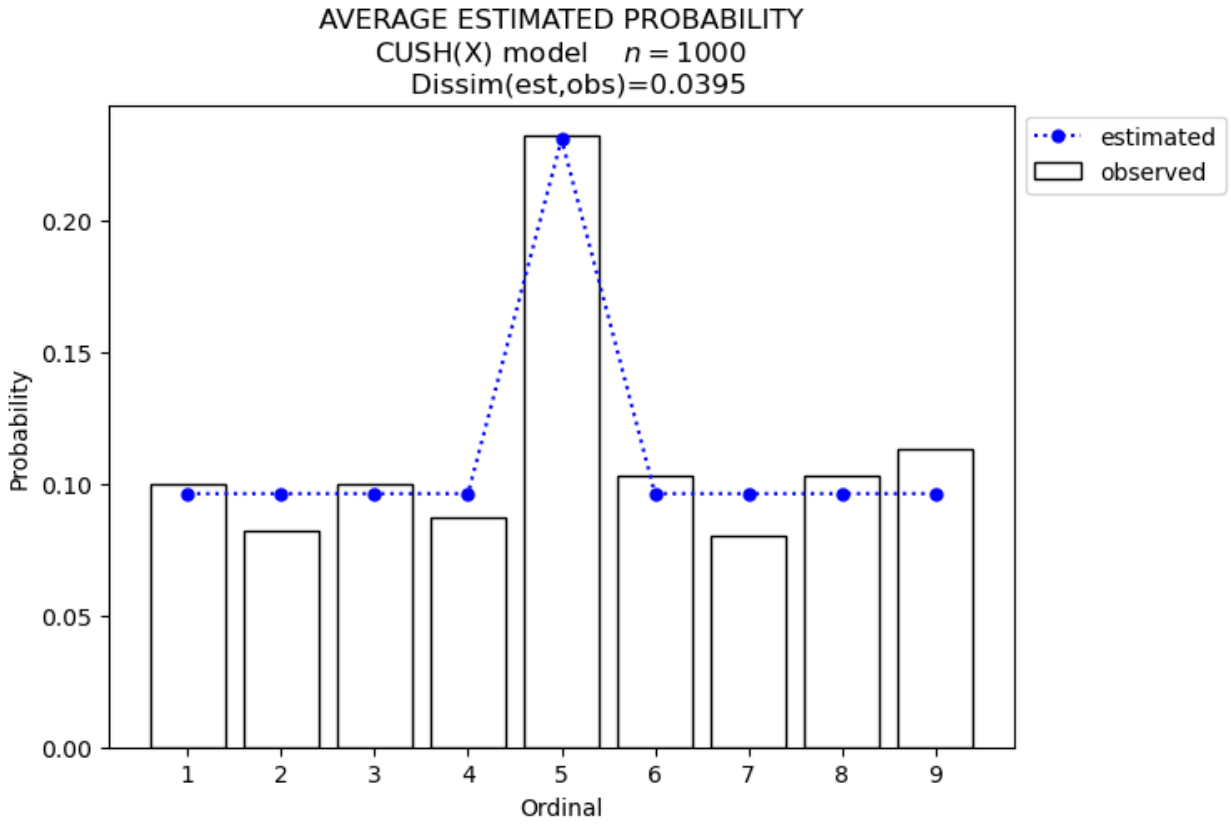
```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from cubmods.general import logit
from cubmods.gem import draw, estimate

# Draw a random sample
n = 1000
np.random.seed(1)
X = np.random.randint(1, 10, n)
df = pd.DataFrame({
    "X": X,
})
drawn = draw(
    formula="fee ~ X",
    model="cush",
    df=df,
    m=9, sh=5,
    omega=[logit(.05), .2],
)

# MLE estimation
fit = estimate(
    formula="fee ~ X",
    model="cush",
    df=drawn.df, sh=5,
)
# Print MLE summary
print(fit.summary())
# plot the results
fit.plot()
plt.show()
```

```
warnings.warn("No m given, max(ordinal) has been taken")
=========================================================================
=====>>> CUSH(X) model <<<===== ML-estimates
=========================================================================
m=9  Shelter=5  Size=1000
-------------------------------------------------------------------------
Shelter effect
```

(continues on next page)

(continued from previous page)

```
          Estimates  StdErr    Wald  p-value
constant     -3.131  0.4361  -7.180   0.0000
X             0.229  0.0629   3.642   0.0003
================================================================================
Dissimilarity = 0.0395
Loglik(MOD)   = -2130.030
Loglik(uni)   = -2197.225
Mean-loglik   = -2.130
--------------------------------------------------------------------------------
AIC = 4264.06
BIC = 4273.87
================================================================================
Elapsed time=0.01704 seconds =====>>> Fri Aug 16 10:54:11 2024
================================================================================
```



2.5 CUSH2 family

Family of the class CUSH with two shelter effects (CUSH2). See the references for details.

2.5.1 References

1. Massimo Pierini. Modelli della classe cub in python. *Universitas Mercatorum, Rome, IT*, pages 16–20, June 2024. (Bachelor's thesis L-41).

2.5.2 Without covariates

$$\Pr(R = r|\boldsymbol{\theta}) = \delta_1 D_r^{(c_1)} + \delta_2 D_r^{(c_2)} + (1 - \delta_1 - \delta_2)/m$$

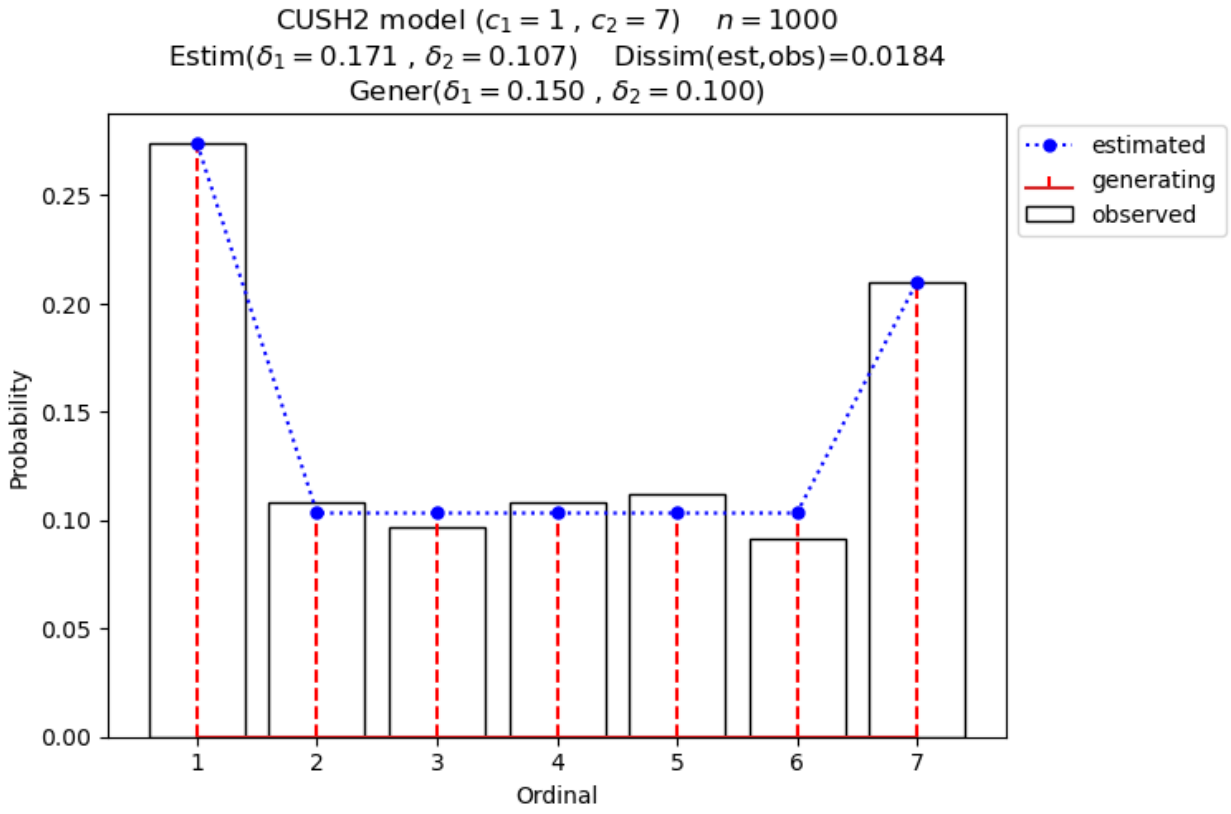
In the example, we'll draw a sample from a CUSH2 model without covariates and then estimate the parameter given the observed sample.

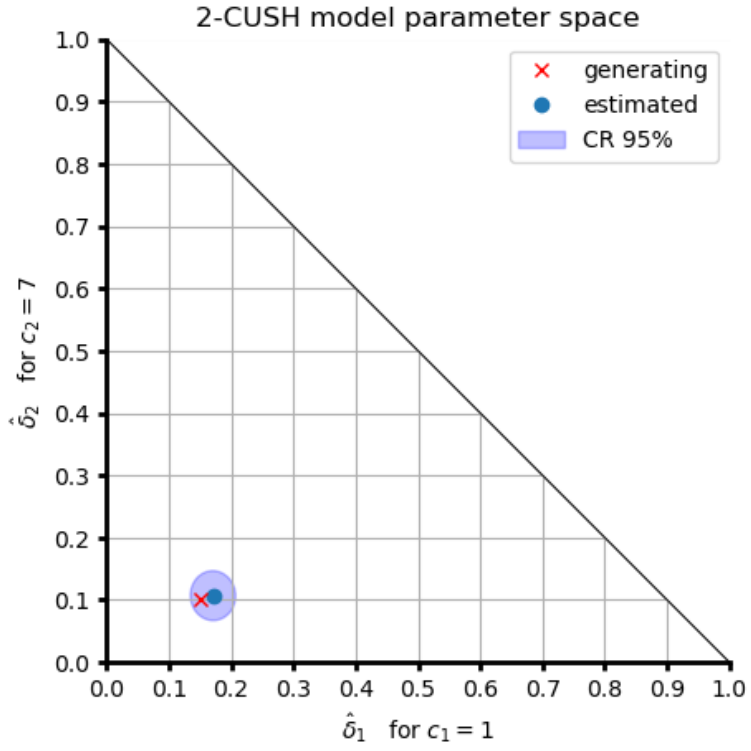
Notice that, since the `model` is not the default `"cub"`, we need to specify it. Passing a list of two shelter categories with the *kwarg* `sh`, a CUSH2 model will be called.

Listing 14: Script

```
# import libraries
import matplotlib.pyplot as plt
from cubmods.gem import draw, estimate

# draw a sample
drawn = draw(
    formula="ord ~ 0 | 0",
    model="cush",
    sh=[1,7],
    m=7,
    delta1=.15, delta2=.1,
    n=1000, seed=42)

# inferential method on drawn sample
fit = estimate(
    df=drawn.df,
    model="cush",
    formula="ord~0|0",
    sh=[1,7],
    gen_pars={
        "delta1": drawn.pars[0],
        "delta2": drawn.pars[1],
    }
)
# print the summary of MLE
print(fit.summary())
# show the plot of MLE
fit.plot()
plt.show()
```

```
warnings.warn("No m given, max(ordinal) has been taken")
==========================================================================
=====>>> CUSH2 model <<<===== ML-estimates
==========================================================================
m=7  Shelter=[1 7]  Size=1000
--------------------------------------------------------------------------
Shelter effects
        Estimates  StdErr    Wald  p-value
delta1      0.171  0.0148  11.535   0.0000
delta2      0.107  0.0163   6.555   0.0000
==========================================================================
Dissimilarity = 0.0184
Loglik(sat)   = -1852.818
Loglik(MOD)   = -1854.344
Loglik(uni)   = -1945.910
Mean-loglik   = -1.854
Deviance      = 3.053
--------------------------------------------------------------------------
AIC = 3712.69
BIC = 3722.50
==========================================================================
Elapsed time=0.00228 seconds =====>>> Fri Aug 16 11:15:21 2024
==========================================================================
```


CUSH2 model ($c_1 = 1$, $c_2 = 7$) $n = 1000$
Estim($\delta_1 = 0.171$, $\delta_2 = 0.107$) Dissim(est,obs)=0.0184
Gener($\delta_1 = 0.150$, $\delta_2 = 0.100$)
estimated
generating
observed
Probability
0.00
0.05
0.10
0.15
0.20
0.25
1
2
3
4
5
6
7
Ordinal


2-CUSH model parameter space
generating
estimated
CR 95%
$\hat{\delta}_2$ for $c_2 = 7$
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
$\hat{\delta}_1$ for $c_1 = 1$

2.5.3 With covariates

$$\Pr(R_i = r|\boldsymbol{\theta}, \boldsymbol{x}_{1i}, \boldsymbol{x}_{2i}) = \delta_{1i} D_r^{(c_1)} + \delta_{2i} D_r^{(c_2)} + (1 - \delta_{1i} - \delta_{2i})/m$$

$$\begin{cases} \delta_{1i} = \dfrac{1}{1 + \exp\{-\boldsymbol{x}_{1i}\boldsymbol{\omega}_1\}} \\ \delta_{2i} = \dfrac{1}{1 + \exp\{-\boldsymbol{x}_{2i}\boldsymbol{\omega}_2\}} \end{cases}$$

In this example we'll draw a sample from a CUSH2 model with covariates for the first shelter choice only and will then estimate the parameters with a CUSH2 model with covariates for both shelter choices but using the symbol 1 in the formula for the second shelter choice to estimate the constant parameter only. This is usually not needed, but we do it here to confirm that $\text{expit}(\hat{\omega}_{20}) = \hat{\delta}_2$.

Notice that, since the `model` is not the default `"cub"`, we need to specify it.

Listing 15: Script

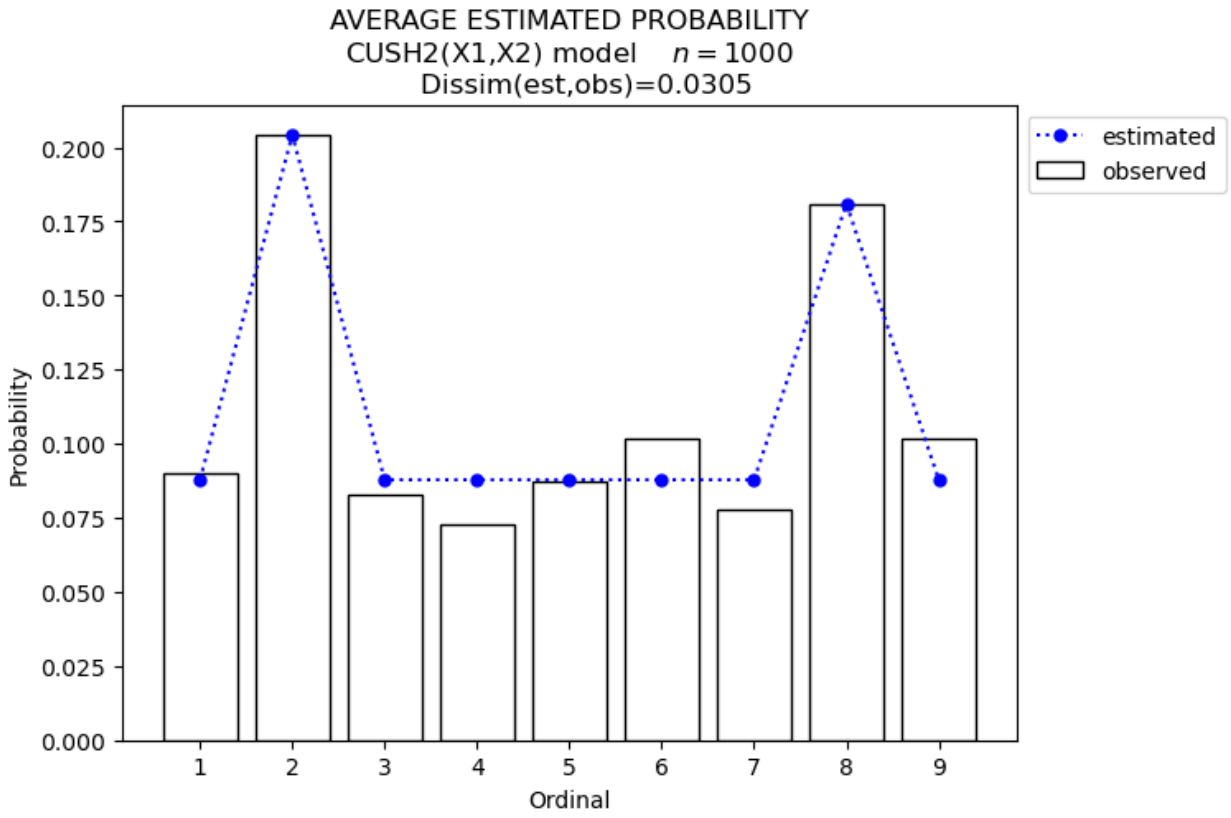
```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from cubmods.general import logit, expit
from cubmods.gem import draw, estimate

# Draw a random sample
n = 1000
np.random.seed(1)
X = np.random.randint(1, 10, n)
df = pd.DataFrame({
    "X": X,
})
drawn = draw(
    formula="fee ~ X | 0",
    model="cush",
    df=df,
    m=9, sh=[2, 8],
    omega1=[logit(.05), .2],
    delta2=.1
)

# MLE estimation
fit = estimate(
    formula="fee ~ X | 1",
    model="cush",
    df=drawn.df, sh=[2, 8],
)
# Print MLE summary
print(fit.summary())
# plot the results
fit.plot()
plt.show()

est_de2 = expit(fit.estimates[2])
est_de2_es = expit(fit.estimates[2]+fit.stderrs[2]) - est_de2
```

(continues on next page)

(continued from previous page)

```
print(
    "     estimates  stderr\n"
    f"delta2  {est_de2:.4f}  {est_de2_es:.4f}"
)
```

```
warnings.warn("No m given, max(ordinal) has been taken")
================================================================================
=====>>> CUSH2(X1,X2) model <<<===== ML-estimates
================================================================================
m=9  Shelter=[2 8]  Size=1000
--------------------------------------------------------------------------------
Shelter effect 1
          Estimates  StdErr     Wald  p-value
constant     -3.170  0.4216   -7.519   0.0000
X             0.207  0.0613    3.379   0.0007
--------------------------------------------------------------------------------
Shelter effect 2
          Estimates  StdErr     Wald  p-value
constant     -2.276  0.1609  -14.149   0.0000
================================================================================
Dissimilarity = 0.0305
Loglik(MOD)   = -2122.463
Loglik(uni)   = -2197.225
Mean-loglik   = -2.122
--------------------------------------------------------------------------------
AIC = 4250.93
BIC = 4265.65
================================================================================
Elapsed time=0.06553 seconds =====>>> Fri Aug 16 11:29:11 2024
================================================================================
```




```
     estimates  stderr
delta2  0.0931  0.0145
```

2.6 CUBE family

Family of the class CUBE (Combination of Uniform and BEtaBinomial). See the references for details.

2.6.1 References

1. Maria Iannario. Modelling uncertainty and overdispersion in ordinal data. *Communications in Statistics-Theory and Methods*, 43(4):771–786, 2014.
2. Maria Iannario. Detecting latent components in ordinal data with overdispersion by means of a mixture distribution. *Quality & Quantity*, 49:977–987, 2015.
3. Domenico Piccolo and Rosaria Simone. The class of cub models: statistical foundations, inferential issues and empirical evidence. *Statistical Methods & Applications*, 28:389–435, 2019.

2.6.2 Without covariates

$$\Pr(R = r|\boldsymbol{\theta}) = \pi b_r(p) + (1 - \pi)/m$$

$$p \sim \text{Beta}(\alpha, \beta)$$

$$\begin{cases} \xi = \dfrac{\beta}{\alpha + \beta} \\ \phi = \dfrac{1}{\alpha + \beta} \end{cases}$$

In this example, we'll draw a sample from a CUBE model and then will estimate the parameters given the observed sample.

Notice that, since the `model` is not the default `"cub"`, we need to specify it.

Listing 16: Script

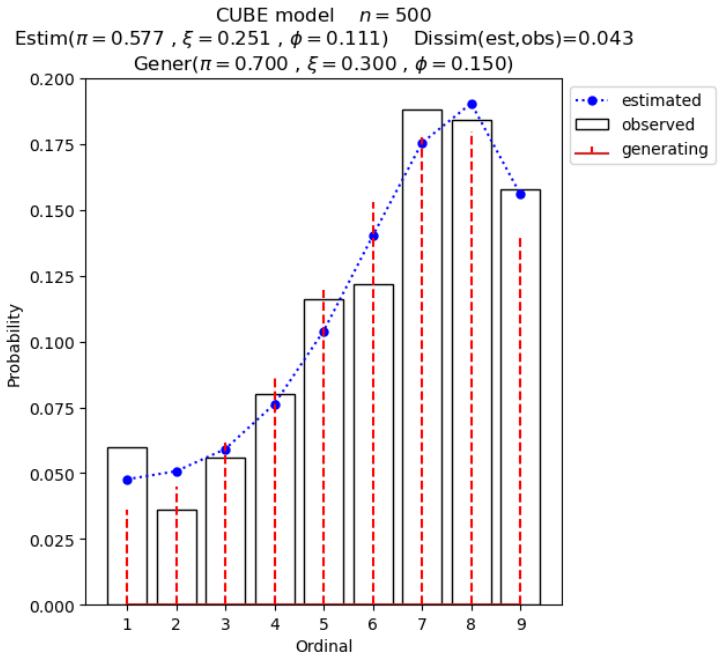
```
# import libraries
import matplotlib.pyplot as plt
from cubmods.gem import draw, estimate

# draw a sample
drawn = draw(
    formula="ord ~ 0 | 0 | 0",
    model="cube",
    m=9, pi=.7, xi=.3, phi=.15,
    n=500, seed=1)

# inferential method on drawn sample
fit = estimate(
    df=drawn.df,
    formula="ord~0|0|0",
    model="cube",
    gen_pars={
        "pi": drawn.pars[0],
        "xi": drawn.pars[1],
        "phi": drawn.pars[2],
    }
)
# print the summary of MLE
print(fit.summary())
# show the plot of MLE
fit.plot()
plt.show()
```

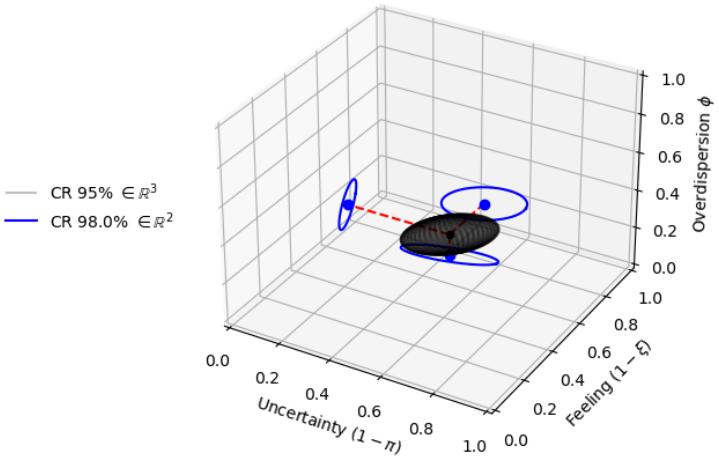
```
warnings.warn("No m given, max(ordinal) has been taken")
=========================================================================
=====>>> CUBE model <<<===== ML-estimates
=========================================================================
m=9  Size=500  Iterations=62  Maxiter=1000  Tol=1E-06
-------------------------------------------------------------------------
Uncertainty
     Estimates  StdErr    Wald  p-value
pi       0.577  0.0633   9.108   0.0000
```

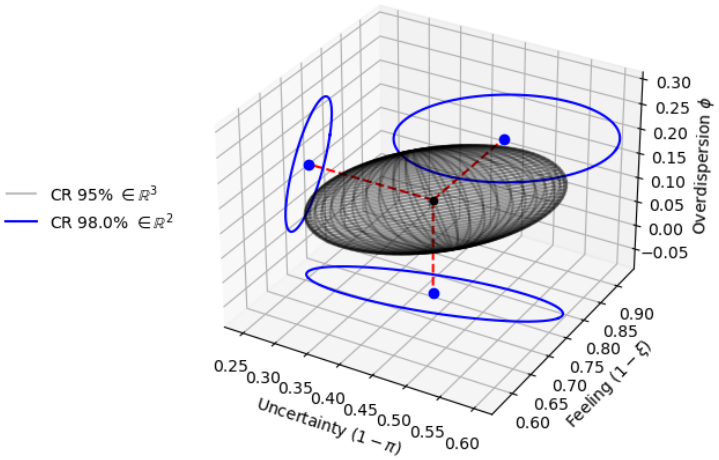
(continues on next page)

(continued from previous page)

```
------------------------------------------------------------------------
Feeling
     Estimates  StdErr    Wald  p-value
xi       0.251  0.0217  11.560   0.0000
------------------------------------------------------------------------
Overdispersion
     Estimates  StdErr    Wald  p-value
phi      0.111  0.0402   2.754   0.0059
========================================================================
Dissimilarity = 0.0426
Loglik(sat)   = -1037.855
Loglik(MOD)   = -1041.100
Loglik(uni)   = -1098.612
Mean-loglik   = -2.082
Deviance      = 6.491
------------------------------------------------------------------------
AIC = 2088.20
BIC = 2100.84
========================================================================
Elapsed time=0.07919 seconds =====>>> Fri Aug 16 12:18:49 2024
========================================================================
```


CUBE model $n = 500$
Estim($\pi = 0.577$, $\xi = 0.251$, $\phi = 0.111$) Dissim(est,obs)=0.043
Gener($\pi = 0.700$, $\xi = 0.300$, $\phi = 0.150$)
estimated
observed
generating
Probability
Ordinal


CR 95% $\in \mathbb{R}^3$
CR 98.0% $\in \mathbb{R}^2$
Uncertainty $(1-\pi)$
Feeling $(1-\xi)$
Overdispersion ϕ


CR 95% $\in \mathbb{R}^3$
CR 98.0% $\in \mathbb{R}^2$
Uncertainty $(1-\pi)$
Feeling $(1-\xi)$
Overdispersion ϕ

2.6.3 With covariates

$$\Pr(R_i = r|\boldsymbol{\theta}, \boldsymbol{y}_i, \boldsymbol{w}_i, \boldsymbol{z}_i) = \pi_i b_r(p_i) + (1 - \pi_i)/m$$

$$p_i \sim \text{Beta}(\alpha_i, \beta_i)$$

$$\begin{cases} \xi_i = \dfrac{\beta_i}{\alpha_i + \beta_i} \\ \phi_i = \dfrac{1}{\alpha_i + \beta_i} \end{cases}$$

$$\begin{cases} \pi_i = \dfrac{1}{1 + \exp\{-\boldsymbol{y}_i\boldsymbol{\beta}\}} \\ \xi_i = \dfrac{1}{1 + \exp\{-\boldsymbol{w}_i\boldsymbol{\gamma}\}} \\ \phi_i = \exp\{\boldsymbol{z}_i\boldsymbol{\alpha}\} \end{cases}$$

Currently, two CUBE models have been defined and implemented: for the *feeling* only and for all components. Nevertheless, the symbol 1 can always be used in the formula for different combinations of covariates.

In this example, we'll draw a sample with covariates for *feeling* only and then will estimate the parameters given the observed sample.

Listing 17: Script

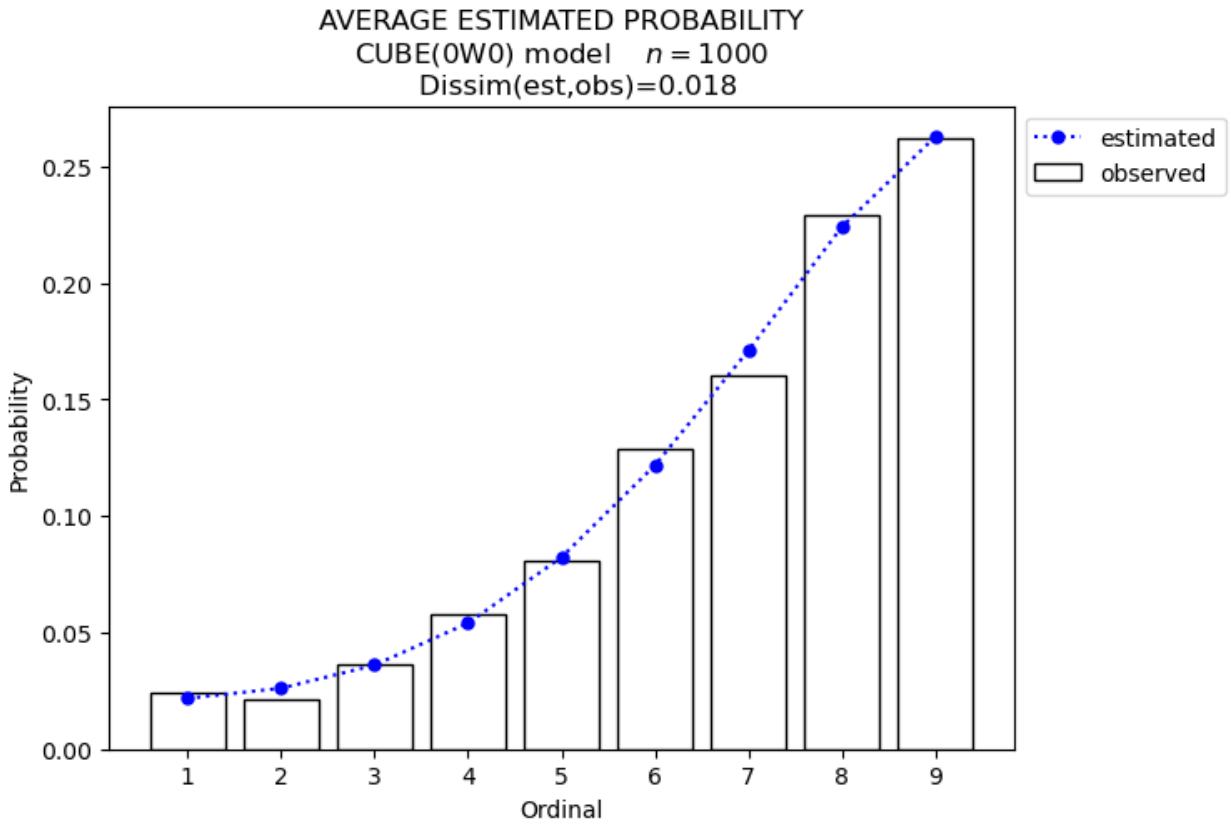
```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from cubmods.general import expit, logit
from cubmods.gem import draw, estimate

# Draw a random sample
n = 1000
np.random.seed(76)
W = np.random.randint(1, 10, n)
df = pd.DataFrame({
    "W": W,
})
drawn = draw(
    formula="fee ~ 0 | W | 0",
    model="cube",
    df=df,
    m=9,
    pi=.8,
    gamma=[logit(.3), -.1],
    phi=.12,
)

# MLE estimation
fit = estimate(
    formula="fee ~ 0 | W | 0",
    model="cube",
    df=drawn.df,
)
# Print MLE summary
```

(continues on next page)

(continued from previous page)

```
print(fit.summary())
# plot the results
fit.plot()
plt.show()
```

```
warnings.warn("No m given, max(ordinal) has been taken")
================================================================================
=====>>> CUBE(0W0) model <<<===== ML-estimates
================================================================================
m=9  Size=1000
--------------------------------------------------------------------------------
Uncertainty
          Estimates  StdErr    Wald  p-value
pi            0.815  0.0343  23.733   0.0000
--------------------------------------------------------------------------------
Feeling
          Estimates  StdErr    Wald  p-value
constant     -0.770  0.1012  -7.612   0.0000
W            -0.116  0.0191  -6.052   0.0000
--------------------------------------------------------------------------------
Overdisperson
          Estimates  StdErr    Wald  p-value
phi           0.150  0.0260   5.779   0.0000
================================================================================
Dissimilarity = 0.0183
Loglik(MOD)   = -1886.654
Loglik(uni)   = -2197.225
Mean-loglik   = -1.887
--------------------------------------------------------------------------------
AIC = 3781.31
BIC = 3800.94
================================================================================
Elapsed time=2.30903 seconds =====>>> Fri Aug 16 12:31:10 2024
================================================================================
```



Notice that the same results can be achieved using a CUBE model with covariates for all components and passing the symbol 1 to the *uncertainty* and *overdispersion* components.

Listing 18: Script

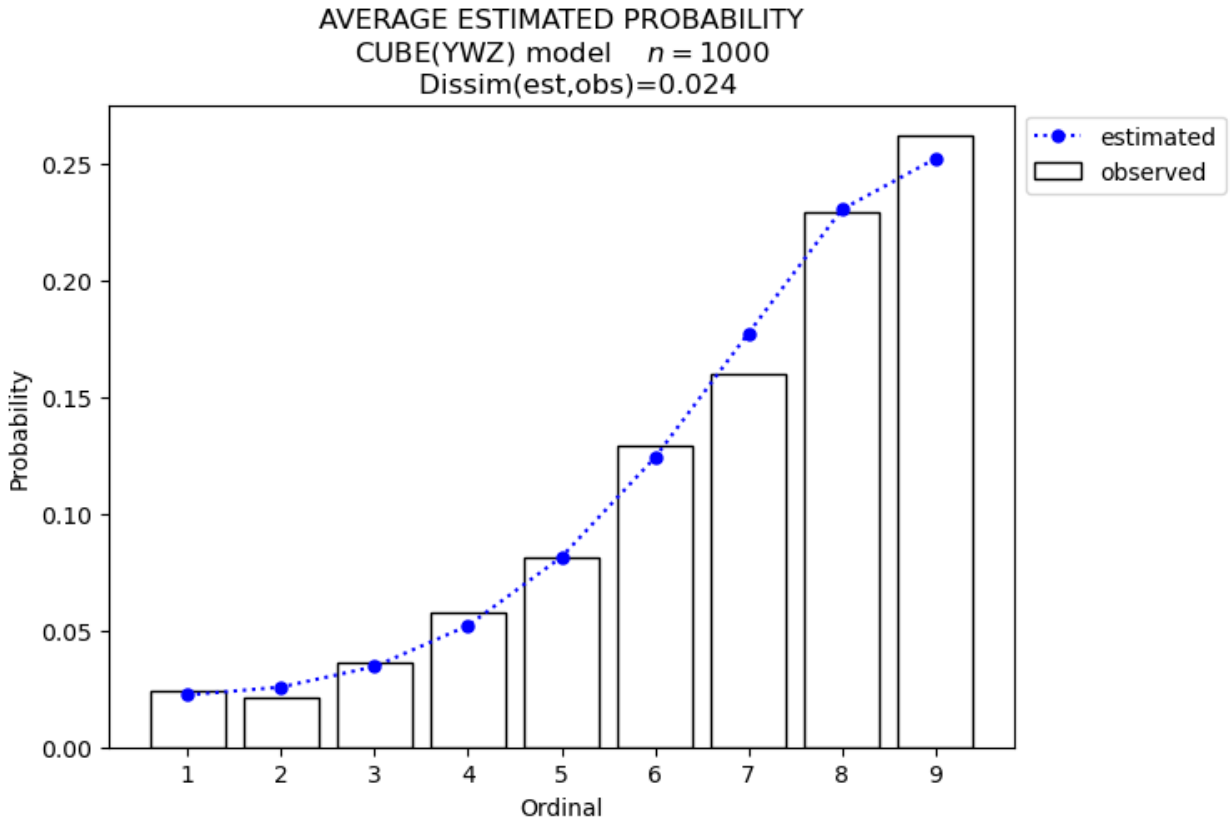
```
# MLE estimation
fit = estimate(
    formula="fee ~ 1 | W | 1",
    model="cube",
    df=drawn.df,
)
# Print MLE summary
print(fit.summary())
# plot the results
fit.plot()
plt.show()
```

```
warnings.warn("No m given, max(ordinal) has been taken")
================================================================================
=====>>> CUBE(YWZ) model <<<===== ML-estimates
================================================================================
m=9  Size=1000  Iterations=29  Maxiter=1000  Tol=1E-02
--------------------------------------------------------------------------------
Uncertainty
          Estimates  StdErr     Wald  p-value
constant      1.423  0.2183    6.518   0.0000
```

(continues on next page)

(continued from previous page)

```
------------------------------------------------------------------------
Feeling
          Estimates  StdErr     Wald  p-value
constant     -0.778  0.1018   -7.639   0.0000
W            -0.117  0.0193   -6.074   0.0000
------------------------------------------------------------------------
Overdispersion
          Estimates  StdErr     Wald  p-value
constant     -1.930  0.1756  -10.989   0.0000
========================================================================
Dissimilarity = 0.0239
Loglik(MOD)   = -1886.690
Loglik(uni)   = -2197.225
Mean-loglik   = -1.887
------------------------------------------------------------------------
AIC = 3781.38
BIC = 3801.01
========================================================================
Elapsed time=50.02969 seconds =====>>> Fri Aug 16 12:33:36 2024
========================================================================
```



In fact:

Listing 19: Script

```
est_pi = expit(fit.estimates[0])
est_ph = np.exp(fit.estimates[3])
est_pi_se = expit(fit.estimates[0]+fit.stderrs[0]) - est_pi
est_ph_se = np.exp(fit.estimates[3]+fit.stderrs[3]) - est_ph
print(
    "     estimates  stderr\n"
    f"pi       {est_pi:.4f}  {est_pi_se:.4f}"
    "\n"
    f"phi      {est_ph:.4f}  {est_ph_se:.4f}"
)
```

```
     estimates  stderr
pi      0.8058  0.0319
phi     0.1451  0.0279
```

2.7 IHG family

Family of the class IHG (Inverse Hyper Geometric). See the references for details.

2.7.1 References

1. A D'Elia, Domenico Piccolo, and others. The moment estimator for the ihg distribution. In *S. Co. Modelli complessi e metodi computazionali intensivi per la stima e la previsione*, pages 245–250. CLEUP, 2005.
2. Angela D'Elia. Modelling ranks using the inverse hypergeometric distribution. *Statistical modelling*, 3(1):65–78, 2003.
3. Domenico Piccolo and Rosaria Simone. The class of cub models: statistical foundations, inferential issues and empirical evidence. *Statistical Methods & Applications*, 28:389–435, 2019.

2.7.2 Without covariates

In this example, we'll draw a sample from an IHG model and the estimate the parameter from the observed sample.

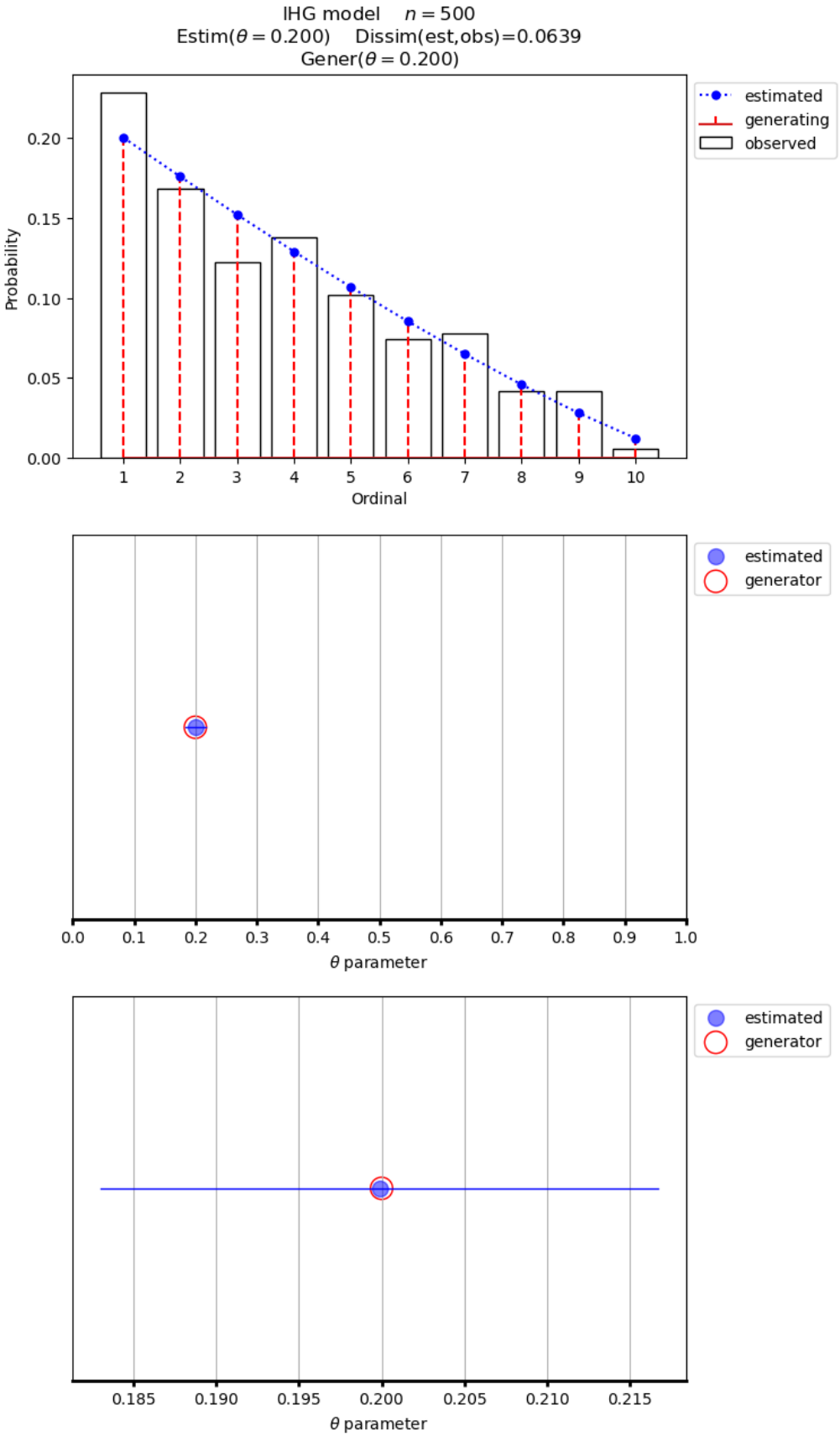
```
# import libraries
import matplotlib.pyplot as plt
from cubmods.gem import draw, estimate

# draw a sample
drawn = draw(
    formula="ord ~ 0",
    model="ihg",
    m=10, theta=.2,
    n=500, seed=42)

# inferential method on drawn sample
```

(continues on next page)

(continued from previous page)

```
fit = estimate(
    df=drawn.df,
    formula="ord ~ 0",
    model="ihg",
    gen_pars={
        "theta": drawn.pars[0],
    }
)
# print the summary of MLE
print(fit.summary())
# show the plot of MLE
fit.plot()
plt.show()
```

```
warnings.warn("No m given, max(ordinal) has been taken")
========================================================================
=====>>> IHG model <<<===== ML-estimates
========================================================================
m=10  Size=500
------------------------------------------------------------------------
Theta
       Estimates  StdErr    Wald  p-value
theta      0.200  0.0086  23.292   0.0000
========================================================================
Dissimilarity = 0.0639
Loglik(sat)   = -1044.100
Loglik(MOD)   = -1050.513
Loglik(uni)   = -1151.293
Mean-loglik   = -2.101
Deviance      = 12.824
------------------------------------------------------------------------
AIC = 2103.03
BIC = 2107.24
========================================================================
Elapsed time=0.00464 seconds =====>>> Fri Aug 16 12:47:55 2024
========================================================================
```


IHG model $n = 500$
Estim($\theta = 0.200$) Dissim(est,obs)=0.0639
Gener($\theta = 0.200$)
estimated
generating
observed
Probability
0.00
0.05
0.10
0.15
0.20
1
2
3
4
5
6
7
8
9
10
Ordinal
estimated
generator
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
θ parameter
estimated
generator
0.185
0.190
0.195
0.200
0.205
0.210
0.215
θ parameter

2.7.3 With covariates

$$\theta_i = \frac{1}{1 + \exp\{-\boldsymbol{\nu}_i \boldsymbol{v}\}}$$

In this example we'll draw a sample from an IHG with covariates and then will estimate the parameters given the observed sample.

Listing 20: Script

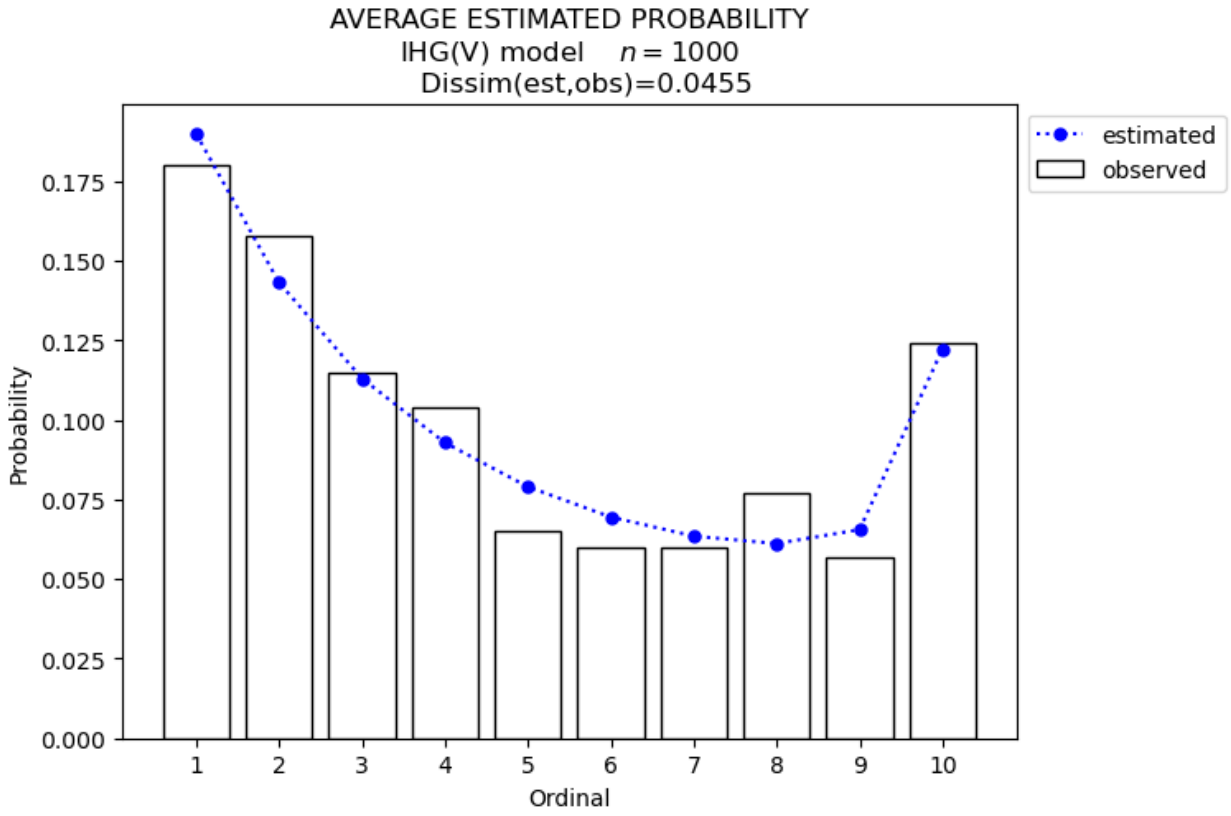
```
# import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from cubmods.gem import draw, estimate
from cubmods.general import logit

# Draw a random sample
n = 1000
np.random.seed(1)
V1 = np.random.random(n)
np.random.seed(42)
V2 = np.random.random(n)
df = pd.DataFrame({
    "V1": V1, "V2": V2
})

# draw a sample
drawn = draw(
    df=df,
    formula="ord ~ V1 + V2",
    model="ihg",
    m=10,
    nu=[logit(.1), -2, 3],
    seed=42)

# inferential method on drawn sample
fit = estimate(
    df=drawn.df,
    formula=drawn.formula,
    model="ihg",
    gen_pars={
        "theta": drawn.pars[0],
    }
)
# print the summary of MLE
print(fit.summary())
# show the plot of MLE
fit.plot()
plt.show()
```

```
warnings.warn("No m given, max(ordinal) has been taken")
======================================================================
=====>>> IHG(V) model <<<===== ML-estimates
```

(continues on next page)

(continued from previous page)

```
=========================================================================
m=10  Size=1000
-------------------------------------------------------------------------
Theta
          Estimates  StdErr     Wald  p-value
constant     -2.368  0.0998  -23.741   0.0000
V1           -1.973  0.1438  -13.721   0.0000
V2            3.230  0.1451   22.261   0.0000
=========================================================================
Dissimilarity = 0.0455
Loglik(MOD)   = -1958.475
Loglik(uni)   = -2302.585
Mean-loglik   = -1.958
-------------------------------------------------------------------------
AIC = 3922.95
BIC = 3937.67
=========================================================================
Elapsed time=1.10664 seconds =====>>> Fri Aug 16 12:53:12 2024
=========================================================================
```



2.8 MULTICUB

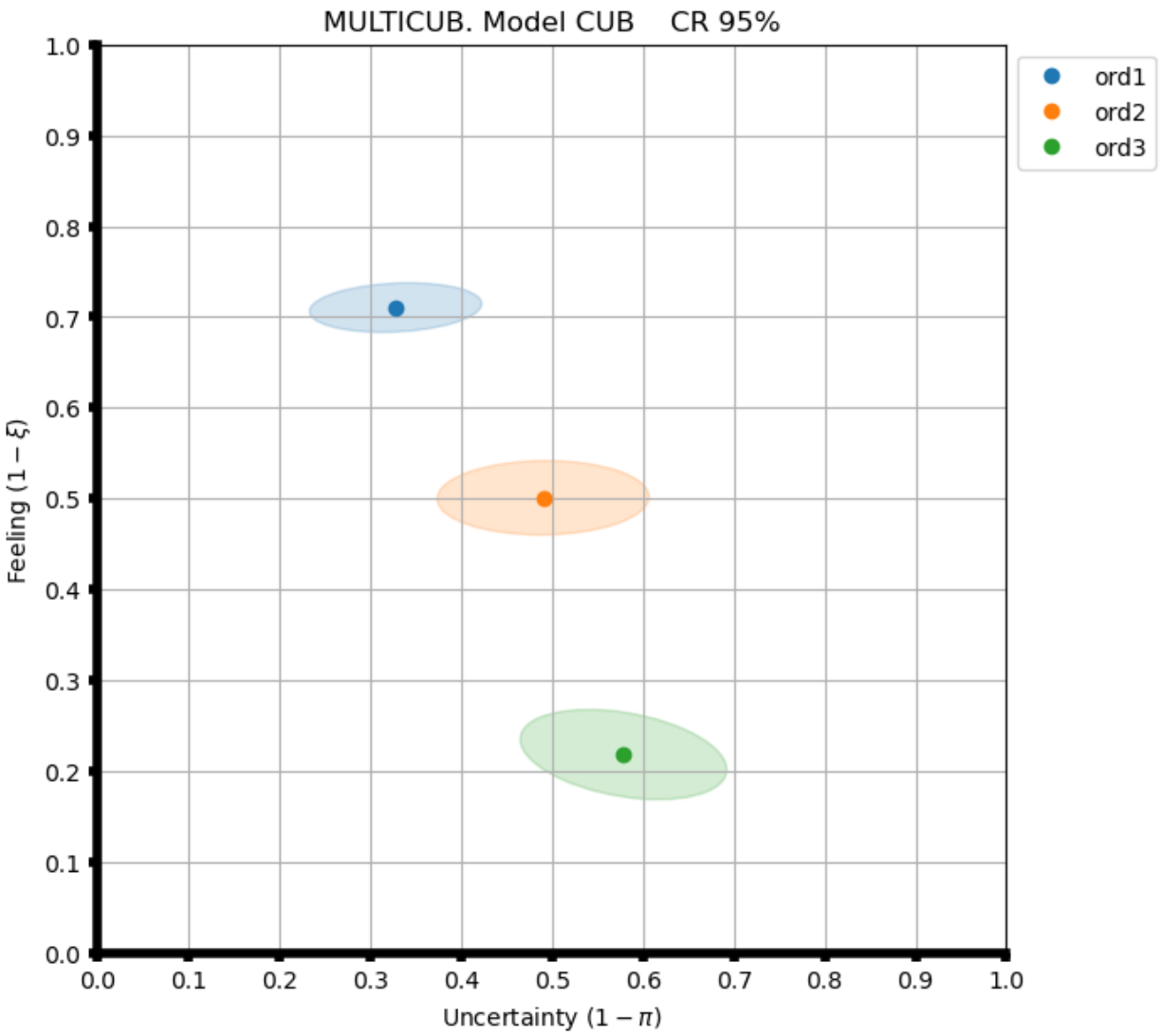
1. Domenico Piccolo and Rosaria Simone. The class of cub models: statistical foundations, inferential issues and empirical evidence. *Statistical Methods & Applications*, 28:389–435, 2019.

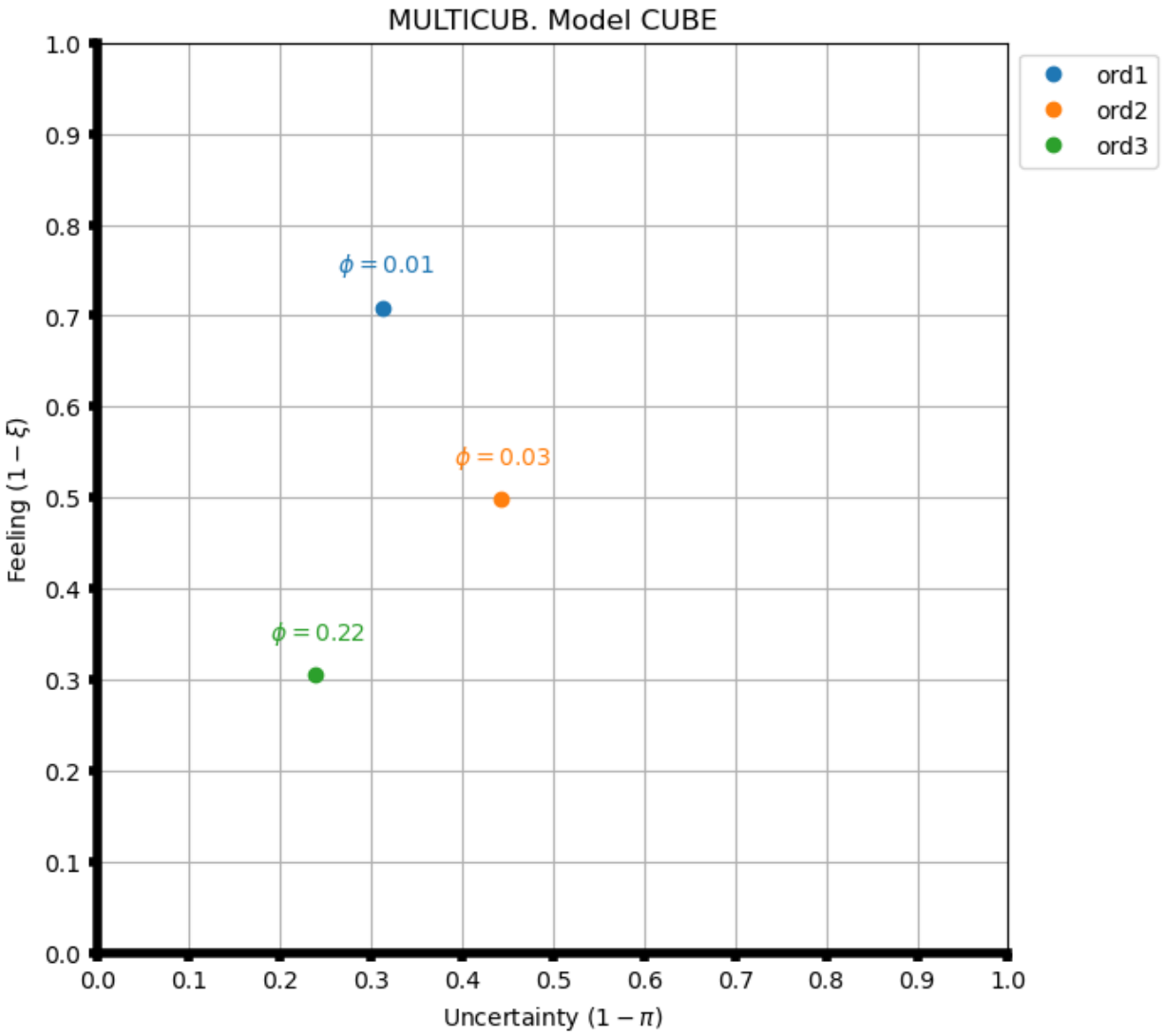
With the **multicub** tool, parameters estimated from multiple observed samples can be shown in a single plot.

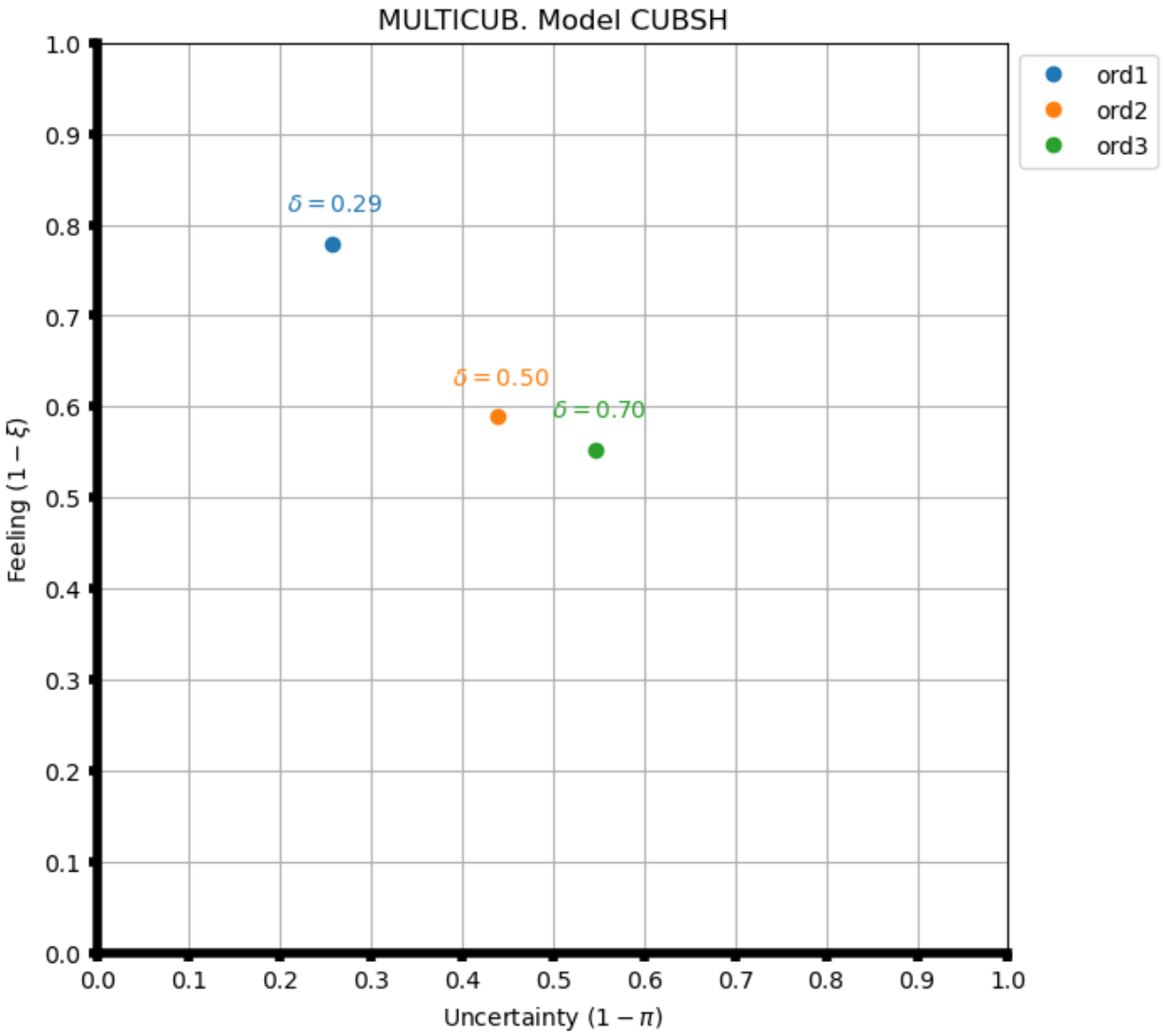
In this example, we'll draw three samples from CUBE models and *manually* add a shelter category. Then we'll use the **multicub** tool for CUB models, CUBE models and CUBSH models.

Listing 21: Script

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from cubmods.gem import draw
from cubmods.multicub import multi

# draw random samples
df = pd.DataFrame()
for i, (pi, xi, phi) in enumerate(
    zip([.9, .8, .7], [.3, .5, .7], [.05, .1, .15])
    ):
    drawn = draw(
        formula="ord ~ 0 | 0 | 0",
        m = 9, model="cube", n=500,
        pi=pi, xi=xi, phi=phi
    )
    # add a shelter category at c=1
    df[f"ord{i+1}"] = np.concatenate((
        drawn.rv, np.repeat(1, 25)
    ))

# MULTI-CUB
multi(
    ords=df, ms=9, model="cub"
)
plt.show()
# MULTI-CUBE
multi(
    ords=df, ms=9, model="cube"
)
plt.show()
# MULTI-CUBSH
multi(
    ords=df, ms=9, model="cub", shs=1
)
plt.show()
```


MULTICUB. Model CUB CR 95%
ord1
ord2
ord3
Feeling $(1-\xi)$
Uncertainty $(1-\pi)$
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0


MULTICUB. Model CUBE
ord1
ord2
ord3
$\phi = 0.01$
$\phi = 0.03$
$\phi = 0.22$
Feeling $(1 - \xi)$
Uncertainty $(1 - \pi)$
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0


MULTICUB. Model CUBSH
ord1
ord2
ord3
$\delta = 0.29$
$\delta = 0.50$
$\delta = 0.70$
Feeling $(1 - \xi)$
Uncertainty $(1 - \pi)$
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

CHAPTER

THREE

CUBMODS

3.1 cubmods package

3.1.1 Submodules

3.1.2 cubmods.cub module

CUB models in Python. Module for CUB (Combination of Uniform and Binomial).

Description

This module contains methods and classes for CUB model family. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

$$\Pr(R = r|\boldsymbol{\theta}) = \pi \binom{m-1}{r-1} (1-\xi)^{r-1} \xi^{m-r} + \frac{1-\pi}{m}$$

Manual and Examples

- Manual https://github.com/maxdevblock/cubmods/blob/main/Manual/02_cub_family.md

References

1. Stefania Capecchi and Domenico Piccolo. Dealing with heterogeneity in ordinal responses. *Quality & Quantity*, 51:2375–2393, 2017.
2. Giovanni Cerulli, R Simone, Francesca Di Iorio, D Piccolo, CF Baum, and others. The stata module for cub models for rating data analysis. In *London Stata Conference 2021*, number 16. Stata Users Group, 2021.
3. A D'Elia, Domenico Piccolo, and others. The moment estimator for the ihg distribution. In *S. Co. Modelli complessi e metodi computazionali intensivi per la stima e la previsione*, pages 245–250. CLEUP, 2005.
4. Angela D'Elia and Domenico Piccolo. A mixture model for preferences data analysis. *Computational Statistics & Data Analysis*, 49(3):917–934, 2005.
5. Angela D'Elia. Modelling ranks using the inverse hypergeometric distribution. *Statistical modelling*, 3(1):65–78, 2003.
6. Nicola Fusco, Paolo Marcellini, and Carlo Sbordone. *Analisi matematica due*. Liguori, 1996.

7. Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. URL: https://doi.org/10.1038/s41586-020-2649-2, doi:10.1038/s41586-020-2649-2.
8. Maria Iannario. Modelling shelter choices in a class of mixture models for ordinal responses. *Statistical Methods & Applications*, 21:1–22, 2012.
9. Maria Iannario. Preliminary estimators for a mixture model of ordinal data. *Advances in Data Analysis and Classification*, 6:163–184, 2012.
10. Maria Iannario. Modelling uncertainty and overdispersion in ordinal data. *Communications in Statistics-Theory and Methods*, 43(4):771–786, 2014.
11. Maria Iannario. Detecting latent components in ordinal data with overdispersion by means of a mixture distribution. *Quality & Quantity*, 49:977–987, 2015.
12. Maria Iannario and Domenico Piccolo. A program in r for cub models inference. *Available via Internet, URL http://www. dipstat. unina. it/CUBmodels/, Version*, 2009.
13. Maria Iannario and Domenico Piccolo. A new statistical model for the analysis of customer satisfaction. *Quality Technology & Quantitative Management*, 7(2):149–168, 2010.
14. Maria Iannario, Domenico Piccolo, and others. Inference for cub models: a program in r. *Statistica & Applicazioni*, 12(2):177–204, 2014.
15. Maria Iannario, Domenico Piccolo, and Maintainer Rosaria Simone. Package 'cub'. *CRAN*, 2022.
16. Norman L Johnson, Adrienne W Kemp, and Samuel Kotz. *Univariate discrete distributions*. Volume 444. John Wiley & Sons, 2005.
17. Norman L Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. *Continuous univariate distributions*. Volume 289. John wiley & sons, 1995.
18. Harikrishna Kundariya. R vs python : our developers recommendation on choosing the best one. 2023. URL: https://www.esparkinfo.com/blog/r-vs-python.html (visited on 2023-03-25).
19. Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.
20. H Lucas, G Shmueli, and others. Too big to fail: large samples and the p-value problem. *Inf. Syst. Res.*, 24(4):906–917, 2013.
21. John Mirowsky. Depression and the sense of control: aging vectors, trajectories, and trends. *Journal of Health and Social Behavior*, 54(4):407–425, 2013.
22. Domenico Piccolo and others. On the moments of a mixture of uniform and shifted binomial random variables. *Quaderni di Statistica*, 5(1):85–104, 2003.
23. Domenico Piccolo and others. Observed information matrix for mub models. *Quaderni di Statistica*, 8(1):33–78, 2006.
24. Domenico Piccolo and others. *Statistica per le decisioni*. Il mulino, 2010.
25. Domenico Piccolo and Rosaria Simone. The class of cub models: statistical foundations, inferential issues and empirical evidence. *Statistical Methods & Applications*, 28:389–435, 2019.
26. Massimo Pierini. Modelli della classe cub in python. *Universitas Mercatorum, Rome, IT*, pages 16–20, June 2024. (Bachelor's thesis L-41).
27. W Stephen Pittard and Shuzhao Li. The essential toolbox of data science: python, r, git, and docker. *Computational Methods and Data Analysis for Metabolomics*, pages 265–311, 2020.

28. Carsten Schelp. An alternative way to plot the covariance ellipse. 2018. URL: https://carstenschelp.github.io/2018/09/14/Plot_Confidence_Ellipse_001.html (visited on 2023-12-19).

29. Skipper Seabold and Josef Perktold. Statsmodels: econometric and statistical modeling with python. In *Proceedings of the 9th Python in Science Conference*, volume 57, 10–25080. Austin, TX, 2010.

30. Rosaria Simone, Francesca Di Iorio, and Riccardo Lucchetti. Cub for gretl. *GNU Regression, Econometrics and Time Series Library*, pages 147, 2019.

31. Guido Van Rossum and Fred L Drake Jr. The python language reference. *Python Software Foundation: Wilmington, DE, USA*, 2014.

32. Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, and others. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

33. Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: l-bfgs-b: fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997.

34. Matplotlib. Plot a confidence ellipse of a two-dimensional dataset. 2023. URL: https://matplotlib.org/stable/gallery/statistics/confidence_ellipse.html (visited on 2023-12-19).

35. R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL: https://www.R-project.org/.

36. University of Cambridge. Testing normality including skewness and kurtosis. 2018. URL: https://imaging.mrc-cbu.cam.ac.uk/statswiki/FAQ/Simon (visited on 2023-12-14).

List of TODOs:

- …

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.cub.**CUBresCUB00**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([ci, saveas, figsize])	Main function to plot an object of the Class.
plot_confell([figsize, ci, equal, ...])	Plots the estimated parameter values in the parameter space and the asymptotic confidence ellipse.
plot_ordinal([figsize, kind, ax, saveas])	Plots relative frequencies of observed sample, estimated probability distribution and, if provided, probability distribution of a known model.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*ci=0.95*, *saveas=None*, *figsize=(7, 15)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **ci** (*float*) – level $(1 - \alpha/2)$ for the confidence ellipse
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns

`ax` or a tuple `(fig, ax)`

plot_confell(*figsize=(7, 5)*, *ci=0.95*, *equal=True*, *magnified=False*, *ax=None*, *saveas=None*)

Plots the estimated parameter values in the parameter space and the asymptotic confidence ellipse.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **ci** (*float*) – level $(1 - \alpha/2)$ for the confidence ellipse
- **equal** (*bool*) – if the plot must have equal aspect
- **magnified** (*bool*) – if False the limits will be the entire parameter space, otherwise let matplotlib choose the limits
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None

- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
ax or a tuple (fig, ax)

plot_ordinal(*figsize=(7, 5)*, *kind='bar'*, *ax=None*, *saveas=None*)

Plots relative frequencies of observed sample, estimated probability distribution and, if provided, probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of (length, height) for the figure (useful only if ax is not None)
- **kind** (*str*) – choose a barplot ('bar' default) of a scatterplot ('scatter')
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
ax or a tuple (fig, ax)

cubmods.cub.**cmf**(*m*, *pi*, *xi*)

Cumulative probability of a specified CUB model.

$\Pr(R \geq r|\boldsymbol{\theta}),\ r = 1 \dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ

Returns
an array of the CMF for the specified model

Return type
numpy array

cubmods.cub.**draw**(*m*, *pi*, *xi*, *n*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified CUB model.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ
- **n** (*int*) – number of ordinal responses to be drawn
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns
an instance of CUBsample containing ordinal responses drawn from the specified model

cubmods.cub.**gini**(*m*, *pi*, *xi*)

The Gini index of a specified CUB model.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ

Returns
the Gini index of the model

Return type
float

cubmods.cub.**init_theta**(*f*, *m*)

Preliminary estimators for CUB models without covariates.

Computes preliminary parameter estimates of a CUB model without covariates for given ordinal responses. These preliminary estimators are used within the package code to start the E-M algorithm.

Parameters

- **f** (*array of int*) – array of the absolute frequencies of given ordinal responses
- **m** (*int*) – number of ordinal categories

Returns
a tuple of $(\pi^{(0)}, \xi^{(0)})$

cubmods.cub.**laakso**(*m*, *pi*, *xi*)

The Laakso index of a specified CUB model.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ

Returns
the Laakso index of the model

Return type
float

cubmods.cub.**loglik**(*m*, *pi*, *xi*, *f*)

Compute the log-likelihood function of a CUB model without covariates for a given absolute frequency distribution.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ
- **f** (*array of int*) – array of absolute frequency distribution

Returns
the log-likelihood value

Return type
float

cubmods.cub.**mean**(*m*, *pi*, *xi*)

Expected value of a specified CUB model.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ

Returns
the expected value of the model

Return type
float

cubmods.cub.**mean_diff**(*m*, *pi*, *xi*)

cubmods.cub.**median**(*m*, *pi*, *xi*)

The median of a specified CUB model.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ

Returns
the median of the model

Return type
float

cubmods.cub.**mle**(*sample*, *m*, *df*, *formula*, *gen_pars=None*, *maxiter=500*, *tol=0.0001*)

Main function for CUB models without covariates.

Function to estimate and validate a CUB model without covariates for given ordinal responses.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None
- **maxiter** (*int*) – maximum number of iterations allowed for running the optimization algorithm
- **tol** (*float*) – fixed error tolerance for final estimates

Returns
an instance of `CUBresCUB00` (see the Class for details)

Return type
object

cubmods.cub.**pmf**(*m*, *pi*, *xi*)

Probability distribution of a specified CUB model.

$\Pr(R = r|\boldsymbol{\theta}),\ r = 1 \dots m$

References:

1. Giovanni Cerulli, R Simone, Francesca Di Iorio, D Piccolo, CF Baum, and others. The stata module for cub models for rating data analysis. In *London Stata Conference 2021*, number 16. Stata Users Group, 2021.
2. Maria Iannario, Domenico Piccolo, and Maintainer Rosaria Simone. Package 'cub'. *CRAN*, 2022.
3. Domenico Piccolo and others. On the moments of a mixture of uniform and shifted binomial random variables. *Quaderni di Statistica*, 5(1):85–104, 2003.
4. Domenico Piccolo and others. Observed information matrix for mub models. *Quaderni di Statistica*, 8(1):33–78, 2006.
5. Domenico Piccolo and Rosaria Simone. The class of cub models: statistical foundations, inferential issues and empirical evidence. *Statistical Methods & Applications*, 28:389–435, 2019.
6. Massimo Pierini. Modelli della classe cub in python. *Universitas Mercatorum, Rome, IT*, pages 16–20, June 2024. (Bachelor's thesis L-41).
7. W Stephen Pittard and Shuzhao Li. The essential toolbox of data science: python, r, git, and docker. *Computational Methods and Data Analysis for Metabolomics*, pages 265–311, 2020.
8. Rosaria Simone, Francesca Di Iorio, and Riccardo Lucchetti. Cub for gretl. *GNU Regression, Econometrics and Time Series Library*, pages 147, 2019.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ

Returns
the vector of the probability distribution of a CUB model.

Return type
numpy array

cubmods.cub.**proba**(*m*, *pi*, *xi*, *r*)

Probability $\Pr(R = r|\boldsymbol{\theta})$ of a specified CUB model.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ
- **r** (*int*) – ordinal value (must be $1 \leq r \leq m$)

Returns
the probability $\Pr(R = r|\boldsymbol{\theta})$

Return type
float

cubmods.cub.**skew**(*pi*, *xi*)

Skewness normalized η index

Parameters

- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ

Returns
the skewness of the model

Return type
float

cubmods.cub.**std**(*m*, *pi*, *xi*)

Standard deviation of a specified CUB model.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ

Returns
the standard deviation of the model

Return type
float

cubmods.cub.**var**(*m*, *pi*, *xi*)

Variance of a specified CUB model.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ

Returns
the variance of the model

Return type
float

cubmods.cub.**varcov**(*m*, *pi*, *xi*, *ordinal*)

Compute the variance-covariance matrix of parameter estimates of a CUB model without covariates.

References
Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ
- **ordinal** (*array of int*) – array of ordinal responses

Returns
the variance-covariance matrix of the CUB model

Return type
numpy ndarray

3.1.3 cubmods.cub_0w module

CUB models in Python. Module for CUB (Combination of Uniform and Binomial) with covariates for the feeling component.

Description:

This module contains methods and classes for CUB_0W model family. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

$$\Pr(R = r_i|\boldsymbol{\theta}_i; \boldsymbol{w}_i) = \pi \binom{m-1}{r_i-1}(1-\xi_i)^{r_i-1}\xi_i^{m-r_i} + \frac{1-\pi}{m}$$

$$\xi_i = \frac{1}{1+e^{-\boldsymbol{w}_i\boldsymbol{\gamma}}}$$

Manual and Examples

- Manual https://github.com/maxdevblock/cubmods/blob/main/Manual/02_cub_family.md

References:

- D'Elia A. (2003). Modelling ranks using the inverse hypergeometric distribution, Statistical Modelling: an International Journal, 3, 65–78
- D'Elia A. and Piccolo D. (2005). A mixture model for preferences data analysis, Computational Statistics & Data Analysis}, bold{49, 917–937
- Capecchi S. and Piccolo D. (2017). Dealing with heterogeneity in ordinal responses, Quality and Quantity, 51(5), 2375–2393
- Iannario M. (2014). Modelling Uncertainty and Overdispersion in Ordinal Data, Communications in Statistics - Theory and Methods, 43, 771–786
- Piccolo D. (2015). Inferential issues for CUBE models with covariates, Communications in Statistics. Theory and Methods, 44(23), 771–786.
- Iannario M. (2015). Detecting latent components in ordinal data with overdispersion by means of a mixture distribution, Quality & Quantity, 49, 977–987
- Iannario M. and Piccolo D. (2016a). A comprehensive framework for regression models of ordinal data. Metron, 74(2), 233–252.
- Iannario M. and Piccolo D. (2016b). A generalized framework for modelling ordinal data. Statistical Methods and Applications, 25, 163–189.

List of TODOs:

- ...

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.cub_0w.**CUBresCUB0W**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([saveas, figsize])	Main function to plot an object of the Class.
plot_ordinal([figsize, ax, kind, saveas])	Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*saveas=None*, *figsize=(7, 5)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

plot_ordinal(*figsize=(7, 5)*, *ax=None*, *kind='bar'*, *saveas=None*)

Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

cubmods.cub_0w.**cmf**(*m*, *pi*, *gamma*, *W*)

Average cumulative probability of a specified CUB model with covariates for the feeling component.

$\Pr(R \geq r|\boldsymbol{\theta}_i;\boldsymbol{w}_i),\ r = 1 \dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns
the array of the cumulative probability distribution.

Return type
numpy array

cubmods.cub_0w.**draw**(*m*, *pi*, *gamma*, *W*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified CUB model with covariates for the feeling component.

Parameters

- **m** (*int*) – number of ordinal categories
- **n** (*int*) – number of ordinal responses to be drawn
- **pi** (*float*) – uncertainty parameter π
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **df** (*DataFrame*) – original DataFrame

- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None; it must be $\neq 0$

Returns
an instance of `CUBsample` containing ordinal responses drawn from the specified model

`cubmods.cub_0w.effe01`(*gamma*, *esterno01*, *m*)

Auxiliary function for the log-likelihood estimation of CUB models with covariates for the feeling component.

Compute the opposite of the scalar function that is maximized when running the E-M algorithm for CUB models with covariates for the feeling parameter.

It is called as an argument for `minimize` within CUB function for models with covariates for feeling or for both feeling and uncertainty.

Parameters

- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **esterno01** – a matrix binding together: the vector $\boldsymbol{\tau}$ of the posterior probabilities that each observation has been generated by the first component distribution of the mixture, the ordinal data $\boldsymbol{r}$ and the matrix $\boldsymbol{w}$ of the selected covariates accounting for an intercept term
- **m** (*int*) – number of ordinal categories

Returns
the expected value of the inconplete log-likelihood

Return type
float

`cubmods.cub_0w.init_gamma`(*sample*, *m*, *W*)

Preliminary parameter estimates of a CUB model with covariates for the feeling component.

Compute preliminary parameter estimates for the feeling component of a CUB model fitted to ordinal responses. These estimates are set as initial values for parameters to start the E-M algorithm.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns
an array $\boldsymbol{\gamma}^{(0)}$ of size $\boldsymbol{w} + 1$

Return type
array of float

`cubmods.cub_0w.loglik`(*sample*, *m*, *pi*, *gamma*, *W*)

Log-likelihood function of a CUB model with covariates for the feeling component

Compute the log-likelihood function of a CUB model fitting ordinal data, with covariates for explaining the feeling component.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories

- **pi** (*float*) – uncertainty parameter π
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns
the log-likelihood value

Return type
float

cubmods.cub_0w.**mle**(*sample*, *m*, *W*, *df*, *formula*, *gen_pars=None*, *maxiter=500*, *tol=0.0001*)

Main function for CUB models with covariates for the feeling component.

Function to estimate and validate a CUB model for given ordinal responses, with covariates for explaining the feeling component.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None
- **maxiter** (*int*) – maximum number of iterations allowed for running the optimization algorithm
- **tol** (*float*) – fixed error tolerance for final estimates

Returns
an instance of `CUBresCUB0W` (see the Class for details)

Return type
object

cubmods.cub_0w.**pmf**(*m*, *pi*, *gamma*, *W*)

Average probability distribution of a specified CUB model with covariates for the feeling component.

$\frac{1}{n}\sum_{i=1}^{n}\Pr(R=r|\boldsymbol{\theta}_i;\boldsymbol{w}_i),\ r=1\dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns
the vector of the probability distribution.

Return type
numpy array

cubmods.cub_0w.**pmfi**(*m*, *pi*, *gamma*, *W*)

Probability distribution for each subject of a specified CUB model with covariates for the feeling component.

Auxiliary function of `.draw()`.

$\Pr(R = r|\boldsymbol{\theta}_i; \boldsymbol{w}_i),\ i = 1 \dots n,\ r = 1 \dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns
the matrix of the probability distribution of dimension $n \times r$

Return type
numpy ndarray

cubmods.cub_0w.**prob**(*sample*, *m*, *pi*, *gamma*, *W*)

Probability distribution of a CUB model with covariates for the feeling component given an observed sample

Compute the probability distribution of a CUB model with covariates for the feeling component, given an observed sample.

$\Pr(R = r_i|\boldsymbol{\theta}_i; \boldsymbol{w}_i),\ i = 1 \dots n$

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns
the array of the probability distribution.

Return type
numpy array

cubmods.cub_0w.**varcov**(*sample*, *m*, *pi*, *gamma*, *W*)

Variance-covariance matrix of CUB models with covariates for the feeling component

Compute the variance-covariance matrix of parameter estimates of a CUB model with covariates for the feeling component.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)

- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns
the variance-covariance matrix of the CUB model

Return type
numpy ndarray

3.1.4 cubmods.cub_y0 module

CUB models in Python. Module for CUB (Combination of Uniform and Binomial) with covariates for the uncertainty component.

Description:

This module contains methods and classes for CUB_Y0 model family. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

$$\Pr(R = r_i|\boldsymbol{\theta}_i; \boldsymbol{y}_i) = \pi_i \binom{m-1}{r_i-1}(1-\xi)^{r_i-1}\xi^{m-r_i} + \frac{1-\pi_i}{m}$$

$$\pi_i = \frac{1}{1+e^{-\boldsymbol{y}_i\boldsymbol{\beta}}}$$

Manual and Examples

- Manual https://github.com/maxdevblock/cubmods/blob/main/Manual/02_cub_family.md

References:

- D'Elia A. (2003). Modelling ranks using the inverse hypergeometric distribution, Statistical Modelling: an International Journal, 3, 65–78
- D'Elia A. and Piccolo D. (2005). A mixture model for preferences data analysis, Computational Statistics & Data Analysis}, bold{49, 917–937
- Capecchi S. and Piccolo D. (2017). Dealing with heterogeneity in ordinal responses, Quality and Quantity, 51(5), 2375–2393
- Iannario M. (2014). Modelling Uncertainty and Overdispersion in Ordinal Data, Communications in Statistics - Theory and Methods, 43, 771–786
- Piccolo D. (2015). Inferential issues for CUBE models with covariates, Communications in Statistics. Theory and Methods, 44(23), 771–786.
- Iannario M. (2015). Detecting latent components in ordinal data with overdispersion by means of a mixture distribution, Quality & Quantity, 49, 977–987
- Iannario M. and Piccolo D. (2016a). A comprehensive framework for regression models of ordinal data. Metron, 74(2), 233–252.
- Iannario M. and Piccolo D. (2016b). A generalized framework for modelling ordinal data. Statistical Methods and Applications, 25, 163–189.

List of TODOs:

- ...

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.cub_y0.**CUBresCUBY0**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([saveas, figsize])	Main function to plot an object of the Class.
plot_ordinal([figsize, ax, kind, saveas])	Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*saveas=None*, *figsize=(7, 5)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

plot_ordinal(*figsize=(7, 5)*, *ax=None*, *kind='bar'*, *saveas=None*)

Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

cubmods.cub_y0.**draw**(*m*, *beta*, *xi*, *Y*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified CUB model with covariates for the uncertainty component.

Parameters

- **m** (*int*) – number of ordinal categories
- **n** (*int*) – number of ordinal responses to be drawn
- **xi** (*float*) – uncertainty parameter ξ
- **beta** (*array of float*) – array β of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None; it must be $\neq 0$

Returns
an instance of `CUBsample` containing ordinal responses drawn from the specified model

cubmods.cub_y0.**effe10**(*beta*, *esterno10*)

Auxiliary function for the log-likelihood estimation of CUB models.

Compute the opposite of the scalar function that is maximized when running the E-M algorithm for CUB models with covariates for the uncertainty parameter.

It is called as an argument for optim within CUB function for models with covariates for
uncertainty or for both feeling and uncertainty.

Parameters

- **beta** (*array of float*) – array β of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **esterno10** – A matrix binding together: the matrix y of the selected covariates (accounting for an intercept term) and a vector τ (whose length equals the number of observations) of the posterior probabilities that each observation has been generated by the first component distribution of the mixture

Returns
the expected value of the inconplete log-likelihood

Return type
float

cubmods.cub_y0.**loglik**(*m*, *sample*, *Y*, *beta*, *xi*)

Log-likelihood function of a CUB model with covariates for the uncertainty component

Compute the log-likelihood function of a CUB model fitting ordinal responses with covariates for explaining the uncertainty component.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **xi** (*float*) – uncertainty parameter ξ
- **beta** (*array of float*) – array β of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component

Returns
the log-likelihood value

Return type
float

cubmods.cub_y0.**mle**(*sample*, *m*, *Y*, *df*, *formula*, *gen_pars=None*, *maxiter=500*, *tol=0.0001*)

Main function for CUB models with covariates for the uncertainty component.

Estimate and validate a CUB model for given ordinal responses, with covariates for explaining the uncertainty component.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None
- **maxiter** (*int*) – maximum number of iterations allowed for running the optimization algorithm
- **tol** (*float*) – fixed error tolerance for final estimates

Returns
an instance of `CUBresCUBY0` (see the Class for details)

Return type
object

`cubmods.cub_y0.pmf`(*m*, *beta*, *xi*, *Y*)

Average probability distribution of a specified CUB model with covariates.

$\frac{1}{n}\sum_{i=1}^{n}\Pr(R=r|\boldsymbol{\theta}_i;\boldsymbol{w}_i),\ r=1\dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **xi** (*float*) – feeling parameter ξ
- **beta** (*array of float*) – array $\boldsymbol{\beta}$ of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component

Returns
the vector of the probability distribution.

Return type
numpy array

`cubmods.cub_y0.pmfi`(*m*, *beta*, *xi*, *Y*)

Probability distribution for each subject of a specified CUB model with covariates.

Auxiliary function of `.draw()`.

$\Pr(R=r|\boldsymbol{\theta}_i;\boldsymbol{y}_i),\ i=1\dots n,\ r=1\dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **xi** (*float*) – feeling parameter ξ
- **beta** (*array of float*) – array $\boldsymbol{\beta}$ of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component

Returns
the matrix of the probability distribution of dimension $n \times r$

Return type
numpy ndarray

`cubmods.cub_y0.prob`(*m*, *sample*, *Y*, *beta*, *xi*)

Probability distribution of a CUB model with covariates for the uncertainty component given an observed sample

Compute the probability distribution of a CUB model with covariates for the feeling component, given an observed sample.

$\Pr(R=r_i|\boldsymbol{\theta}_i;\boldsymbol{w}_i),\ i=1\dots n$

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **xi** (*float*) – uncertainty parameter ξ

- **beta** (*array of float*) – array β of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component

Returns
the array of the probability distribution.

Return type
numpy array

`cubmods.cub_y0.varcov`(*m*, *sample*, *Y*, *beta*, *xi*)

Variance-covariance matrix of CUB model with covariates for the uncertainty parameter.

Compute the variance-covariance matrix of parameter estimates of a CUB model with covariates for the uncertainty component.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **xi** (*float*) – uncertainty parameter ξ
- **beta** (*array of float*) – array β of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component

Returns
the variance-covariance matrix of the CUB model

Return type
numpy ndarray

3.1.5 cubmods.cub_yw module

CUB models in Python. Module for CUB (Combination of Uniform and Binomial) with covariates for both feeling and uncertainty.

Description:

This module contains methods and classes for CUB_YW model family. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

$$\Pr(R = r_i|\boldsymbol{\theta}_i;\boldsymbol{y}_i;\boldsymbol{w}_i) = \pi_i \binom{m-1}{r_i-1}(1-\xi_i)^{r_i-1}\xi_i^{m-r_i} + \frac{1-\pi_i}{m}$$

$$\xi_i = \frac{1}{1+e^{-\boldsymbol{w}_i\boldsymbol{\gamma}}}$$

$$\pi_i = \frac{1}{1+e^{-\boldsymbol{y}_i\boldsymbol{\beta}}}$$

Manual and Examples

- Manual https://github.com/maxdevblock/cubmods/blob/main/Manual/02_cub_family.md

References:

- D'Elia A. (2003). Modelling ranks using the inverse hypergeometric distribution, Statistical Modelling: an International Journal, 3, 65–78
- D'Elia A. and Piccolo D. (2005). A mixture model for preferences data analysis, Computational Statistics & Data Analysis}, bold{49, 917–937
- Capecchi S. and Piccolo D. (2017). Dealing with heterogeneity in ordinal responses, Quality and Quantity, 51(5), 2375–2393
- Iannario M. (2014). Modelling Uncertainty and Overdispersion in Ordinal Data, Communications in Statistics - Theory and Methods, 43, 771–786
- Piccolo D. (2015). Inferential issues for CUBE models with covariates, Communications in Statistics. Theory and Methods, 44(23), 771–786.
- Iannario M. (2015). Detecting latent components in ordinal data with overdispersion by means of a mixture distribution, Quality & Quantity, 49, 977–987
- Iannario M. and Piccolo D. (2016a). A comprehensive framework for regression models of ordinal data. Metron, 74(2), 233–252.
- Iannario M. and Piccolo D. (2016b). A generalized framework for modelling ordinal data. Statistical Methods and Applications, 25, 163–189.

List of TODOs:

- …

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.cub_yw.**CUBresCUBYW**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

"Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([saveas, figsize])	Main function to plot an object of the Class.
plot_ordinal([figsize, ax, kind, saveas])	Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*saveas=None*, *figsize=(7, 5)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns

`ax` or a tuple `(fig, ax)`

plot_ordinal(*figsize=(7, 5)*, *ax=None*, *kind='bar'*, *saveas=None*)

Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns

`ax` or a tuple `(fig, ax)`

cubmods.cub_yw.**draw**(*m*, *beta*, *gamma*, *Y*, *W*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified CUB model with covariates for both feeling and uncertainty.

Parameters

- **n** (*int*) – number of ordinal responses to be drawn
- **m** (*int*) – number of ordinal categories
- **beta** (*array of float*) – array β of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array γ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used

Returns

an instance of `CUBsample` containing ordinal responses drawn from the specified model

cubmods.cub_yw.**loglik**(*m*, *sample*, *Y*, *W*, *beta*, *gamma*)

Log-likelihood function of a CUB model with covariates for both feeling and uncertainty.

Compute the log-likelihood function of a CUB model fitting ordinal data with covariates for explaining both the feeling and the uncertainty components.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **beta** (*array of float*) – array β of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array γ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns

the log-likelihood value

Return type

float

cubmods.cub_yw.**mle**(*sample*, *m*, *Y*, *W*, *df*, *formula*, *gen_pars=None*, *maxiter=500*, *tol=0.0001*)

Main function for CUB models with covariates for both the uncertainty and the feeling components.

Estimate and validate a CUB model for given ordinal responses, with covariates for explaining both the feeling and the uncertainty components by means of logistic transform.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component

- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None
- **maxiter** (*int*) – maximum number of iterations allowed for running the optimization algorithm
- **tol** (*float*) – fixed error tolerance for final estimates

Returns
an instance of `CUBresCUBYW` (see the Class for details)

Return type
object

`cubmods.cub_yw.pmf`(*m*, *beta*, *gamma*, *Y*, *W*)

Average probability distribution of a specified CUB model with covariates for both feeling and uncertainty.

$\frac{1}{n}\sum_{i=1}^{n}\Pr(R=r|\boldsymbol{\theta}_i;\boldsymbol{w}_i;\boldsymbol{y}_i),\ r=1\dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **beta** (*array of float*) – array $\boldsymbol{\beta}$ of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns
the vector of the probability distribution.

Return type
numpy array

`cubmods.cub_yw.pmfi`(*m*, *beta*, *gamma*, *Y*, *W*)

Probability distribution for each subject of a specified CUB model with covariates for both feeling and uncertainty.

Auxiliary function of `.draw()`.

$\Pr(R=r|\boldsymbol{\theta}_i;\boldsymbol{y}_i;\boldsymbol{w}_i),\ i=1\dots n,\ r=1\dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **beta** (*array of float*) – array $\boldsymbol{\beta}$ of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns
the matrix of the probability distribution of dimension $n \times r$

Return type
numpy ndarray

cubmods.cub_yw.**prob**(*m*, *sample*, *Y*, *W*, *beta*, *gamma*)

Probability distribution of a CUB model with covariates for both feeling and uncertainty.

Compute the probability distribution of a CUB model with covariates for both the feeling and the uncertainty components.

$\Pr(R = r_i|\boldsymbol{\theta}_i; \boldsymbol{w}_i; \boldsymbol{y}_i),\ i = 1 \dots n$

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **beta** (*array of float*) – array $\boldsymbol{\beta}$ of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns
the array of the probability distribution.

Return type
numpy array

cubmods.cub_yw.**varcov**(*m*, *sample*, *Y*, *W*, *beta*, *gamma*)

Variance-covariance matrix of a CUB model with covariates for both uncertainty and feeling.

Compute the variance-covariance matrix of parameter estimates of a CUB model with covariates for both the uncertainty and the feeling components.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **beta** (*array of float*) – array $\boldsymbol{\beta}$ of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns
the variance-covariance matrix of the CUB model

Return type
numpy ndarray

3.1.6 cubmods.cube module

CUB models in Python. Module for CUBE (Combination of Uniform and Beta-Binomial).

Description:

This module contains methods and classes for CUBE model family. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

$$\Pr(R = r|\boldsymbol{\theta}) = \pi \text{Beta}(\xi, \phi) + \frac{1-\pi}{m}$$

$$\xi = \frac{\beta}{\alpha + \beta}$$

$$\phi = \frac{1}{\alpha + \beta}$$

Manual and Examples

- Manual https://github.com/maxdevblock/cubmods/blob/main/Manual/03_cube_family.md

References:

- D'Elia A. (2003). Modelling ranks using the inverse hypergeometric distribution, Statistical Modelling: an International Journal, 3, 65–78
- D'Elia A. and Piccolo D. (2005). A mixture model for preferences data analysis, Computational Statistics & Data Analysis}, bold{49, 917–937
- Capecchi S. and Piccolo D. (2017). Dealing with heterogeneity in ordinal responses, Quality and Quantity, 51(5), 2375–2393
- Iannario M. (2014). Modelling Uncertainty and Overdispersion in Ordinal Data, Communications in Statistics - Theory and Methods, 43, 771–786
- Piccolo D. (2015). Inferential issues for CUBE models with covariates, Communications in Statistics. Theory and Methods, 44(23), 771–786.
- Iannario M. (2015). Detecting latent components in ordinal data with overdispersion by means of a mixture distribution, Quality & Quantity, 49, 977–987
- Iannario M. and Piccolo D. (2016a). A comprehensive framework for regression models of ordinal data. Metron, 74(2), 233–252.
- Iannario M. and Piccolo D. (2016b). A generalized framework for modelling ordinal data. Statistical Methods and Applications, 25, 163–189.

List of TODOs:

- TODO: adjust 3d plots legend

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.cube.**CUBresCUBE**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([ci, saveas, confell, test3, figsize])	Main function to plot an object of the Class.
plot3d(ax[, ci, magnified])	Plots the estimated parameter values in the parameter space and the asymptotic confidence ellipsoid with its projections.
plot_ordinal([figsize, ax, kind, saveas])	Plots relative frequencies of observed sample, estimated probability distribution and, if provided, probability distribution of a known model.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*ci=0.95*, *saveas=None*, *confell=False*, *test3=True*, *figsize=(7, 15)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **ci** (*float*) – level $(1 - \alpha/2)$ for the confidence ellipsoid
- **confell** (*bool*) – **DEPRECATED**, defaults to False
- **test3** (*bool*) – **DEPRECATED**, defaults to True
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

plot3d(*ax*, *ci=0.95*, *magnified=False*)

Plots the estimated parameter values in the parameter space and the asymptotic confidence ellipsoid with its projections.

Parameters

- **ci** (*float*) – level $(1 - \alpha/2)$ for the confidence ellipsoid
- **magnified** (*bool*) – if False the limits will be the entire parameter space, otherwise let matplotlib choose the limits
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None

plot_ordinal(*figsize=(7, 5)*, *ax=None*, *kind='bar'*, *saveas=None*)

Plots relative frequencies of observed sample, estimated probability distribution and, if provided, probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

`cubmods.cube.`**betar**(*m*, *xi*, *phi*)

Beta-Binomial distribution.

Return the Beta-Binomial distribution with given parameters.

Parameters

- **m** (*int*) – number of ordinal categories
- **xi** (*float*) – feeling parameter ξ
- **phi** (*float*) – overdispersion parameter ϕ

Returns
array of length m of the Beta-Binomial distribution.

Return type
numpy array

cubmods.cube.**cmf**(*m*, *pi*, *xi*, *phi*)

Cumulative probability of a specified CUBE model.

$\Pr(R \geq r|\boldsymbol{\theta}),\ r = 1 \dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ
- **phi** (*float*) – overdispersion parameter ϕ

Returns
array of length m of the cumulative probability of a CUBE model without covariates.

Return type
numpy array

cubmods.cube.**draw**(*m*, *pi*, *xi*, *phi*, *n*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified CUBE model.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ
- **phi** (*float*) – overdispersion parameter ϕ
- **n** (*int*) – number of ordinal responses to be drawn
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns
an instance of `CUBsample` containing ordinal responses drawn from the specified model

cubmods.cube.**effecube**(*params*, *tau*, *f*, *m*)

Auxiliary function for the log-likelihood estimation of CUBE models without covariates.

Define the opposite of the scalar function that is maximized when running the E-M algorithm for CUBE models without covariates.

Parameters

- **params** (*array of float*) – array of initial estimates for the feeling and the overdispersion parameters
- **tau** (*array*) – a column vector of length m containing the posterior probabilities that each observed category has been generated by the first component distribution of the mixture
- **f** (*array*) – array of the absolute frequencies of the observations
- **m** (*int*) – number of ordinal categories

Returns
the expected value of the inconplete log-likelihood

Return type
float

cubmods.cube.**init_theta**(*sample*, *m*)

Naive estimates for CUBE models without covariates.

Compute naive parameter estimates of a CUBE model without covariates for given ordinal responses. These preliminary estimators are used within the package code to start the E-M algorithm.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories

Returns
a tuple of $(\pi^{(0)}, \xi^{(0)}, \phi^{(0)})$

Return type
tuple of float

cubmods.cube.**loglik**(*m*, *pi*, *xi*, *phi*, *f*)

Log-likelihood function of a CUBE model without covariates.

Compute the log-likelihood function of a CUBE model without covariates fitting the given absolute frequency distribution.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ
- **phi** (*float*) – overdispersion parameter ϕ
- **f** (*array of int*) – array of absolute frequency distribution

Returns
the log-likelihood value

Return type
float

cubmods.cube.**mean**(*m*, *pi*, *xi*)

Mean of a CUBE model.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ

Returns
the expected value of the model

Return type
float

cubmods.cube.**mle**(*sample*, *m*, *df*, *formula*, *gen_pars=None*, *maxiter=1000*, *tol=1e-06*)

Main function for CUBE models without covariates.

Estimate and validate a CUBE model without covariates.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None
- **maxiter** (*int*) – maximum number of iterations allowed for running the optimization algorithm
- **tol** (*float*) – fixed error tolerance for final estimates

Returns
an instance of CUBresCUBE (see the Class for details)

Return type
object

cubmods.cube.**pmf**(*m*, *pi*, *xi*, *phi*)

Probability distribution of a specified CUBE model.

$\Pr(R = r|\boldsymbol{\theta}),\ r = 1 \dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ
- **phi** (*float*) – overdispersion parameter ϕ

Returns
array of length m of the distribution of a CUBE model without covariates.

Return type
numpy array

cubmods.cube.**proba**(*m*, *pi*, *xi*, *phi*, *r*)

Probability $\Pr(R = r|\boldsymbol{\theta})$ of a CUBE model without covariates.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ
- **phi** (*float*) – overdispersion parameter ϕ
- **r** (*int*) – ordinal response

Returns
the probability $\Pr(R = r|\boldsymbol{\theta})$ of a CUBE model without covariates.

Return type
numpy array

cubmods.cube.**var**(*m*, *pi*, *xi*, *phi*)

Variance of a CUBE model.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ
- **phi** (*float*) – overdispersion parameter ϕ

Returns
the variance of the model

Return type
float

cubmods.cube.**varcov**(*m*, *pi*, *xi*, *phi*, *sample*)

Variance-covariance matrix for CUBE models based on the observed information matrix.

Compute the variance-covariance matrix of parameter estimates for a CUBE model without covariates as the inverse of the observed information matrix.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **xi** (*float*) – feeling parameter ξ
- **phi** (*float*) – overdispersion parameter ϕ
- **sample** (*array of int*) – array of ordinal responses

Returns
the variance-covariance matrix of the CUBE model

Return type
numpy ndarray

3.1.7 cubmods.cube_0w0 module

CUB models in Python. Module for CUBE (Combination of Uniform and Beta-Binomial) with covariates for the feeling component.

Description:

This module contains methods and classes for CUBE_0W0 model family. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

$$\Pr(R = r|\boldsymbol{\theta}) = \pi \text{Beta}(\xi, \phi) + \frac{1-\pi}{m}$$

$$\xi = \frac{\beta}{\alpha+\beta} = \frac{1}{1+e^{-\boldsymbol{w}_i\boldsymbol{\gamma}}}$$

$$\phi = \frac{1}{\alpha+\beta}$$

Manual and Examples

- Manual https://github.com/maxdevblock/cubmods/blob/main/Manual/03_cube_family.md

References:

- D'Elia A. (2003). Modelling ranks using the inverse hypergeometric distribution, Statistical Modelling: an International Journal, 3, 65–78
- D'Elia A. and Piccolo D. (2005). A mixture model for preferences data analysis, Computational Statistics & Data Analysis}, bold{49, 917–937
- Capecchi S. and Piccolo D. (2017). Dealing with heterogeneity in ordinal responses, Quality and Quantity, 51(5), 2375–2393
- Iannario M. (2014). Modelling Uncertainty and Overdispersion in Ordinal Data, Communications in Statistics - Theory and Methods, 43, 771–786
- Piccolo D. (2015). Inferential issues for CUBE models with covariates, Communications in Statistics. Theory and Methods, 44(23), 771–786.
- Iannario M. (2015). Detecting latent components in ordinal data with overdispersion by means of a mixture distribution, Quality & Quantity, 49, 977–987
- Iannario M. and Piccolo D. (2016a). A comprehensive framework for regression models of ordinal data. Metron, 74(2), 233–252.
- Iannario M. and Piccolo D. (2016b). A generalized framework for modelling ordinal data. Statistical Methods and Applications, 25, 163–189.

List of TODOs:

- Manual and Examples
- Remove unused imports

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.cube_0w0.**CUBresCUBE0W0**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([saveas, figsize])	Main function to plot an object of the Class.
plot_ordinal([figsize, ax, kind, saveas])	Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*saveas=None*, *figsize=(7, 5)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns

`ax` or a tuple `(fig, ax)`

plot_ordinal(*figsize=(7, 5)*, *ax=None*, *kind='bar'*, *saveas=None*)

Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns

`ax` or a tuple `(fig, ax)`

cubmods.cube_0w0.**betabinomialxi**(*m*, *sample*, *xivett*, *phi*)

Beta-Binomial probabilities of ordinal responses, given feeling parameter for each observation.

Compute the Beta-Binomial probabilities of given ordinal responses, with feeling parameter specified for each observation, and with the same overdispersion parameter for all the responses.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array*) – array of ordinal responses. Missing values are not allowed: they should be preliminarily deleted
- **xivett** (*array*) – array of feeling parameters of the Beta-Binomial distribution for given ordinal responses
- **phi** (*float*) – overdispersion parameter ϕ

Returns

array of the same length as ordinal: each entry is the Beta-Binomial probability for the given observation for the corresponding feeling and overdispersion parameters.

Return type

array

cubmods.cube_0w0.**draw**(*m*, *pi*, *gamma*, *phi*, *W*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified CUBE model.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **phi** (*float*) – overdispersion parameter ϕ
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **n** (*int*) – number of ordinal responses to be drawn
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns

an instance of `CUBsample` containing ordinal responses drawn from the specified model

cubmods.cube_0w0.**effe**(*pars*, *sample*, *W*, *m*)

Auxiliary function for the log-likelihood estimation of CUBE models with covariates only for the feeling component.

Compute the opposite of the scalar function that is maximized when running the E-M algorithm for CUBE models with covariates only for the feeling component.

Parameters

- **pars** (*array*) – array of length equal to `W.index.size+3` whose entries are the initial parameters estimates
- **sample** (*array of int*) – array of ordinal responses

- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **m** (*int*) – number of ordinal categories

Returns
negative log-likelihood

Return type
float

cubmods.cube_0w0.**init_theta**(*m*, *sample*, *W*, *maxiter*, *tol*)

Preliminary estimates of parameters for CUBE models with covariates only for feeling.

Compute preliminary parameter estimates of a CUBE model with covariates only for feeling, given ordinal responses. These estimates are set as initial values to start the corresponding E-M algorithm within the package. Preliminary estimates for the uncertainty and the overdispersion parameters are computed by short runs of EM. As to the feeling component, it considers the nested CUB model with covariates and calls code{link{inibestgama}} to derive initial estimates for the coefficients of the selected covariates for feeling.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **maxiter** (*int*) – maximum number of iterations allowed for preliminary iterations
- **tol** (*float*) – fixed error tolerance for final estimates for preliminary iterations

Returns
a tuple of $(\pi^{(0)}, \boldsymbol{\gamma}^{(0)}, \phi^{(0)})$, where $\pi^{(0)}$ is the initial estimate for the uncertainty parameter, $\boldsymbol{\gamma}^{(0)}$ is the vector of initial estimates for the feeling component (including an intercept term in the first entry), and $\phi^{(0)}$ is the initial estimate for the overdispersion parameter.

"rtype": tuple

cubmods.cube_0w0.**loglik**(*m*, *sample*, *W*, *pi*, *gamma*, *phi*)

Log-likelihood function of CUBE model with covariates only for feeling.

Compute the log-likelihood function of a CUBE model for ordinal data with subjects' covariates only for feeling.

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **phi** (*float*) – overdispersion parameter ϕ
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **sample** (*array of int*) – array of ordinal responses

Returns
the log-likelihood value

Return type
float

cubmods.cube_0w0.**mle**(*sample*, *m*, *W*, *df*, *formula*, *gen_pars=None*, *maxiter=1000*, *tol=1e-06*)

Main function for CUBE models with covariates only for feeling

Estimate and validate a CUBE model for ordinal data, with covariates only for explaining the feeling component.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None
- **maxiter** (*int*) – maximum number of iterations allowed for preliminary iterations
- **tol** (*float*) – fixed error tolerance for final estimates for preliminary iterations; the informatio matrix (to compute the variance-covariance matrix) is approximated with `approx_hess()` (see `statsmodels.tools.numdiff` for details)

Returns
an instance of `CUBresCUBE0W0` (see the Class for details)

Return type
object

cubmods.cube_0w0.**pmf**(*m*, *pi*, *gamma*, *phi*, *W*)

Average probability distribution of a specified CUB model with covariates for the feeling component.

$\frac{1}{n}\sum_{i=1}^{n}\Pr(R=r|\boldsymbol{\theta}_i;\boldsymbol{w}_i),\ r=1\dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size`+1 to include an intercept term in the model (first entry)
- **phi** (*float*) – overdispersion parameter ϕ
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns
the array of the average probability distribution

Return type
numpy array

cubmods.cube_0w0.**pmfi**(*m*, *pi*, *gamma*, *phi*, *W*)

Probability distribution for each subject of a specified CUBE model with covariates for feeling only.

Auxiliary function of `.draw()`.

$\Pr(R=r|\boldsymbol{\theta}_i;\boldsymbol{w}_i),\ i=1\dots n,\ r=1\dots m$

Parameters

- **m** (*int*) – number of ordinal categories

- **pi** (*float*) – uncertainty parameter π
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **phi** (*float*) – overdispersion parameter ϕ
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns
the matrix of the probability distribution of dimension $n \times r$

Return type
numpy ndarray

`cubmods.cube_0w0.prob`(*m*, *sample*, *W*, *pi*, *gamma*, *phi*)

Probability distribution of a CUBE model with covariates for feeling.

Compute the probability distribution of a CUB model with covariates for both the feeling and the uncertainty components. Auxiliary function of `.loglik()`

$\Pr(R = r_i|\boldsymbol{\theta}_i; \boldsymbol{w}_i),\ i = 1 \dots n$

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **phi** (*float*) – overdispersion parameter ϕ
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **sample** (*array of int*) – array of ordinal responses

Returns
the array of the probability distribution.

Return type
numpy array

3.1.8 cubmods.cube_ywz module

CUB models in Python. Module for CUBE (Combination of Uniform and Beta-Binomial) with covariates.

Description:

This module contains methods and classes for CUB_YWZ model family. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

$$\Pr(R = r|\boldsymbol{\theta}) = \pi \text{Beta}(\xi, \phi) + \frac{1-\pi}{m}$$

$$\pi = \frac{1}{1 + e^{-\boldsymbol{y}_i\boldsymbol{\beta}}}$$

$$\xi = \frac{\beta}{\alpha + \beta} = \frac{1}{1 + e^{-\boldsymbol{w}_i\boldsymbol{\gamma}}}$$

$$\phi = \frac{1}{\alpha + \beta} = e^{\boldsymbol{z}_i\boldsymbol{\alpha}}$$

Manual and Examples

- Manual https://github.com/maxdevblock/cubmods/blob/main/Manual/04_cube_family.md

References:

- D'Elia A. (2003). Modelling ranks using the inverse hypergeometric distribution, Statistical Modelling: an International Journal, 3, 65–78
- D'Elia A. and Piccolo D. (2005). A mixture model for preferences data analysis, Computational Statistics & Data Analysis}, bold{49, 917–937
- Capecchi S. and Piccolo D. (2017). Dealing with heterogeneity in ordinal responses, Quality and Quantity, 51(5), 2375–2393
- Iannario M. (2014). Modelling Uncertainty and Overdispersion in Ordinal Data, Communications in Statistics - Theory and Methods, 43, 771–786
- Piccolo D. (2015). Inferential issues for CUBE models with covariates, Communications in Statistics. Theory and Methods, 44(23), 771–786.
- Iannario M. (2015). Detecting latent components in ordinal data with overdispersion by means of a mixture distribution, Quality & Quantity, 49, 977–987
- Iannario M. and Piccolo D. (2016a). A comprehensive framework for regression models of ordinal data. Metron, 74(2), 233–252.
- Iannario M. and Piccolo D. (2016b). A generalized framework for modelling ordinal data. Statistical Methods and Applications, 25, 163–189.

List of TODOs:

- Manual and Examples

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.cube_ywz.**CUBresCUBEYWZ**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([saveas, figsize])	Main function to plot an object of the Class.
plot_ordinal([figsize, ax, kind, saveas])	Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*saveas=None*, *figsize=(7, 5)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns

`ax` or a tuple `(fig, ax)`

plot_ordinal(*figsize=(7, 5)*, *ax=None*, *kind='bar'*, *saveas=None*)

Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns

`ax` or a tuple `(fig, ax)`

cubmods.cube_ywz.**Qdue**(*pars*, *tauno*, *sample*, *W*, *Z*, *m*)

Auxiliary function for the log-likelihood estimation of CUBE models with covariates.

Define the opposite of one of the two scalar functions that are maximized when running the E-M algorithm for CUBE models with covariates for feeling, uncertainty and overdispersion.

Parameters

- **pars** (*array*) – array of initial estimates of parameters for the feeling component and the overdispersion effect
- **tauno** (*array*) – the column vector of the posterior probabilities that each observed rating has been generated by the distribution of the first component of the mixture
- **sample** (*array of int*) – array of ordinal responses
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **Z** (*pandas dataframe*) – dataframe of covariates for explaining the overdispersion
- **m** (*int*) – number of ordinal categories

cubmods.cube_ywz.**Quno**(*beta*, *esterno1*)

Auxiliary function for the log-likelihood estimation of CUBE models with covariates.

Define the opposite one of the two scalar functions that are maximized when running the E-M algorithm for CUBE models with covariates for feeling, uncertainty and overdispersion.

It is iteratively called as an argument of "optim" within CUBE function (with covariates) as the function to minimize to compute the maximum likelihood estimates for the feeling and the overdispersion components.

Parameters

- **beta** (*array of float*) – array β of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **esterno1** (*ndarray*) – matrix binding together the column vector of the posterior probabilities that each observed rating has been generated by the first component distribution of the mixture, with the matrix $\boldsymbol{y}$ of explicative variables for the uncertainty component, expanded with a unitary vector in the first column to consider also an intercept term

cubmods.cube_ywz.**auxmat**(*m*, *xi*, *phi*, *a*, *b*, *c*, *d*, *e*)

Auxiliary matrix.

Returns an auxiliary matrix needed for computing the variance-covariance matrix of a CUBE model with covariates.

Parameters

- **m** (*int*) – number of ordinal categories
- **xi** (*array of float*) – feeling parameters $\boldsymbol{\xi}$
- **phi** (*array of float*) – overdispersion parameter $\boldsymbol{\phi}$
- **a,b,c,d,e** (*float*) – see the reference paper *DOI: 10.1080/03610926.2013.821487* for details

cubmods.cube_ywz.**betabinomial**(*m*, *sample*, *xi*, *phi*)

Beta-Binomial probabilities of ordinal responses, with feeling and overdispersion parameters for each observation.

Compute the Beta-Binomial probabilities of ordinal responses, given feeling and overdispersion parameters for each observation.

The Beta-Binomial distribution is the Binomial distribution in which the probability of success at each trial is random and follows the Beta distribution. It is frequently used in Bayesian statistics, empirical Bayes methods and classical statistics as an overdispersed binomial distribution.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **xi** (*float*) – feeling parameter ξ
- **phi** (*float*) – overdispersion parameter ϕ

Returns
array of the same length as `sample`, containing the Beta-Binomial probabilities of each observation, for the corresponding feeling and overdispersion parameters.

Return type
array

cubmods.cube_ywz.**draw**(*m*, *beta*, *gamma*, *alpha*, *df*, *formula*, *Y*, *W*, *Z*, *seed=None*)

Draw a random sample from a specified CUBE model.

Parameters

- **m** (*int*) – number of ordinal categories
- **n** (*int*) – number of ordinal responses to be drawn
- **beta** (*array of float*) – array β of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array γ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **alpha** (*array of float*) – array α of parameters for the overdispersion, whose length equals `Z.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **Z** (*pandas dataframe*) – dataframe of covariates for explaining the overdispersion
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns
an instance of `CUBsample` containing ordinal responses drawn from the specified model

cubmods.cube_ywz.**init_theta**(*m*, *sample*, *W*, *p*, *v*)

Preliminary parameter estimates for CUBE models with covariates.

Compute preliminary parameter estimates for a CUBE model with covariates for all the three parameters. These estimates are set as initial values to start the E-M algorithm within maximum likelihood estimation.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

- **p** (*int*) – number of covariates for the uncertainty component
- **v** (*int*) – number of covariates for the overdispersion

Returns

a tuple of $(\boldsymbol{\beta}^{(0)}, \boldsymbol{\gamma}^{(0)}, \boldsymbol{\alpha}^{(0)})$ of preliminary estimates of parameter vectors for $\pi = \pi(\boldsymbol{\beta})$, ; xi=xi(pmb{gamma}),; phi=phi(pmb{alpha})` respectively, of a CUBE model with covariates for all the three parameters. In details, they have length equal to `Y.columns.size+1`, `W.columns.size+1` and `Z.columns.size+1`, respectively, to account for an intercept term for each component.

Return type

tuple of arrays

cubmods.cube_ywz.**loglik**(*m*, *sample*, *Y*, *W*, *Z*, *beta*, *gamma*, *alpha*)

Log-likelihood function of a CUBE model with covariates.

Compute the log-likelihood function of a CUBE model for ordinal responses, with covariates for explaining all the three parameters.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **Z** (*pandas dataframe*) – dataframe of covariates for explaining the overdispersion
- **beta** (*array of float*) – array $\boldsymbol{\beta}$ of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **alpha** (*array of float*) – array $\boldsymbol{\alpha}$ of parameters for the overdispersion, whose length equals `Z.columns.size+1` to include an intercept term in the model (first entry)

Returns

the log-likelihood value

Return type

float

cubmods.cube_ywz.**mle**(*m*, *sample*, *Y*, *W*, *Z*, *df*, *formula*, *gen_pars=None*, *maxiter=1000*, *tol=0.01*)

Main function for CUBE models with covariates.

Function to estimate and validate a CUBE model with explicative covariates for all the three parameters.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **Z** (*pandas dataframe*) – dataframe of covariates for explaining the overdispersion
- **df** (*DataFrame*) – original DataFrame

- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None
- **maxiter** (*int*) – maximum number of iterations allowed for running the optimization algorithm
- **tol** (*float*) – fixed error tolerance for final estimates

Returns
an instance of CUBresCUBEYWZ (see the Class for details)

Return type
object

cubmods.cube_ywz.**pmf**(*m*, *beta*, *gamma*, *alpha*, *Y*, *W*, *Z*)

Average probability distribution of a specified CUB model with covariates for the feeling component.

$\frac{1}{n}\sum_{i=1}^{n}\Pr(R=r|\boldsymbol{\theta}_i;\boldsymbol{w}_i),\ r=1\dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **beta** (*array of float*) – array $\boldsymbol{\beta}$ of parameters for the uncertainty component, whose length equals Y.columns.size+1 to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals W.columns.size+1 to include an intercept term in the model (first entry)
- **alpha** (*array of float*) – array $\boldsymbol{\alpha}$ of parameters for the overdispersion, whose length equals Z.columns.size+1 to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **Z** (*pandas dataframe*) – dataframe of covariates for explaining the overdispersion

Returns
the array of the average probability distribution

Return type
numpy array

cubmods.cube_ywz.**pmfi**(*m*, *beta*, *gamma*, *alpha*, *Y*, *W*, *Z*)

Probability distribution for each subject of a specified CUBE model with covariates.

Auxiliary function of .draw().

$\Pr(R=r|\boldsymbol{\theta}_i;\boldsymbol{w}_i),\ i=1\dots n,\ r=1\dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **beta** (*array of float*) – array $\boldsymbol{\beta}$ of parameters for the uncertainty component, whose length equals Y.columns.size+1 to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals W.columns.size+1 to include an intercept term in the model (first entry)
- **alpha** (*array of float*) – array $\boldsymbol{\alpha}$ of parameters for the overdispersion, whose length equals Z.columns.size+1 to include an intercept term in the model (first entry)

- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **Z** (*pandas dataframe*) – dataframe of covariates for explaining the overdispersion

Returns
the matrix of the probability distribution of dimension $n \times r$

Return type
numpy ndarray

cubmods.cube_ywz.**varcov**(*m*, *sample*, *beta*, *gamma*, *alpha*, *Y*, *W*, *Z*)

Variance-covariance matrix of a CUBE model with covariates.

Compute the variance-covariance matrix of parameter estimates of a CUBE model with covariates for all the three parameters.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **Z** (*pandas dataframe*) – dataframe of covariates for explaining the overdispersion
- **beta** (*array of float*) – array $\boldsymbol{\beta}$ of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **alpha** (*array of float*) – array $\boldsymbol{\alpha}$ of parameters for the overdispersion, whose length equals `Z.columns.size+1` to include an intercept term in the model (first entry)

Returns
the variance-covariance matrix

Return type
ndarray

3.1.9 cubmods.cubsh module

CUB models in Python. Module for CUBSH (Combination of Uniform and Binomial with Shelter Effect).

Description:

This module contains methods and classes for CUBSH model family. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

$\Pr(R = r|\boldsymbol{\theta}) = \delta D_r^{(c)} + (1-\delta)[\pi b_r(\xi) + (1-\pi)/m]$

References:

- D'Elia A. (2003). Modelling ranks using the inverse hypergeometric distribution, Statistical Modelling: an International Journal, 3, 65–78
- D'Elia A. and Piccolo D. (2005). A mixture model for preferences data analysis, Computational Statistics & Data Analysis}, bold{49, 917–937
- Capecchi S. and Piccolo D. (2017). Dealing with heterogeneity in ordinal responses, Quality and Quantity, 51(5), 2375–2393
- Iannario M. (2014). Modelling Uncertainty and Overdispersion in Ordinal Data, Communications in Statistics - Theory and Methods, 43, 771–786
- Piccolo D. (2015). Inferential issues for CUBE models with covariates, Communications in Statistics. Theory and Methods, 44(23), 771–786.
- Iannario M. (2015). Detecting latent components in ordinal data with overdispersion by means of a mixture distribution, Quality & Quantity, 49, 977–987
- Iannario M. and Piccolo D. (2016a). A comprehensive framework for regression models of ordinal data. Metron, 74(2), 233–252.
- Iannario M. and Piccolo D. (2016b). A generalized framework for modelling ordinal data. Statistical Methods and Applications, 25, 163–189.

List of TODOs:

- TODO: fix 3d plots legend
- TODO: too long title in CUBsample.plot() ?
- TODO: test all `def _*():` (optional functions)

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.cubsh.**CUBresCUBSH**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([ci, saveas, confell, debug, test3, ...])	Main function to plot an object of the Class.
plot3d(ax[, ci, magnified])	Plots the estimated parameter values in the parameter space and the asymptotic confidence ellipsoid with its projections.
plot_ordinal([figsize, ax, kind, saveas])	Plots relative frequencies of observed sample, estimated probability distribution and, if provided, probability distribution of a known model.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*ci=0.95*, *saveas=None*, *confell=False*, *debug=False*, *test3=True*, *figsize=(7, 15)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **ci** (*float*) – level $(1 - \alpha/2)$ for the confidence ellipsoid
- **confell** (*bool*) – **DEPRECATED**, defaults to False
- **test3** (*bool*) – **DEPRECATED**, defaults to True
- **debug** (*bool*) – **DEPRECATED**, defaults to False
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns

`ax` or a tuple `(fig, ax)`

plot3d(*ax*, *ci=0.95*, *magnified=False*)

Plots the estimated parameter values in the parameter space and the asymptotic confidence ellipsoid with its projections.

Parameters

- **ci** (*float*) – level $(1 - \alpha/2)$ for the confidence ellipsoid
- **magnified** (*bool*) – if False the limits will be the entire parameter space, otherwise let matplotlib choose the limits

- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None

plot_ordinal(*figsize=(7, 5)*, *ax=None*, *kind='bar'*, *saveas=None*)

Plots relative frequencies of observed sample, estimated probability distribution and, if provided, probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

cubmods.cubsh.**cmf**(*m*, *sh*, *pi1*, *pi2*, *xi*)

Cumulative probability of a specified CUBSH model, using alternative parametrization (π_1, π_2).

$\Pr(R \geq r|\boldsymbol{\theta}),\ r = 1 \dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **pi1** (*float*) – Mixing coefficient for the shifted Binomial component of the mixture distribution π_1
- **pi2** (*float*) – Mixing coefficient for the discrete Uniform component of the mixture distribution π_2
- **xi** (*float*) – feeling parameter ξ

Returns
the cumulative probability distribution

Return type
array

cubmods.cubsh.**cmf_delta**(*m*, *sh*, *pi*, *xi*, *delta*)

Cumulative probability of a specified CUBSH model, using canonic parametrization (π, δ).

$\Pr(R \geq r|\boldsymbol{\theta}),\ r = 1 \dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **pi** (*float*) – uncertainty parameter π
- **delta** (*float*) – shelter choice parameter δ
- **xi** (*float*) – feeling parameter ξ

Returns
the cumulative probability distribution

Return type
array

cubmods.cubsh.**draw**(*m*, *sh*, *pi*, *xi*, *delta*, *n*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified CUBSH model, using canonic parametrization (π, δ).

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **pi** (*float*) – uncertainty parameter π
- **delta** (*float*) – shelter choice parameter δ
- **xi** (*float*) – feeling parameter ξ
- **n** (*int*) – number of ordinal responses
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns
an instance of `CUBsample` containing ordinal responses drawn from the specified model

cubmods.cubsh.**draw2**(*m*, *sh*, *pi1*, *pi2*, *xi*, *n*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified CUBSH model, using alternative parametrization (π_1, π_2).

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **pi1** (*float*) – Mixing coefficient for the shifted Binomial component of the mixture distribution π_1
- **pi2** (*float*) – Mixing coefficient for the discrete Uniform component of the mixture distribution π_2
- **xi** (*float*) – feeling parameter ξ
- **n** (*int*) – number of ordinal responses
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns
an instance of `CUBsample` containing ordinal responses drawn from the specified model

cubmods.cubsh.**init_theta**(*f*, *m*, *sh*)

Preliminary estimators for CUBSH models.

Computes preliminary parameter estimates of a CUBSH model without covariates for given ordinal responses. These preliminary estimators are used within the package code to start the E-M algorithm.

Parameters

- **f** (*array of int*) – array of the absolute frequencies of given ordinal responses
- **m** (*int*) – number of ordinal categories

- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$

Returns

a tuple of $(\pi_1^{(0)}, \pi_2^{(0)}, \xi^{(0)})$

cubmods.cubsh.**loglik**(*m*, *sh*, *pi1*, *pi2*, *xi*, *f*)

Log-likelihood of a CUB model with shelter effect

Compute the log-likelihood of a CUB model with a shelter effect for the given absolute frequency distribution.

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **pi1** (*float*) – Mixing coefficient for the shifted Binomial component of the mixture distribution π_1
- **pi2** (*float*) – Mixing coefficient for the discrete Uniform component of the mixture distribution π_2
- **xi** (*float*) – feeling parameter ξ
- **f** (*array*) – Vector of the absolute frequency distribution

Returns

the log-likehood value

Return type

float

cubmods.cubsh.**mean_delta**(*m*, *sh*, *pi*, *xi*, *delta*)

Expected value of a specified CUBSH model, using canonic parametrization (π, δ).

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **pi** (*float*) – uncertainty parameter π
- **delta** (*float*) – shelter choice parameter δ
- **xi** (*float*) – feeling parameter ξ

Returns

the expected value of the model

Return type

float

cubmods.cubsh.**mle**(*sample*, *m*, *sh*, *df*, *formula*, *maxiter=500*, *tol=0.0001*, *gen_pars=None*)

Main function for CUB models with a shelter effect

Estimate and validate a CUB model with a shelter effect.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **df** (*DataFrame*) – original DataFrame

- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None
- **maxiter** (*int*) – maximum number of iterations allowed for running the optimization algorithm
- **tol** (*float*) – fixed error tolerance for final estimates

Returns
an instance of CUBresCUBSH (see the Class for details)

Return type
object

Raise
Exception if $m \leq 4$

cubmods.cubsh.**pi1pi2_to_pidelta**(*pi1*, *pi2*)

Compute (π, δ) from (π_1, π_2)

$\pi = \dfrac{\pi_1}{\pi_1 + \pi_2}$

$\delta = 1 - \pi_1 - \pi_2$

Parameters

- **pi1** (*float*) – Mixing coefficient for the shifted Binomial component of the mixture distribution π_1
- **pi2** (*float*) – Mixing coefficient for the discrete Uniform component of the mixture distribution π_2

Returns
a tuple of (π, δ) the parameters of uncertainty and shelter choice, respectively

Return type
tuple

cubmods.cubsh.**pidelta_to_pi1pi2**(*pi*, *delta*)

Compute (π_1, π_2) from (π, δ)

$\pi_1 = (1 - \delta)\pi$

$\pi_2 = (1 - \delta)(1 - \pi)$

Parameters

- **pi** (*float*) – uncertainty parameter π
- **delta** (*float*) – shelter choice parameter δ

Returns
a tuple of (π_1, π_2) the mixing coefficient of the shifted Binomial and the Uniform components, respectively

Return type
tuple

cubmods.cubsh.**plot_simplex**(*pi1pi2list*, *ax=None*, *fname=None*)

Plot simplex of parameters of a CUBSH model.

Note: see the reference DOI 10.1007/s10260-011-0176-x for details

Warning: this function still needs several fixes

Parameters

- **pi1pi2list** (*list*) – list of `[pi1, pi2]` parameters
- **ax** – matplotlib axis
- **fname** – if provided, save the plot to `fname`, defaults to None
- **fname** – str

cubmods.cubsh.**pmf**(*m*, *sh*, *pi1*, *pi2*, *xi*)

Probability distribution of a specified CUBSH model, using alternative parametrization (π_1, π_2).

$\Pr(R = r|\boldsymbol{\theta}),\ r = 1 \dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **pi1** (*float*) – Mixing coefficient for the shifted Binomial component of the mixture distribution π_1
- **pi2** (*float*) – Mixing coefficient for the discrete Uniform component of the mixture distribution π_2
- **xi** (*float*) – feeling parameter ξ

Returns
the probability distribution

Return type
array

cubmods.cubsh.**pmf_delta**(*m*, *sh*, *pi*, *xi*, *delta*)

Probability distribution of a specified CUBSH model, using canonic parametrization (π, δ).

$\Pr(R = r|\boldsymbol{\theta}),\ r = 1 \dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **pi** (*float*) – uncertainty parameter π
- **delta** (*float*) – shelter choice parameter δ
- **xi** (*float*) – feeling parameter ξ

Returns
the probability distribution

Return type
array

cubmods.cubsh.**proba**(*m*, *sh*, *pi1*, *pi2*, *xi*, *r*)

Probability $\Pr(R = r|\boldsymbol{\theta})$ of a CUBSH model without covariates, using alternative parametrization (π_1, π_2).

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **pi1** (*float*) – Mixing coefficient for the shifted Binomial component of the mixture distribution π_1
- **pi2** (*float*) – Mixing coefficient for the discrete Uniform component of the mixture distribution π_2
- **xi** (*float*) – feeling parameter ξ
- **r** (*int*) – ordinal response

Returns
the probability $\Pr(R = r|\boldsymbol{\theta})$

Return type
float

cubmods.cubsh.**proba_delta**(*m*, *sh*, *pi*, *xi*, *delta*, *r*)

Probability $\Pr(R = r|\boldsymbol{\theta})$ of a CUBSH model without covariates, using canonic parametrization (π, δ).

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **pi** (*float*) – uncertainty parameter π
- **delta** (*float*) – shelter choice parameter δ
- **xi** (*float*) – feeling parameter ξ
- **r** (*int*) – ordinal response

Returns
the probability $\Pr(R = r|\boldsymbol{\theta})$

Return type
float

cubmods.cubsh.**std_delta**(*m*, *pi*, *xi*, *delta*)

Standard deviation of a specified CUB model, using canonic parametrization (π, δ).

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **delta** (*float*) – shelter choice parameter δ
- **xi** (*float*) – feeling parameter ξ

Returns
the standard deviation of the model

Return type
float

cubmods.cubsh.**var_delta**(*m*, *pi*, *xi*, *delta*)

Variance of a specified CUBSH model, using canonic parametrization (π, δ).

Parameters

- **m** (*int*) – number of ordinal categories
- **pi** (*float*) – uncertainty parameter π
- **delta** (*float*) – shelter choice parameter δ
- **xi** (*float*) – feeling parameter ξ

Returns
the variance of the model

Return type
float

cubmods.cubsh.**varcov**(*m*, *sh*, *pi1*, *pi2*, *xi*, *n*)

Variance-covariance matrix for CUB models with shelter effect, using alternative parametrization (π_1, π_2).

Compute the variance-covariance matrix of parameter estimates of a CUB model with shelter effect.

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **pi1** (*float*) – Mixing coefficient for the shifted Binomial component of the mixture distribution π_1
- **pi2** (*float*) – Mixing coefficient for the discrete Uniform component of the mixture distribution π_2
- **xi** (*float*) – feeling parameter ξ
- **n** (*int*) – number of ordinal responses

Returns
the variance-covariance matrix

Return type
numpy ndarray

cubmods.cubsh.**varcov_pxd**(*m*, *sh*, *pi*, *xi*, *de*, *n*)

Variance-covariance matrix for CUB models with shelter effect, using canonic parametrization (π, δ).

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **pi** (*float*) – uncertainty parameter π
- **de** (*float*) – shelter choice parameter δ
- **xi** (*float*) – feeling parameter ξ
- **n** (*int*) – number of ordinal responses

Returns
the variance-covariance matrix

Return type
numpy ndarray

3.1.10 cubmods.cubsh_ywx module

CUB models in Python. Module for CUBSH (Combination of Uniform and Binomial with Shelter Effect) with covariates.

Description:

This module contains methods and classes for CUBSH_YWX model family. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

$$\Pr(R = r_i|\boldsymbol{\theta}_i; \boldsymbol{y}_i; \boldsymbol{w}_i; \boldsymbol{x}_i) = \delta_i D_r^{(c)} + (1 - \delta_i)[\pi_i b_r(\xi_i) + (1 - \pi_i)/m]$$

$$\xi_i = \frac{1}{1 + e^{-\boldsymbol{w}_i\boldsymbol{\gamma}}}$$

$$\pi_i = \frac{1}{1 + e^{-\boldsymbol{y}_i\boldsymbol{\beta}}}$$

$$\delta_i = \frac{1}{1 + e^{-\boldsymbol{x}_i\boldsymbol{\omega}}}$$

Manual and Examples

- Manual https://github.com/maxdevblock/cubmods/blob/main/Manual/03_cubsh_family.md

References:

- D'Elia A. (2003). Modelling ranks using the inverse hypergeometric distribution, Statistical Modelling: an International Journal, 3, 65–78
- D'Elia A. and Piccolo D. (2005). A mixture model for preferences data analysis, Computational Statistics & Data Analysis}, bold{49, 917–937
- Capecchi S. and Piccolo D. (2017). Dealing with heterogeneity in ordinal responses, Quality and Quantity, 51(5), 2375–2393
- Iannario M. (2014). Modelling Uncertainty and Overdispersion in Ordinal Data, Communications in Statistics - Theory and Methods, 43, 771–786
- Piccolo D. (2015). Inferential issues for CUBE models with covariates, Communications in Statistics. Theory and Methods, 44(23), 771–786.
- Iannario M. (2015). Detecting latent components in ordinal data with overdispersion by means of a mixture distribution, Quality & Quantity, 49, 977–987
- Iannario M. and Piccolo D. (2016a). A comprehensive framework for regression models of ordinal data. Metron, 74(2), 233–252.
- Iannario M. and Piccolo D. (2016b). A generalized framework for modelling ordinal data. Statistical Methods and Applications, 25, 163–189.

List of TODOs:

- ...

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.cubsh_ywx.**CUBresCUBSHYWX**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([saveas, figsize])	Main function to plot an object of the Class.
plot_ordinal([figsize, ax, kind, saveas])	Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*saveas=None*, *figsize=(7, 5)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

plot_ordinal(*figsize=(7, 5)*, *ax=None*, *kind='bar'*, *saveas=None*)

Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

cubmods.cubsh_ywx.**Q1**(*param*, *dati1*, *p*)

Auxiliary function for the log-likelihood estimation of GeCUB models.

Define the opposite one of the two scalar functions that are maximized when running the E-M algorithm for GeCUB models with covariates for feeling, uncertainty and shelter effect.

Parameters

- **param** (*array*) – array of initial estimates of parameters for the uncertainty component
- **dati1** (*ndarray or dataframe*) – auxiliary matrix
- **p** (*int*) – number of covariates for the uncertainty component

cubmods.cubsh_ywx.**Q2**(*param*, *dati2*, *m*)

Auxiliary function for the log-likelihood estimation of GeCUB models.

Define the opposite one of the two scalar functions that are maximized when running the E-M algorithm for GeCUB models with covariates for feeling, uncertainty and shelter effect.

Parameters

- **param** (*array*) – array of initial estimates of parameters for the feeling component
- **dati2** (*ndarray or dataframe*) – auxiliary matrix
- **m** (*int*) – number of ordinal categories

cubmods.cubsh_ywx.**draw**(*m*, *sh*, *beta*, *gamma*, *omega*, *Y*, *W*, *X*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified CUBSH model with covariates (aka GeCUB model).

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **beta** (*array of float*) – array β of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)

- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **omega** (*array*) – array $\boldsymbol{\omega}$ of parameters for the shelter effect, whose length equals `X.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **X** (*pandas dataframe*) – dataframe of covariates for explaining the shelter effect
- **n** (*int*) – number of ordinal responses to be drawn
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns

an instance of `CUBsample` containing ordinal responses drawn from the specified model

cubmods.cubsh_ywx.**init_theta**(*m*, *sample*, *p*, *s*, *W*)

Preliminary estimators for CUBSH models with covariates.

Computes preliminary parameter estimates of a CUBSH model without covariates for given ordinal responses. These preliminary estimators are used within the package code to start the E-M algorithm.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **p** (*int*) – number of covariates for the uncertainty component
- **s** (*int*) – number of covariates for the shelter effect
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component

Returns

a tuple of $(\boldsymbol{\beta}^{(0)}, \boldsymbol{\gamma}^{(0)}, \boldsymbol{\omega}^{(0)})$ of preliminary estimates of parameter vectors for $\pi = \pi(\boldsymbol{\beta})$, ; xi=xi(pmb{gamma}),; delta=delta(pmb{omega})` respectively, of a CUBSH model with covariates for all the three parameters. In details, they have length equal to `Y.columns.size+1`, `W.columns.size+1` and `X.columns.size+1`, respectively, to account for an intercept term for each component.

Return type

tuple of arrays

cubmods.cubsh_ywx.**loglik**(*m*, *sample*, *sh*, *Y*, *W*, *X*, *beta*, *gamma*, *omega*)

Log-likelihood function of a CUBSH model with covariates.

Compute the log-likelihood function of a CUBE model for ordinal responses, with covariates for explaining all the three parameters (GeCUB model).

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **beta** (*array of float*) – array $\boldsymbol{\beta}$ of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)

- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **omega** (*array*) – array $\boldsymbol{\omega}$ of parameters for the shelter effect, whose length equals `X.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **X** (*pandas dataframe*) – dataframe of covariates for explaining the shelter effect
- **sample** (*array of int*) – array of ordinal responses

Returns
the log-likelihood value

Return type
float

cubmods.cubsh_ywx.**mle**(*m*, *sample*, *sh*, *Y*, *W*, *X*, *df*, *formula*, *gen_pars=None*, *maxiter=500*, *tol=0.0001*)

Main function for CUBSH models with covariates for all the components

Function to estimate and validate a CUBSH model for given ordinal responses, with covariates for explaining all the components and the shelter effect.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **X** (*pandas dataframe*) – dataframe of covariates for explaining the shelter effect
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None
- **maxiter** (*int*) – maximum number of iterations allowed for running the optimization algorithm
- **tol** (*float*) – fixed error tolerance for final estimates

Returns
an instance of `CUBresCUBSHYWZ` (see the Class for details)

Return type
object

cubmods.cubsh_ywx.**pmf**(*m*, *sh*, *beta*, *gamma*, *omega*, *Y*, *W*, *X*)

Average probability distribution of a specified CUBSH model with covariates (aka GeCUB model).

$\frac{1}{n}\sum_{i=1}^{n} \Pr(R = r|\boldsymbol{\theta}_i; \boldsymbol{w}_i; \boldsymbol{y}_i, \boldsymbol{x}_i),\ r = 1 \dots m$

Parameters

- **m** (*int*) – number of ordinal categories

- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **beta** (*array of float*) – array $\boldsymbol{\beta}$ of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **omega** (*array*) – array $\boldsymbol{\omega}$ of parameters for the shelter effect, whose length equals `X.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **X** (*pandas dataframe*) – dataframe of covariates for explaining the shelter effect

Returns
the probability distribution

Return type
array

cubmods.cubsh_ywx.**pmfi**(*m*, *sh*, *beta*, *gamma*, *omega*, *Y*, *W*, *X*)

Probability distribution for each subject of a specified CUBSH model with covariates (aka GeCUB model).

Auxiliary function of `.draw()`.

$\Pr(R = r|\boldsymbol{\theta}_i; \boldsymbol{y}_i; \boldsymbol{w}_i, \boldsymbol{x}_i),\ i = 1 \dots n,\ r = 1 \dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **beta** (*array of float*) – array $\boldsymbol{\beta}$ of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array $\boldsymbol{\gamma}$ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **omega** (*array*) – array $\boldsymbol{\omega}$ of parameters for the shelter effect, whose length equals `X.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **X** (*pandas dataframe*) – dataframe of covariates for explaining the shelter effect

Returns
the matrix of the probability distribution of dimension $n \times r$

Return type
numpy ndarray

cubmods.cubsh_ywx.**prob**(*m*, *sample*, *sh*, *Y*, *W*, *X*, *beta*, *gamma*, *omega*)

Probability distribution of a CUBSH model with covariates.

Compute the probability distribution of a CUBSH model with covariates.

$\Pr(R = r_i|\boldsymbol{\theta}_i; \boldsymbol{w}_i; \boldsymbol{y}_i; \boldsymbol{x}_i),\ i = 1 \dots n$

Parameters

- **m** (*int*) – number of ordinal categories

- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **beta** (*array of float*) – array β of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array γ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **omega** (*array*) – array ω of parameters for the shelter effect, whose length equals `X.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **X** (*pandas dataframe*) – dataframe of covariates for explaining the shelter effect
- **sample** (*array of int*) – array of ordinal responses

Returns
the array of the probability distribution.

Return type
numpy array

cubmods.cubsh_ywx.**varcov**(*sample*, *m*, *sh*, *Y*, *W*, *X*, *beta*, *gamma*, *omega*)

Variance-covariance matrix of a CUBSH model with covariates

Compute the variance-covariance matrix of parameter estimates of a CUBSH model with covariates.

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **beta** (*array of float*) – array β of parameters for the uncertainty component, whose length equals `Y.columns.size+1` to include an intercept term in the model (first entry)
- **gamma** (*array of float*) – array γ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)
- **omega** (*array*) – array ω of parameters for the shelter effect, whose length equals `X.columns.size+1` to include an intercept term in the model (first entry)
- **Y** (*pandas dataframe*) – dataframe of covariates for explaining the uncertainty component
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **X** (*pandas dataframe*) – dataframe of covariates for explaining the shelter effect
- **sample** (*array of int*) – array of ordinal responses

Returns
the variance-covariance matrix of the model

Return type
numpy ndarray

3.1.11 cubmods.cush module

CUB models in Python. Module for CUSH (Combination of Uniform and Shelter effect).

Description:

This module contains methods and classes for CUSH model family. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

$\Pr(R = r|\boldsymbol{\theta}) = \delta D_r^{(c)} + (1 - \delta)/m$

Manual and Examples

- Manual https://github.com/maxdevblock/cubmods/blob/main/Manual/05_cush_family.md

References:

1. Stefania Capecchi and Domenico Piccolo. Dealing with heterogeneity in ordinal responses. *Quality & Quantity*, 51:2375–2393, 2017.

List of TODOs:

- TODO: check and fix gini & laakso

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.cush.**CUBresCUSH**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([ci, saveas, figsize])	Main function to plot an object of the Class.
plot_estim([ci, ax, magnified, figsize, saveas])	Plots the estimated parameter values in the parameter space and the asymptotic standard error.
plot_ordinal([figsize, kind, ax, saveas])	Plots relative frequencies of observed sample, estimated probability distribution and, if provided, probability distribution of a known model.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*ci=0.95*, *saveas=None*, *figsize=(7, 15)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **ci** (*float*) – level $(1 - \alpha/2)$ for the standard error
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns

`ax` or a tuple `(fig, ax)`

plot_estim(*ci=0.95*, *ax=None*, *magnified=False*, *figsize=(7, 7)*, *saveas=None*)

Plots the estimated parameter values in the parameter space and the asymptotic standard error.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **ci** (*float*) – level $(1 - \alpha/2)$ for the confidence ellipse
- **magnified** (*bool*) – if False the limits will be the entire parameter space, otherwise let matplotlib choose the limits
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
ax or a tuple (fig, ax)

plot_ordinal(*figsize=(7, 7)*, *kind='bar'*, *ax=None*, *saveas=None*)

Plots relative frequencies of observed sample, estimated probability distribution and, if provided, probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of (length, height) for the figure (useful only if ax is not None)
- **kind** (*str*) – choose a barplot ('bar' default) of a scatterplot ('scatter')
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
ax or a tuple (fig, ax)

cubmods.cush.**LRT**(*m*, *fc*, *n*)

Likelihood Ratio Test between the CUSH model and the null model.

Parameters

- **m** (*int*) – number of ordinal categories
- **fc** (*float*) – relative frequency of the shelter category
- **n** (*int*) – number of observations

Returns
the value of the LRT

Return type
float

cubmods.cush.**draw**(*m*, *sh*, *delta*, *n*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified CUSH model.

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **delta** (*float*) – shelter choice parameter δ
- **n** (*int*) – number of ordinal responses
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns
an instance of CUBsample containing ordinal responses drawn from the specified model

cubmods.cush.**gini**(*delta*)

The Gini index of a specified CUSH model.

Parameters
delta (*float*) – shelter choice parameter δ

Returns
the Gini index of the model

Return type
float

cubmods.cush.**laakso**(*m*, *delta*)

The Laakso index of a specified CUSH model.

Parameters

- **m** (*int*) – number of ordinal categories
- **delta** (*float*) – shelter choice parameter δ

Returns
the Laakso index of the model

Return type
float

cubmods.cush.**loglik**(*sample*, *m*, *sh*, *delta*)

Log-likelihood function for a CUSH model without covariates

Compute the log-likelihood function for a CUSH model without covariate for the given ordinal responses.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **delta** (*float*) – shelter choice parameter δ

Returns
the log-likehood value

Return type
float

cubmods.cush.**mean**(*m*, *sh*, *delta*)

Expected value of a specified CUSH model.

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **delta** (*float*) – shelter choice parameter δ

Returns
the expected value of the model

Return type
float

cubmods.cush.**mle**(*sample*, *m*, *sh*, *df*, *formula*, *gen_pars=None*, *maxiter=None*, *tol=None*)

Main function for CUSH model without covariates.

Estimate and validate a CUSH model for given ordinal responses, without covariates.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None
- **maxiter** (*None*) – default to None; ensure compatibility with `gem.from_formula()`
- **tol** (*None*) – default to None; ensure compatibility with `gem.from_formula()`

Returns
an instance of `CUBresCUSH` (see the Class for details)

Return type
object

`cubmods.cush.`**pmf**(*m*, *sh*, *delta*)

Probability distribution of a specified CUSH model.

$\Pr(R = r|\boldsymbol{\theta}),\ r = 1 \ldots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **delta** (*float*) – shelter choice parameter δ

Returns
the probability distribution

Return type
array

`cubmods.cush.`**var**(*m*, *sh*, *delta*)

Variance of a specified CUSH model.

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **delta** (*float*) – shelter choice parameter δ

Returns
the variance of the model

Return type
float

3.1.12 cubmods.cush2 module

CUB models in Python. Module for CUSH2 (Combination of Uniform and 2 Shelter Choices).

Description:

This module contains methods and classes for CUSH2 model family. It is based upon the works of Domenico Piccolo et Al. and CUB package in R. The CUSH2 family has been defined and implemented by Massimo Pierini (2024) in the thesis *Modelli della classe CUB in Python*.

$\Pr(R = r|\boldsymbol{\theta}) = \delta_1 D_r^{(c_1)} + \delta_2 D_r^{(c_2)} + (1 - \delta_1 - \delta_2)/m$

References:

1. Giovanni Cerulli, R Simone, Francesca Di Iorio, D Piccolo, CF Baum, and others. The stata module for cub models for rating data analysis. In *London Stata Conference 2021*, number 16. Stata Users Group, 2021.
2. Maria Iannario, Domenico Piccolo, and Maintainer Rosaria Simone. Package 'cub'. *CRAN*, 2022.
3. Domenico Piccolo and others. On the moments of a mixture of uniform and shifted binomial random variables. *Quaderni di Statistica*, 5(1):85–104, 2003.
4. Domenico Piccolo and others. Observed information matrix for mub models. *Quaderni di Statistica*, 8(1):33–78, 2006.
5. Domenico Piccolo and Rosaria Simone. The class of cub models: statistical foundations, inferential issues and empirical evidence. *Statistical Methods & Applications*, 28:389–435, 2019.
6. Massimo Pierini. Modelli della classe cub in python. *Universitas Mercatorum, Rome, IT*, pages 16–20, June 2024. (Bachelor's thesis L-41).
7. W Stephen Pittard and Shuzhao Li. The essential toolbox of data science: python, r, git, and docker. *Computational Methods and Data Analysis for Metabolomics*, pages 265–311, 2020.
8. Rosaria Simone, Francesca Di Iorio, and Riccardo Lucchetti. Cub for gretl. *GNU Regression, Econometrics and Time Series Library*, pages 147, 2019.

List of TODOs:

- …

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.cush2.**CUBresCUSH2**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([ci, saveas, figsize])	Main function to plot an object of the Class.
plot_ordinal([figsize, ax, kind, saveas])	Plots relative frequencies of observed sample, estimated probability distribution and, if provided, probability distribution of a known model.
plot_par_space([figsize, ax, ci, saveas])	Plots the estimated parameter values in the parameter space and the asymptotic standard error.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*ci=0.95*, *saveas=None*, *figsize=(7, 11)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **ci** (*float*) – level $(1 - \alpha/2)$ for the standard error
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

plot_ordinal(*figsize=(7, 5)*, *ax=None*, *kind='bar'*, *saveas=None*)

Plots relative frequencies of observed sample, estimated probability distribution and, if provided, probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)

- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
ax or a tuple (fig, ax)

plot_par_space(*figsize=(7, 5)*, *ax=None*, *ci=0.95*, *saveas=None*)

Plots the estimated parameter values in the parameter space and the asymptotic standard error.

Parameters

- **figsize** (*tuple of float*) – tuple of (length, height) for the figure (useful only if ax is not None)
- **ci** (*float*) – level $(1 - \alpha/2)$ for the confidence ellipse
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
ax or a tuple (fig, ax)

cubmods.cush2.**draw**(*m*, *sh1*, *sh2*, *df*, *formula*, *delta1*, *delta2*, *n*, *seed=None*)

Draw a random sample from a specified CUSH2 model.

Parameters

- **m** (*int*) – number of ordinal categories
- **sh1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **sh2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$
- **delta1** (*float*) – 1st shelter choice parameter δ_1
- **delta2** (*float*) – 2nd shelter choice parameter δ_2
- **n** (*int*) – number of ordinal responses
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns
an instance of CUBsample containing ordinal responses drawn from the specified model

cubmods.cush2.**loglik**(*sample*, *m*, *c1*, *c2*)

Log-likelihood function for a CUSH2 model without covariates.

Compute the log-likelihood function for a CUSH2 model without covariate for the given ordinal responses.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **c1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **c2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$

Returns
the log-likehood value

Return type
float

cubmods.cush2.**mle**(*sample*, *m*, *c1*, *c2*, *df*, *formula*, *gen_pars=None*, *maxiter=None*, *tol=None*)

Main function for CUSH2 models without covariates.

Estimate and validate a CUSH2 model for ordinal responses, without covariates.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **c1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **c2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None
- **maxiter** (*None*) – default to None; ensure compatibility with `gem.from_formula()`
- **tol** (*None*) – default to None; ensure compatibility with `gem.from_formula()`

Returns
an instance of `CUBresCUSH2` (see the Class for details)

Return type
object

cubmods.cush2.**pmf**(*m*, *c1*, *c2*, *d1*, *d2*)

Probability distribution of a specified CUSH2 model.

$\Pr(R = r|\boldsymbol{\theta}),\ r = 1 \ldots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **c1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **c2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$
- **d1** (*float*) – 1st shelter choice parameter δ_1
- **d2** (*float*) – 2nd shelter choice parameter δ_2

Returns
the probability distribution

Return type
array

cubmods.cush2.**varcov**(*m*, *n*, *d1*, *d2*, *fc1*, *fc2*)

Compute the variance-covariance matrix of parameter estimates of a CUSH2 model without covariates.

Parameters

- **m** (*int*) – number of ordinal categories

- **n** (*int*) – number of ordinal responses
- **d1** (*float*) – 1st shelter choice parameter δ_1
- **d2** (*float*) – 2nd shelter choice parameter δ_2
- **fc1** (*float*) – relative frequency of 1st shelter choice
- **fc2** (*float*) – relative frequency of 2nd shelter choice

Returns
the variance-covariance matrix

Return type
numpy ndarray

3.1.13 cubmods.cush2_x0 module

CUB models in Python. Module for CUSH2 (Combination of Uniform and 2 Shelter Choices) with covariates.

Description:

This module contains methods and classes for CUSH2 model family. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

References:

1. Giovanni Cerulli, R Simone, Francesca Di Iorio, D Piccolo, CF Baum, and others. The stata module for cub models for rating data analysis. In *London Stata Conference 2021*, number 16. Stata Users Group, 2021.
2. Maria Iannario, Domenico Piccolo, and Maintainer Rosaria Simone. Package 'cub'. *CRAN*, 2022.
3. Domenico Piccolo and others. On the moments of a mixture of uniform and shifted binomial random variables. *Quaderni di Statistica*, 5(1):85–104, 2003.
4. Domenico Piccolo and others. Observed information matrix for mub models. *Quaderni di Statistica*, 8(1):33–78, 2006.
5. Domenico Piccolo and Rosaria Simone. The class of cub models: statistical foundations, inferential issues and empirical evidence. *Statistical Methods & Applications*, 28:389–435, 2019.
6. Massimo Pierini. Modelli della classe cub in python. *Universitas Mercatorum, Rome, IT*, pages 16–20, June 2024. (Bachelor's thesis L-41).
7. W Stephen Pittard and Shuzhao Li. The essential toolbox of data science: python, r, git, and docker. *Computational Methods and Data Analysis for Metabolomics*, pages 265–311, 2020.
8. Rosaria Simone, Francesca Di Iorio, and Riccardo Lucchetti. Cub for gretl. *GNU Regression, Econometrics and Time Series Library*, pages 147, 2019.

List of TODOs:

- ...

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.cush2_x0.**CUBresCUSH2X0**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([saveas, figsize])	Main function to plot an object of the Class.
plot_ordinal([figsize, ax, kind, saveas])	Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*saveas=None*, *figsize=(7, 5)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

plot_ordinal(*figsize=(7, 5)*, *ax=None*, *kind='bar'*, *saveas=None*)

Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

cubmods.cush2_x0.**draw**(*m*, *sh1*, *sh2*, *omega1*, *delta2*, *X1*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified CUSH2 model, with covariates for the 1st shelter choice only.

Parameters

- **m** (*int*) – number of ordinal categories
- **sh1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **sh2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$
- **omega1** (*array*) – array $\boldsymbol{\omega}_1$ of parameters for the 1st shelter effect, whose length equals `X1.columns.size+1` to include an intercept term in the model (first entry)
- **delta2** (*float*) – 2nd shelter choice parameter δ_2
- **X1** (*DataFrame*) – dataframe of covariates for explaining the 1st shelter effect
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns
an instance of `CUBsample` containing ordinal responses drawn from the specified model

cubmods.cush2_x0.**effe**(*pars*, *sample*, *m*, *sh1*, *sh2*, *X1*)

Auxiliary function for the log-likelihood estimation of CUSH2 models.

Compute the opposite of the scalar function that is maximized when running the E-M algorithm for CUSH2 models with covariates for the 1st shelter choice.

Parameters

- **pars** (*array*) – array of parameters
- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories

- **sh1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **sh2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$
- **X1** (*DataFrame*) – dataframe of covariates for explaining the 1st shelter effect

cubmods.cush2_x0.**loglik**(*sample*, *m*, *sh1*, *sh2*, *omega1*, *delta2*, *X1*)

Log-likelihood function for a CUSH2 model with covariates for the 1st shelter choice only.

Compute the log-likelihood function for a CUSH2 model with covariates for the 1st shelter choice only, for the given ordinal responses.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **sh1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **sh2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$
- **omega1** (*array*) – array $\boldsymbol{\omega}_1$ of parameters for the 1st shelter effect, whose length equals `X1.columns.size+1` to include an intercept term in the model (first entry)
- **delta2** (*float*) – 2nd shelter choice parameter δ_2
- **X1** (*DataFrame*) – dataframe of covariates for explaining the 1st shelter effect

Returns
the log-likehood value

Return type
float

cubmods.cush2_x0.**mle**(*sample*, *m*, *sh1*, *sh2*, *X1*, *df*, *formula*, *gen_pars=None*)

Main function for CUSH2 models with covariates for the 1st shelter choice only.

Estimate and validate a CUSH2 model for given ordinal responses, with covariates for the 1st shelter choice only.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **sh1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **sh2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$
- **X1** (*DataFrame*) – dataframe of covariates for explaining the 1st shelter effect
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None

Returns
an instance of `CUBresCUSH2X0` (see the Class for details)

Return type
object

cubmods.cush2_x0.**pmf**(*m*, *sh1*, *sh2*, *omega1*, *delta2*, *X1*)

Average probability distribution of a specified CUSH2 model with covariates for the 1st shelter choice.

$\frac{1}{n}\sum_{i=1}^{n}\Pr(R=r|\boldsymbol{\theta}_i;\boldsymbol{x}_{1i}),\ r=1\dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **sh1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **sh2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$
- **omega1** (*array*) – array $\boldsymbol{\omega}_1$ of parameters for the 1st shelter effect, whose length equals `X1.columns.size`+1 to include an intercept term in the model (first entry)
- **delta2** (*float*) – 2nd shelter choice parameter δ_2
- **X1** (*DataFrame*) – dataframe of covariates for explaining the 1st shelter effect

Returns
the average probability distribution

Return type
array

`cubmods.cush2_x0.pmfi`(*m*, *sh1*, *sh2*, *omega1*, *delta2*, *X1*)

Probability distribution for each subject of a specified CUSH2 model with covariates for the first shelter choice only.

Auxiliary function of `.draw()`.

$\Pr(R = r|\boldsymbol{\theta}_i; \boldsymbol{x}_{1i}),\ i = 1 \dots n,\ r = 1 \dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **sh1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **sh2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$
- **omega1** (*array*) – array $\boldsymbol{\omega}_1$ of parameters for the 1st shelter effect, whose length equals `X1.columns.size`+1 to include an intercept term in the model (first entry)
- **delta2** (*float*) – 2nd shelter choice parameter δ_2
- **X1** (*DataFrame*) – dataframe of covariates for explaining the 1st shelter effect

Returns
the matrix of the probability distribution of dimension $n \times r$

Return type
numpy ndarray

3.1.14 cubmods.cush2_xx module

CUB models in Python. Module for CUSH2 (Combination of Uniform and 2 Shelter Choices) with covariates.

Description:

This module contains methods and classes for CUSH2 model family with covariates for both shelter choices. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

References:

1. Giovanni Cerulli, R Simone, Francesca Di Iorio, D Piccolo, CF Baum, and others. The stata module for cub models for rating data analysis. In *London Stata Conference 2021*, number 16. Stata Users Group, 2021.
2. Maria Iannario, Domenico Piccolo, and Maintainer Rosaria Simone. Package 'cub'. *CRAN*, 2022.
3. Domenico Piccolo and others. On the moments of a mixture of uniform and shifted binomial random variables. *Quaderni di Statistica*, 5(1):85–104, 2003.
4. Domenico Piccolo and others. Observed information matrix for mub models. *Quaderni di Statistica*, 8(1):33–78, 2006.
5. Domenico Piccolo and Rosaria Simone. The class of cub models: statistical foundations, inferential issues and empirical evidence. *Statistical Methods & Applications*, 28:389–435, 2019.
6. Massimo Pierini. Modelli della classe cub in python. *Universitas Mercatorum, Rome, IT*, pages 16–20, June 2024. (Bachelor's thesis L-41).
7. W Stephen Pittard and Shuzhao Li. The essential toolbox of data science: python, r, git, and docker. *Computational Methods and Data Analysis for Metabolomics*, pages 265–311, 2020.
8. Rosaria Simone, Francesca Di Iorio, and Riccardo Lucchetti. Cub for gretl. *GNU Regression, Econometrics and Time Series Library*, pages 147, 2019.

List of TODOs:

- …

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.cush2_xx.**CUBresCUSH2XX**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([saveas, figsize])	Main function to plot an object of the Class.
plot_ordinal([figsize, ax, kind, saveas])	Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*saveas=None*, *figsize=(7, 5)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns

`ax` or a tuple `(fig, ax)`

plot_ordinal(*figsize=(7, 5)*, *ax=None*, *kind='bar'*, *saveas=None*)

Plots relative average frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns

`ax` or a tuple `(fig, ax)`

cubmods.cush2_xx.**draw**(*m*, *sh1*, *sh2*, *omega1*, *omega2*, *X1*, *X2*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified CUSH2 model, with covariates for both shelter choices.

Parameters

- **m** (*int*) – number of ordinal categories
- **sh1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **sh2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$
- **omega1** (*array*) – array $\boldsymbol{\omega}_1$ of parameters for the 1st shelter effect, whose length equals `X1.columns.size`+1 to include an intercept term in the model (first entry)
- **omega2** (*array*) – array $\boldsymbol{\omega}_2$ of parameters for the 2nd shelter effect, whose length equals `X2.columns.size`+1 to include an intercept term in the model (first entry)
- **X1** (*DataFrame*) – dataframe of covariates for explaining the 1st shelter effect
- **X2** (*DataFrame*) – dataframe of covariates for explaining the 2nd shelter effect
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns

an instance of `CUBsample` containing ordinal responses drawn from the specified model

cubmods.cush2_xx.**effe**(*pars*, *sample*, *m*, *sh1*, *sh2*, *X1*, *X2*)

Auxiliary function for the log-likelihood estimation of CUSH2 models.

Compute the opposite of the scalar function that is maximized when running the E-M algorithm for CUSH2 models with covariates for both shelter choices.

Parameters

- **pars** (*array*) – array of parameters
- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **sh1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **sh2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$
- **X1** (*DataFrame*) – dataframe of covariates for explaining the 1st shelter effect
- **X2** (*DataFrame*) – dataframe of covariates for explaining the 2nd shelter effect

cubmods.cush2_xx.**loglik**(*sample*, *m*, *sh1*, *sh2*, *omega1*, *omega2*, *X1*, *X2*)

Log-likelihood function for a CUSH2 model with covariates for both shelter choices.

Compute the log-likelihood function for a CUSH2 model with covariates for both shelter choices, for the given ordinal responses.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **sh1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **sh2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$

- **omega1** (*array*) – array $\boldsymbol{\omega}_1$ of parameters for the 1st shelter effect, whose length equals `X1.columns.size+1` to include an intercept term in the model (first entry)
- **omega2** (*array*) – array $\boldsymbol{\omega}_2$ of parameters for the 2nd shelter effect, whose length equals `X2.columns.size+1` to include an intercept term in the model (first entry)
- **X1** (*DataFrame*) – dataframe of covariates for explaining the 1st shelter effect
- **X2** (*DataFrame*) – dataframe of covariates for explaining the 2nd shelter effect

Returns
the log-likehood value

Return type
float

`cubmods.cush2_xx.mle`(*sample*, *m*, *sh1*, *sh2*, *X1*, *X2*, *df*, *formula*, *gen_pars=None*)

Main function for CUSH2 models with covariates for both shelter choices.

Estimate and validate a CUSH2 model for given ordinal responses, with covariates for both shelter choices.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **sh1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **sh2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$
- **X1** (*DataFrame*) – dataframe of covariates for explaining the 1st shelter effect
- **X2** (*DataFrame*) – dataframe of covariates for explaining the 2nd shelter effect
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None

Returns
an instance of `CUBresCUSH2XX` (see the Class for details)

Return type
object

`cubmods.cush2_xx.pmf`(*m*, *sh1*, *sh2*, *omega1*, *omega2*, *X1*, *X2*)

Average probability distribution of a specified CUSH2 model with covariates for both shelter choices.

$\frac{1}{n}\sum_{i=1}^{n}\Pr(R=r|\boldsymbol{\theta}_i;\boldsymbol{x}_{1i};;\boldsymbol{x}_{2i}),\ r=1\ldots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **sh1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **sh2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$
- **omega1** (*array*) – array $\boldsymbol{\omega}_1$ of parameters for the 1st shelter effect, whose length equals `X1.columns.size+1` to include an intercept term in the model (first entry)
- **omega2** (*array*) – array $\boldsymbol{\omega}_2$ of parameters for the 2nd shelter effect, whose length equals `X2.columns.size+1` to include an intercept term in the model (first entry)
- **X1** (*DataFrame*) – dataframe of covariates for explaining the 1st shelter effect

- **X2** (*DataFrame*) – dataframe of covariates for explaining the 2nd shelter effect

Returns
the average probability distribution

Return type
array

`cubmods.cush2_xx.pmfi`(*m*, *sh1*, *sh2*, *omega1*, *omega2*, *X1*, *X2*)

Probability distribution for each subject of a specified CUSH2 model with covariates for both shelter choices.

Auxiliary function of `.draw()`.

$\Pr(R = r|\boldsymbol{\theta}_i; \boldsymbol{x}_{1i}; \boldsymbol{x}_{2i}),\ i = 1 \dots n,\ r = 1 \dots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **sh1** (*int*) – Category corresponding to the 1st shelter choice $[1, m]$
- **sh2** (*int*) – Category corresponding to the 2nd shelter choice $[1, m]$
- **omega1** (*array*) – array $\boldsymbol{\omega}_1$ of parameters for the 1st shelter effect, whose length equals `X1.columns.size+1` to include an intercept term in the model (first entry)
- **omega2** (*array*) – array $\boldsymbol{\omega}_2$ of parameters for the 2nd shelter effect, whose length equals `X2.columns.size+1` to include an intercept term in the model (first entry)
- **X1** (*DataFrame*) – dataframe of covariates for explaining the 1st shelter effect
- **X2** (*DataFrame*) – dataframe of covariates for explaining the 2nd shelter effect

Returns
the matrix of the probability distribution of dimension $n \times r$

Return type
numpy ndarray

3.1.15 cubmods.cush_x module

CUB models in Python. Module for CUSH (Combination of Uniform and Shelter effect) with covariates.

Description:

This module contains methods and classes for CUSH model family. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

$\Pr(R_i = r|\boldsymbol{\theta}; \boldsymbol{x}_i) = \delta_i D_r^{(c)} + (1 - \delta_i)/m$

References:

1. Stefania Capecchi and Domenico Piccolo. Dealing with heterogeneity in ordinal responses. *Quality & Quantity*, 51:2375–2393, 2017.
2. Giovanni Cerulli, R Simone, Francesca Di Iorio, D Piccolo, CF Baum, and others. The stata module for cub models for rating data analysis. In *London Stata Conference 2021*, number 16. Stata Users Group, 2021.
3. Maria Iannario, Domenico Piccolo, and Maintainer Rosaria Simone. Package 'cub'. *CRAN*, 2022.
4. Domenico Piccolo and others. On the moments of a mixture of uniform and shifted binomial random variables. *Quaderni di Statistica*, 5(1):85–104, 2003.
5. Domenico Piccolo and others. Observed information matrix for mub models. *Quaderni di Statistica*, 8(1):33–78, 2006.
6. Domenico Piccolo and Rosaria Simone. The class of cub models: statistical foundations, inferential issues and empirical evidence. *Statistical Methods & Applications*, 28:389–435, 2019.
7. Massimo Pierini. Modelli della classe cub in python. *Universitas Mercatorum, Rome, IT*, pages 16–20, June 2024. (Bachelor's thesis L-41).
8. W Stephen Pittard and Shuzhao Li. The essential toolbox of data science: python, r, git, and docker. *Computational Methods and Data Analysis for Metabolomics*, pages 265–311, 2020.
9. Rosaria Simone, Francesca Di Iorio, and Riccardo Lucchetti. Cub for gretl. *GNU Regression, Econometrics and Time Series Library*, pages 147, 2019.

List of TODOs:

- …

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.cush_x.**CUBresCUSHX**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([saveas, figsize])	Main function to plot an object of the Class.
plot_ordinal([figsize, ax, kind, saveas])	Plots avreage relative frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*saveas=None*, *figsize=(7, 5)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns

`ax` or a tuple `(fig, ax)`

plot_ordinal(*figsize=(7, 5)*, *ax=None*, *kind='bar'*, *saveas=None*)

Plots avreage relative frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns

`ax` or a tuple `(fig, ax)`

cubmods.cush_x.**draw**(*m*, *sh*, *omega*, *X*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified CUSH model with covariates

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **omega** (*array*) – array ω of parameters for the shelter effect, whose length equals **X.columns.size+1** to include an intercept term in the model (first entry)
- **X** (*pandas dataframe*) – dataframe of covariates for explaining the shelter effect
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns

an instance of CUBsample containing ordinal responses drawn from the specified model

cubmods.cush_x.**effe**(*pars*, *esterno*, *m*, *sh*)

Auxiliary function for the log-likelihood estimation of CUSH models with covariates

Compute the opposite of the loglikelihood function for CUSH models with covariates to explain the shelter effect. It is called as an argument for "optim" within .mle() function as the function to minimize.

Parameters

- **pars** (*array*) – array of the initial parameters estimates
- **esterno** (*ndarray*) – matrix binding together the vector of ordinal data and the matrix **XX** of explanatory variables whose first column is a column of ones needed to consider an intercept term
- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$

cubmods.cush_x.**loglik**(*m*, *sample*, *X*, *omega*, *sh*)

Log-likelihood function for CUSH models with covariates.

Compute the log-likelihood function for CUSH models with covariates to explain the shelter effect.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **omega** (*array*) – array ω of parameters for the shelter effect, whose length equals **X.columns.size+1** to include an intercept term in the model (first entry)
- **X** (*pandas dataframe*) – dataframe of covariates for explaining the shelter effect

Returns

the log-likelihood value

Return type

float

cubmods.cush_x.**mle**(*m*, *sample*, *X*, *sh*, *df*, *formula*, *gen_pars=None*, *maxiter=None*, *tol=None*)

Main function for CUSH models with covariates.

Estimate and validate a CUSH model for ordinal responses, with covariates to explain the shelter effect.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **X** (*pandas dataframe*) – dataframe of covariates for explaining the shelter effect
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None
- **maxiter** (*None*) – default to None; ensure compatibility with `gem.from_formula()`
- **tol** (*None*) – default to None; ensure compatibility with `gem.from_formula()`

Returns
an instance of `CUBresCUSHX` (see the Class for details)

Return type
object

cubmods.cush_x.**pmf**(*m*, *sh*, *omega*, *X*)

Average probability distribution of a specified CUSH model with covariates.

$\frac{1}{n}\sum_{i=1}^{n}\Pr(R=r|\boldsymbol{\theta}_i\boldsymbol{x}_i),\ r=1\ldots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **omega** (*array*) – array $\boldsymbol{\omega}$ of parameters for the shelter effect, whose length equals `X.columns.size+1` to include an intercept term in the model (first entry)
- **X** (*pandas dataframe*) – dataframe of covariates for explaining the shelter effect

Returns
the probability distribution

Return type
array

cubmods.cush_x.**pmfi**(*m*, *sh*, *omega*, *X*)

Probability distribution for each subject of a specified CUSH model with covariates

$\Pr(R=r|\boldsymbol{\theta}_i;\boldsymbol{x}_i),\ i=1\ldots n,\ r=1\ldots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **omega** (*array*) – array $\boldsymbol{\omega}$ of parameters for the shelter effect, whose length equals `X.columns.size+1` to include an intercept term in the model (first entry)

- **X** (*pandas dataframe*) – dataframe of covariates for explaining the shelter effect

Returns
the matrix of the probability distribution of dimension $n \times r$

Return type
numpy ndarray

cubmods.cush_x.**proba**(*m*, *sample*, *X*, *omega*, *sh*)

Probability $\Pr(R = r|\boldsymbol{\theta})$ of a specified CUSH model with covariates.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **sh** (*int*) – Category corresponding to the shelter choice $[1, m]$
- **omega** (*array*) – array $\boldsymbol{\omega}$ of parameters for the shelter effect, whose length equals **X.columns.size**+1 to include an intercept term in the model (first entry)
- **X** (*pandas dataframe*) – dataframe of covariates for explaining the shelter effect

Returns
the probability array $\Pr(R = r|\boldsymbol{\theta})$ for observed responses

Return type
float

3.1.16 cubmods.gem module

CUB models in Python. Module for GEM (Generalized Mixtures).

Description:

This module contains methods and classes for GEM maximum likelihood estimation and sample drawing.
It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

References:

- D'Elia A. (2003). Modelling ranks using the inverse hypergeometric distribution, Statistical Modelling: an International Journal, 3, 65–78
- D'Elia A. and Piccolo D. (2005). A mixture model for preferences data analysis, Computational Statistics & Data Analysis}, old{49, 917–937
- Capecchi S. and Piccolo D. (2017). Dealing with heterogeneity in ordinal responses, Quality and Quantity, 51(5), 2375–2393
- Iannario M. (2014). Modelling Uncertainty and Overdispersion in Ordinal Data, Communications in Statistics - Theory and Methods, 43, 771–786
- Piccolo D. (2015). Inferential issues for CUBE models with covariates, Communications in Statistics. Theory and Methods, 44(23), 771–786.
- Iannario M. (2015). Detecting latent components in ordinal data with overdispersion by means of a mixture distribution, Quality & Quantity, 49, 977–987

- Iannario M. and Piccolo D. (2016a). A comprehensive framework for regression models of ordinal data. Metron, 74(2), 233–252.
- Iannario M. and Piccolo D. (2016b). A generalized framework for modelling ordinal data. Statistical Methods and Applications, 25, 163–189.

List of TODOs:

- TODO: implement best shelter search

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

`cubmods.gem.draw`(*formula*, *df=None*, *m=7*, *model='cub'*, *n=500*, *sh=None*, *seed=None*, ***params*)

Main function to draw a sample from GEneralized Mixture models.

Parameters

- **formula** (*str*) – a formula used to draw the sample, see Manual for details
- **df** (*DataFrame*) – the DataFrame with covariates (if any)
- **m** (*int*) – number of ordinal categories
- **model** (*str*) – the model family; default to "cub"; options "cube" and "cush"
- **sh** (*int*) – category corresponding to the shelter choice $[1, m]$
- **n** (*int*) – number of ordinal responses; it is only effective if the model is without covariates
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None
- **options** (*dict*) – a dictionary of extra options maxiter and tol; see the reference guide for details
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns

an instance of CUBsample containing ordinal responses drawn from the specified model

Return type
obj

cubmods.gem.**estimate**(*formula*, *df*, *m=None*, *model='cub'*, *sh=None*, *gen_pars=None*, *options={}*)

Main function to estimate and validate GEneralized Mixture models.

Parameters

- **formula** (*str*) – a formula used to estimate the model's parameters, see Manual for details
- **df** (*DataFrame*) – the DataFrame with observed ordinal sample and covariates (if any)
- **m** (*int*) – number of ordinal categories
- **model** (*str*) – the model family; default to "cub"; options "cube" and "cush"
- **sh** (*int*) – category corresponding to the shelter choice $[1, m]$
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None
- **options** (*dict*) – a dictionary of extra options maxiter and tol; see the reference guide for details

Returns
an instance of the Base Class CUBres extended by the family module; see each module for details

Return type
obj

3.1.17 cubmods.general module

CUB models in Python. Module for General functions.

Description:

This module contains methods and classes for general functions. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

References:

- D'Elia A. (2003). Modelling ranks using the inverse hypergeometric distribution, Statistical Modelling: an International Journal, 3, 65–78
- D'Elia A. and Piccolo D. (2005). A mixture model for preferences data analysis, Computational Statistics & Data Analysis}, old{49, 917–937
- Capecchi S. and Piccolo D. (2017). Dealing with heterogeneity in ordinal responses, Quality and Quantity, 51(5), 2375–2393
- Iannario M. (2014). Modelling Uncertainty and Overdispersion in Ordinal Data, Communications in Statistics - Theory and Methods, 43, 771–786
- Piccolo D. (2015). Inferential issues for CUBE models with covariates, Communications in Statistics. Theory and Methods, 44(23), 771–786.
- Iannario M. (2015). Detecting latent components in ordinal data with overdispersion by means of a mixture distribution, Quality & Quantity, 49, 977–987

- Iannario M. and Piccolo D. (2016a). A comprehensive framework for regression models of ordinal data. Metron, 74(2), 233–252.
- Iannario M. and Piccolo D. (2016b). A generalized framework for modelling ordinal data. Statistical Methods and Applications, 25, 163–189.

List of TODOs:

- ...

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

exception cubmods.general.**InvalidCategoriesError**(*m*, *model*)

Bases: Exception

Exception: if m is not suitable for model.

exception cubmods.general.**InvalidSampleSizeError**(*n*)

Bases: Exception

Exception: if the sample size is not strictly greater than zero.

exception cubmods.general.**NoShelterError**(*model*)

Bases: Exception

Exception: if a shelter choice is needed but it hasn't been provided.

exception cubmods.general.**NotImplementedModelError**(*model*, *formula*)

Bases: Exception

Exception: if the requested model is known but not yet implemented.

exception cubmods.general.**ParameterOutOfBoundsError**(*param*, *value*)

Bases: Exception

Exception: if the provided parameter value is out of bounds.

exception cubmods.general.**ShelterGreaterThanM**(*m*, *sh*)

Bases: Exception

Exception: if the provided shelter choice is greater than m.

exception cubmods.general.**UnknownModelError**(*model*)

Bases: Exception

Exception: if the requested family is unknown.

cubmods.general.**addones**(*A*)

Expand with a unitary vector in the first column of the given matrix to consider also an intercept term for CUB models with covariates.

Parameters
A – a matrix to be expanded

Returns
the expanded matrix

Return type
same of A

cubmods.general.**aic**(*l*, *p*)

Akaike Information Criterion.

Parameters

- **l** (*float*) – log-likelihood
- **p** (*int*) – number of parameters

Returns
the AIC value

Return type
float

cubmods.general.**bic**(*l*, *p*, *n*)

Bayesian Information Criterion.

Parameters

- **l** (*float*) – log-likelihood
- **p** (*int*) – number of parameters
- **n** (*int*) – number of observations

Returns
the BIC value

Return type
float

cubmods.general.**bitgamma**(*sample*, *m*, *W*, *gamma*)

Shifted Binomial distribution with covariates.

Return the shifted Binomial probabilities of ordinal responses where the feeling component is explained by covariates via a logistic link.

Parameters

- **sample** (*array*) – array of ordinal responses

- **m** (*int*) – number of ordinal categories
- **W** (*pandas dataframe*) – dataframe of covariates for explaining the feeling component
- **gamma** (*array of float*) – array γ of parameters for the feeling component, whose length equals `W.columns.size+1` to include an intercept term in the model (first entry)

Returns
an array of the same length as `sample`, where each entry is the shifted Binomial probability for the corresponding observation and feeling value.

Return type
array

`cubmods.general.`**bitxi**(*m*, *sample*, *xi*)

Shifted Binomial probabilities of ordinal responses

Compute the shifted Binomial probabilities of ordinal responses.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array*) – array of ordinal responses
- **xi** (*float*) – feeling parameter ξ

Returns
A vector of the same length as `sample`, where each entry is the shifted Binomial probability of the corresponding observation.

Return type
array

`cubmods.general.`**choices**(*m*)

Array of ordinal categories.

Parameters
m (*int*) – number of ordinal categories

Returns
array of int from 1 to m

Return type
array

`cubmods.general.`**colsof**(*A*)

Number of columns of the given matrix or dataframe.

Parameters
A (*ndarray, dataframe*) – the matrix or dataframe

Returns
number of columns

Return type
int

`cubmods.general.`**conf_border**(*Sigma*, *mx*, *my*, *ax*, *conf=0.95*, *plane='z'*, *xyz0=(0, 0, 0)*)

Plot the bivariate projection of a trivariate confidence ellipse on a plane.

Auxiliary function of `plot_ellipsoid()`.

Note: Solution by https://gist.github.com/randolf-scholz.

Parameters

- **Sigma** (*ndarray*) – bivariate variance-covariance matrix
- **mx** (*float*) – center of the ellipse on the x axies
- **my** (*float*) – center of the ellipse on the y axies
- **ax** – matpplotlib axis
- **conf** (*float*) – confidence level of the trivariate ellipsoid.
- **plane** (*str*) – plane for the projection; could be `x`, `y` or `z`
- **xyz0** (*tuple*) – tuple of the bivariate ellipse position

cubmods.general.**conf_ell**(*vcov*, *mux*, *muy*, *ci*, *ax*, *color='b'*, *label=True*, *alpha=0.25*)

Plot bivariate confidence ellipse of estimated parameters at level `ci`$= (1 - \alpha/2)$

Parameters

- **vcov** (*ndarray*) – Variance-covariance matrix 2×2
- **mux** (*float*) – estimate of first parameter
- **muy** (*float*) – estimate of second parameter
- **ci** (*float*) – confidence level $= (1 - \alpha/2)$
- **ax** – matplotlib axis
- **color** (*str*) – color of confidence ellipse
- **label** (*bool*) – whether to add a label of confidence level
- **alpha** (*float*) – transparency of confidence ellipse

cubmods.general.**dissimilarity**(*p_obs*, *p_est*)

Normalized dissimilarity measure.

Compute the normalized dissimilarity measure between observed relative frequencies and estimated (theoretical) probabilities of a discrete distribution.

Parameters

- **p_obs** (*array*) – Vector of observed relative frequencies
- **p_est** (*array*) – Vector of estimated (theoretical) probabilities

Returns
Numeric value of the dissimilarity index, assessing the distance to a perfect fit.

Return type
float

cubmods.general.**dummies2**(*df*, *DD*)

Create dummy variables from polychotomous variables.

Auxiliary function of `cubmods.gem.from_formula()`. A dummy variable is created for all polychotomous variables named `C(<varname>)`.

Parameters

- **df** (*DataFrame*) – a DataFrame with all the covariates and the ordinal response
- **DD** (*list*) – the list of all covariates for each component

Returns
a tuple of the DataFrame with the dummy variables and the column names

Return type
tuple

cubmods.general.**equal3d**(*ax*)

Equalize 3d axes.

Auxiliary function of `.plot_ellipsoid()`.

cubmods.general.**expit**(*x*)

Expit function.

It is the inverse of logit. Aka sigmoid or standard logistic.

Parameters
x (*float*) – the argument

Returns
the expit of x

Return type
float

cubmods.general.**formula_parser**(*formula*, *model='cub'*)

Parse a CUB class formula.

Auxiliary function of `cubmods.gem` functions.

TODO: add specific Exceptions for formula

Parameters

- **formula** (*str*) – the formula to be parsed
- **model** (*str*) – the model family

Returns
a tuple of the ordinal response column name and a list of all covariates' column names for each component

Return type
tuple

cubmods.general.**freq**(*sample*, *m*, *dataframe=False*)

Absolute frequecies of an observed sample of ordinal responses.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **dataframe** (*bool*) – if `True` return a DataFrame instead of an array, defaults to `False`

Returns
the absolute frequencies of the observed sample

Return type
array or dataframe

cubmods.general.**get_cov_ellipsoid**(*cov*, *mu=array([0., 0., 0.])*, *ci=0.95*)

Return the 3d points representing the covariance matrix `cov` centred at `mu`, at confidence level `ci`$= (1 - \alpha/2)$.

Auxiliary function of `.plot_ellipsoid()`.

Parameters

- **cov** (*ndarray*) – Variance-covariance matrix 3×3
- **mu** (*array*) – ellispoid center (x_0, y_0, z_0)
- **ci** (*float*) – confidence level $= (1 - \alpha/2)$

Returns
a tuple of 3d points `(X, Y, Z)`

Return type
tuple

cubmods.general.**get_minor**(*A*, *i*, *j*)

Get a minor of a matrix.

Auxiliary function of `.plot_ellipsoid()`.

Note: Solution by PaulDong

Parameters

- **A** (*ndarray*) – a generic matrix
- **i** (*int*) – row of the minor
- **j** (*int*) – column of the minor

Returns
the minor of `A`

Return type
ndarray

cubmods.general.**hadprod**(*Amat*, *xvett*)

Hadamard product of a matrix with a vector

Return the Hadamard product between the given matrix and vector: this operation corresponds to multiply every row of the matrix by the corresponding element of the vector, and it is equivalent to the standard matrix multiplication to the right with the diagonal matrix whose diagonal is the given vector. It is possible only if the length of the vector equals the number of rows of the matrix. It is an auxiliary function needed for computing the variance-covariance matrix of the estimated model with covariates.

Note: if `xvett` is a row vector, reshapes it to column vector

Parameters

- **Amat** (*ndarray*) – A generic matrix
- **xvett** (*array*) – A generic vector

Returns
the Hadamard product $\boldsymbol{A} \odot \boldsymbol{x}$

Return type
ndarray

cubmods.general.**kkk**(*sample*, *m*)

Sequence of combinatorial coefficients

Compute the sequence of binomial coefficients $\binom{m-1}{r-1}$, for $r = 1, \ldots m$, and then returns a vector of the same length as ordinal, whose i-th component is the corresponding binomial coefficient $\binom{m-1}{r_i-1}$

Parameters

- **sample** (*array*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories

Returns
an array of $\binom{m-1}{r_i-1}$

Return type
array

cubmods.general.**load_object**(*fname*)

Load a saved object from file.

It can used be used to load a `CUBsample` or a `CUBres` object, previously saved on a file.

Note: see the Classes for details about these objects

Parameters
fname (*str*) – filename

Returns
the loaded object, instance of `CUBsample` or `CUBres`

Return type
object

cubmods.general.**logis**(*Y*, *param*)

The logistic transform.

Create a matrix `YY` binding array `Y` with a vector of ones, placed as the first column of `YY`. It applies the logistic transform componentwise to the standard matrix multiplication between `YY` and `param`.

Parameters

- **Y** (*ndarray, dataframe*) – A generic matrix or a dataframe
- **param** (*array*) – Vector of coefficients, whose length is `Y.columns.size+1` (to consider also an intercept term)

Returns
a vector whose length is `Y.index.size` and whose i-th component is the logistic function

cubmods.general.**logit**(*x*)

Logit function.

It is the inverse of the standard logistic function, aka log-odds.

Parameters
x (*float*) – the argument

Returns
the logit of x

Return type
float

cubmods.general.**lsat**(*f*, *n*)

Log-likelihood of saturated model.

Ssaturated level,that is the theoretically maximum information that can be obtained by a model using as many parameters as possible. Then, the saturated log-likelihood is computed by assuming that the model is specified by as many parameters as available observations. This is the extreme benchmark for comparing previous log-likelihood quantities.

Parameters

- **f** (*array*) – absolute frequencies of observed ordinal responses
- **n** (*int*) – number of observations

Returns
log-likelihood of saturated model

Return type
float

cubmods.general.**luni**(*m*, *n*)

Log-likelihood of null model.

Null level, that is when no structure is searched for. Specifically, this is equivalent to assume a discrete Uniform over the support so that any category has the same probability.

Parameters

- **m** (*int*) – number of ordinal categories
- **n** (*int*) – number of observations

Returns
the log-likelihood of null model

Return type
float

cubmods.general.**plot_ellipsoid**(*V*, *E*, *ax*, *zlabel*, *ci=0.95*, *magnified=False*)

Plot a trivariate confidence ellipsoid.

Parameters

- **V** (*ndarray*) – Variance-covariance matrix
- **E** (*array*) – Vector of estimated parameters
- **ax** – matplotlib axis
- **zlabel** (*str*) – label for z axis
- **ci** (*float*) – confidence level $(1 - \alpha/2)$
- **magnified** (*bool*) – if `False` plots in the full parameter space

cubmods.general.**probbit**(*m*, *xi*)

Probability distribution of shifted binomial random variable.

Parameters

- **m** (*int*) – number of ordinal categories
- **xi** (*float*) – feeling parameter ξ

Returns
the vector of the probability distribution of a shifted Binomial model.

Return type
array

cubmods.general.**unique**(*l*)

Unique elements in a 3-dimensional list.

Auxiliary function of `.dummies2()`.

Parameters
l (*list*) – the list to analyze

Returns
the list of unique elements

Return type
list

3.1.18 cubmods.ihg module

CUB models in Python. Module for IHG (Inverse HyperGeometric).

Description:

This module contains methods and classes for IHG model family without covariates. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

References:

- D'Elia A. (2003). Modelling ranks using the inverse hypergeometric distribution, Statistical Modelling: an International Journal, 3, 65–78
- D'Elia A. and Piccolo D. (2005). A mixture model for preferences data analysis, Computational Statistics & Data Analysis}, old{49, 917–937
- Capecchi S. and Piccolo D. (2017). Dealing with heterogeneity in ordinal responses, Quality and Quantity, 51(5), 2375–2393
- Iannario M. (2014). Modelling Uncertainty and Overdispersion in Ordinal Data, Communications in Statistics - Theory and Methods, 43, 771–786
- Piccolo D. (2015). Inferential issues for CUBE models with covariates, Communications in Statistics. Theory and Methods, 44(23), 771–786.
- Iannario M. (2015). Detecting latent components in ordinal data with overdispersion by means of a mixture distribution, Quality & Quantity, 49, 977–987
- Iannario M. and Piccolo D. (2016a). A comprehensive framework for regression models of ordinal data. Metron, 74(2), 233–252.
- Iannario M. and Piccolo D. (2016b). A generalized framework for modelling ordinal data. Statistical Methods and Applications, 25, 163–189.

List of TODOs:

- ...

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.ihg.**CUBresIHG**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Object returned by `.mle()` function. See the Base for details.

Methods

`as_dataframe`()	DataFrame of estimated parameters
`as_txt`()	Print the summary.
plot([ci, saveas, figsize])	Main function to plot an object of the Class.
plot_estim([ci, ax, magnified])	Plots the estimated parameter values in the parameter space and the asymptotic standard error.
plot_ordinal([figsize, ax, kind, saveas])	Plots relative frequencies of observed sample, estimated probability distribution and, if provided, probability distribution of a known model.
`save`(fname)	Save a CUBresult object to file named `fname + .cub.fit`
`summary`()	Call `as_txt()`

plot(*ci=0.95*, *saveas=None*, *figsize=(7, 15)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **ci** (*float*) – level $(1 - \alpha/2)$ for the standard error
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

plot_estim(*ci=0.95*, *ax=None*, *magnified=False*)

Plots the estimated parameter values in the parameter space and the asymptotic standard error.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **ci** (*float*) – level $(1 - \alpha/2)$ for the confidence ellipse
- **magnified** (*bool*) – if False the limits will be the entire parameter space, otherwise let matplotlib choose the limits
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None

Returns
`ax` or a tuple `(fig, ax)`

plot_ordinal(*figsize=(7, 5)*, *ax=None*, *kind='bar'*, *saveas=None*)

Plots relative frequencies of observed sample, estimated probability distribution and, if provided, probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

cubmods.ihg.**draw**(*m*, *theta*, *n*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified IHG model.

Parameters

- **m** (*int*) – number of ordinal categories
- **theta** (*float*) – parameter θ (probability of 1st shelter category)
- **n** (*int*) – number of ordinal responses to be drawn
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns
an instance of `CUBsample` containing ordinal responses drawn from the specified model

`cubmods.ihg.effe(theta, m, f)`

Compute the log-likelihood function of a IHG model without covariates for a given absolute frequency distribution.

Parameters

- **theta** (*float*) – parameter θ (probability of 1st shelter category)
- **m** (*int*) – number of ordinal categories
- **f** (*array of int*) – array of absolute frequency distribution

Returns
the log-likelihood value

Return type
float

`cubmods.ihg.init_theta(m, f)`

Preliminary estimators for IHG models without covariates.

Computes preliminary parameter estimates of a IHG model without covariates for given ordinal responses. These preliminary estimators are used within the package code to start the E-M algorithm.

Parameters

- **f** (*array of int*) – array of the absolute frequencies of given ordinal responses
- **m** (*int*) – number of ordinal categories

Returns
the value of $\theta^{(0)}$

`cubmods.ihg.loglik(m, theta, f)`

`cubmods.ihg.mle(m, sample, df, formula, gen_pars=None)`

Main function for CUB models without covariates.

Function to estimate and validate a CUB model without covariates for given ordinal responses.

Parameters

- **sample** (*array of int*) – array of ordinal responses
- **m** (*int*) – number of ordinal categories
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None

Returns
an instance of `CUBresIHG` (see the Class for details)

Return type
object

`cubmods.ihg.pmf(m, theta)`

Probability distribution of a specified IHG model without covariates.

$\Pr(R = r|\boldsymbol{\theta}),\ r = 1 \dots m$

References:

1. Giovanni Cerulli, R Simone, Francesca Di Iorio, D Piccolo, CF Baum, and others. The stata module for cub models for rating data analysis. In *London Stata Conference 2021*, number 16. Stata Users Group, 2021.
2. Maria Iannario, Domenico Piccolo, and Maintainer Rosaria Simone. Package 'cub'. *CRAN*, 2022.
3. Domenico Piccolo and others. On the moments of a mixture of uniform and shifted binomial random variables. *Quaderni di Statistica*, 5(1):85–104, 2003.
4. Domenico Piccolo and others. Observed information matrix for mub models. *Quaderni di Statistica*, 8(1):33–78, 2006.
5. Domenico Piccolo and Rosaria Simone. The class of cub models: statistical foundations, inferential issues and empirical evidence. *Statistical Methods & Applications*, 28:389–435, 2019.
6. Massimo Pierini. Modelli della classe cub in python. *Universitas Mercatorum, Rome, IT*, pages 16–20, June 2024. (Bachelor's thesis L-41).
7. W Stephen Pittard and Shuzhao Li. The essential toolbox of data science: python, r, git, and docker. *Computational Methods and Data Analysis for Metabolomics*, pages 265–311, 2020.
8. Rosaria Simone, Francesca Di Iorio, and Riccardo Lucchetti. Cub for gretl. *GNU Regression, Econometrics and Time Series Library*, pages 147, 2019.

Parameters

- **m** (*int*) – number of ordinal categories
- **theta** (*float*) – parameter θ (probability of 1st shelter category)

Returns
the vector of the probability distribution of a CUB model.

Return type
numpy array

cubmods.ihg.**var**(*m*, *theta*)

Variance of a specified IHG model.

Parameters

- **m** (*int*) – number of ordinal categories
- **theta** (*float*) – parameter θ (probability of 1st shelter category)

Returns
the variance of the model

Return type
float

3.1.19 cubmods.ihg_v module

CUB models in Python. Module for IHG (Inverse HyperGeometric) with covariates.

Description:

This module contains methods and classes for IHG model family with covariates. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

References:

- TODO: aggiungere tesi?
- D'Elia A. (2003). Modelling ranks using the inverse hypergeometric distribution, Statistical Modelling: an International Journal, 3, 65–78
- D'Elia A. and Piccolo D. (2005). A mixture model for preferences data analysis, Computational Statistics & Data Analysis}, old{49, 917–937
- Capecchi S. and Piccolo D. (2017). Dealing with heterogeneity in ordinal responses, Quality and Quantity, 51(5), 2375–2393
- Iannario M. (2014). Modelling Uncertainty and Overdispersion in Ordinal Data, Communications in Statistics - Theory and Methods, 43, 771–786
- Piccolo D. (2015). Inferential issues for CUBE models with covariates, Communications in Statistics. Theory and Methods, 44(23), 771–786.
- Iannario M. (2015). Detecting latent components in ordinal data with overdispersion by means of a mixture distribution, Quality & Quantity, 49, 977–987
- Iannario M. and Piccolo D. (2016a). A comprehensive framework for regression models of ordinal data. Metron, 74(2), 233–252.
- Iannario M. and Piccolo D. (2016b). A generalized framework for modelling ordinal data. Statistical Methods and Applications, 25, 163–189.

List of TODOs:

- ...

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.ihg_v.**CUBresIHGV**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: *CUBres*

Methods

as_dataframe()	DataFrame of estimated parameters
as_txt()	Print the summary.
plot([saveas, figsize])	Main function to plot an object of the Class.
plot_ordinal([figsize, ax, kind, saveas])	Plots avreage relative frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.
save(fname)	Save a CUBresult object to file named `fname + .cub.fit`
summary()	Call `as_txt()`

plot(*saveas=None*, *figsize=(7, 5)*)

Main function to plot an object of the Class.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

plot_ordinal(*figsize=(7, 5)*, *ax=None*, *kind='bar'*, *saveas=None*)

Plots avreage relative frequencies of observed sample, estimated average probability distribution and, if provided, average probability distribution of a known model.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None
- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

cubmods.ihg_v.**draw**(*m*, *nu*, *V*, *df*, *formula*, *seed=None*)

Draw a random sample from a specified IHG model with covariates

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **nu** (*array*) – array $\boldsymbol{\nu}$ of parameters for θ, whose length equals `V.columns.size+1` to include an intercept term in the model (first entry)
- **V** (*pandas dataframe*) – dataframe of covariates for explaining the parameter θ
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **seed** (*int, optional*) – the *seed* to ensure reproducibility, defaults to None

Returns

an instance of `CUBsample` containing ordinal responses drawn from the specified model

cubmods.ihg_v.**effe**(*nu*, *m*, *sample*, *V*)

Auxiliary function for the log-likelihood estimation of IHG models with covariates

Compute the opposite of the loglikelihood function for IHG models with covariates. It is called as an argument for "optim" within `.mle()` function as the function to minimize.

Parameters

- **nu** (*float*) – initial parameter estimate
- **V** (*pandas dataframe*) – dataframe of covariates for explaining the parameter θ
- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses

cubmods.ihg_v.**init_theta**(*m*, *f*)

Preliminary estimators for IHG models without covariates.

Computes preliminary parameter estimates of a IHG model without covariates for given ordinal responses. These preliminary estimators are used within the package code to start the E-M algorithm.

Parameters

- **f** (*array of int*) – array of the absolute frequencies of given ordinal responses
- **m** (*int*) – number of ordinal categories

Returns

the array of $\boldsymbol{\nu}^{(0)}$

cubmods.ihg_v.**loglik**(*m*, *sample*, *V*, *nu*)

Log-likelihood function for IHG models with covariates.

Compute the log-likelihood function for CUSH models with covariates to explain the shelter effect.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **nu** (*array*) – array $\boldsymbol{\nu}$ of parameters for θ, whose length equals `V.columns.size+1` to include an intercept term in the model (first entry)

- **V** (*pandas dataframe*) – dataframe of covariates for explaining the parameter θ

Returns
the log-likelihood value

Return type
float

cubmods.ihg_v.**mle**(*m*, *sample*, *V*, *df*, *formula*, *gen_pars=None*)

Main function for IHG models with covariates.

Estimate and validate a IHG model for ordinal responses, with covariates.

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **V** (*pandas dataframe*) – dataframe of covariates for explaining the parameter θ
- **df** (*DataFrame*) – original DataFrame
- **formula** (*str*) – the formula used
- **gen_pars** (*dictionary, optional*) – dictionary of hypothesized parameters, defaults to None

Returns
an instance of `CUBresIHGV` (see the Class for details)

Return type
object

cubmods.ihg_v.**pmf**(*m*, *V*, *nu*)

Average probability distribution of a specified IHG model with covariates.

$\frac{1}{n}\sum_{i=1}^{n}\Pr(R=r|\boldsymbol{\theta}_i\boldsymbol{v}_i),\ r=1\ldots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **nu** (*array*) – array $\boldsymbol{\nu}$ of parameters for θ, whose length equals `V.columns.size+1` to include an intercept term in the model (first entry)
- **V** (*pandas dataframe*) – dataframe of covariates for explaining the parameter θ

Returns
the probability distribution

Return type
array

cubmods.ihg_v.**pmfi**(*m*, *V*, *nu*)

Probability distribution for each subject of a specified IHG model with covariates

$\Pr(R=r|\boldsymbol{\theta}_i;\boldsymbol{v}_i),\ i=1\ldots n,\ r=1\ldots m$

Parameters

- **m** (*int*) – number of ordinal categories
- **nu** (*array*) – array $\boldsymbol{\nu}$ of parameters for θ, whose length equals `V.columns.size+1` to include an intercept term in the model (first entry)
- **V** (*pandas dataframe*) – dataframe of covariates for explaining the parameter θ

Returns
the matrix of the probability distribution of dimension $n \times r$

Return type
numpy ndarray

cubmods.ihg_v.**probi**(*m*, *sample*, *V*, *nu*)

Probability distribution of a IHG model with covariates given an observed sample.

Compute the probability distribution of a IHG model with covariates, given an observed sample.

$\Pr(R = r_i|\boldsymbol{\theta}_i; \boldsymbol{w}_i),\ i = 1 \dots n$

Parameters

- **m** (*int*) – number of ordinal categories
- **sample** (*array of int*) – array of ordinal responses
- **nu** (*array*) – array $\boldsymbol{\nu}$ of parameters for θ, whose length equals `V.columns.size+1` to include an intercept term in the model (first entry)
- **V** (*pandas dataframe*) – dataframe of covariates for explaining the parameter θ

Returns
the array of the probability distribution.

Return type
numpy array

3.1.20 cubmods.multicub module

CUB models in Python. Module for MULTICUB and MULTICUBE.

Description:

This module contains methods and classes for MULTICUB and MULTICUBE tool. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

References:

- D'Elia A. (2003). Modelling ranks using the inverse hypergeometric distribution, Statistical Modelling: an International Journal, 3, 65–78
- D'Elia A. and Piccolo D. (2005). A mixture model for preferences data analysis, Computational Statistics & Data Analysis}, old{49, 917–937
- Capecchi S. and Piccolo D. (2017). Dealing with heterogeneity in ordinal responses, Quality and Quantity, 51(5), 2375–2393
- Iannario M. (2014). Modelling Uncertainty and Overdispersion in Ordinal Data, Communications in Statistics - Theory and Methods, 43, 771–786
- Piccolo D. (2015). Inferential issues for CUBE models with covariates, Communications in Statistics. Theory and Methods, 44(23), 771–786.
- Iannario M. (2015). Detecting latent components in ordinal data with overdispersion by means of a mixture distribution, Quality & Quantity, 49, 977–987

- Iannario M. and Piccolo D. (2016a). A comprehensive framework for regression models of ordinal data. Metron, 74(2), 233–252.
- Iannario M. and Piccolo D. (2016b). A generalized framework for modelling ordinal data. Statistical Methods and Applications, 25, 163–189.

List of TODOs:

- ...

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

cubmods.multicub.**multi**(*ords*, *ms=None*, *model='cub'*, *title=None*, *labels=None*, *shs=None*, *plot=True*, *print_res=False*, *pos=None*, *xlim=(0, 1)*, *ylim=(0, 1)*, *equal=True*, *confell=True*, *alpha=0.2*, *ci=0.95*, *figsize=(7, 7)*, *ax=None*)

Joint plot of estimated CUB models in the parameter space

Return a plot of estimated CUB models represented as points in the parameter space.

Parameters

- **ords** (*list*) – list of arrays of observed ordinal responses
- **model** (*str*) – model; defaults to cub; options cube
- **title** (*str*) – title of the plot
- **labels** (*list*) – labels of the points
- **shs** (*int or list*) – shelter effect(s); can be an *int* if the same shelter effect is valid for all samples or a *list* to specify different shelter choices
- **plot** (*bool*) – if True (default) plot the results;
- **print_res** (*bool*) – if True print the results; defaults to False
- **pos** (*list*) – position of the δ or ϕ estimated values
- **xlim** (*tuple*) – x-axis limits
- **ylim** (*tuple*) – y-axis limits

- **equal** (*bool*) – if the plot must have equal aspect; defaults to `True`
- **alpha** (*float*) – confidence ellipse transparency
- **confell** (*bool*) – if `True` (default) plot confidence ellipse (for CUB model only)
- **ci** (*float*) – level $(1 - \alpha/2)$ for the confidence ellipse
- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None

Returns
`ax`

`cubmods.multicub.`**pos_kwargs**(*pos*)

Position of the δ or ϕ estimated values

```
    1
  8   2
7   @   3
  6   4
    5
```

Parameters
pos (*int*) – position (1..8)

Returns
a dictionary for `matplotlib`

Return type
dict

3.1.21 cubmods.smry module

CUB models in Python. Module for summary tools.

Description:

This module contains methods and classes for summary tools. It is based upon the works of Domenico Piccolo et Al. and CUB package in R.

References:

- D'Elia A. (2003). Modelling ranks using the inverse hypergeometric distribution, Statistical Modelling: an International Journal, 3, 65–78
- D'Elia A. and Piccolo D. (2005). A mixture model for preferences data analysis, Computational Statistics & Data Analysis}, bold{49, 917–937
- Capecchi S. and Piccolo D. (2017). Dealing with heterogeneity in ordinal responses, Quality and Quantity, 51(5), 2375–2393
- Iannario M. (2014). Modelling Uncertainty and Overdispersion in Ordinal Data, Communications in Statistics - Theory and Methods, 43, 771–786

- Piccolo D. (2015). Inferential issues for CUBE models with covariates, Communications in Statistics. Theory and Methods, 44(23), 771–786.
- Iannario M. (2015). Detecting latent components in ordinal data with overdispersion by means of a mixture distribution, Quality & Quantity, 49, 977–987
- Iannario M. and Piccolo D. (2016a). A comprehensive framework for regression models of ordinal data. Metron, 74(2), 233–252.
- Iannario M. and Piccolo D. (2016b). A generalized framework for modelling ordinal data. Statistical Methods and Applications, 25, 163–189.

List of TODOs:

- TODO: risultati inferenziali come DataFrame nel Manuale e negli esempi
- TODO: bounds opzionali in CUBE mle (anche CUBSH?)
- TODO: 2 decimali nei 3d plot?
- TODO: dissim in multicub plot (aggiungere opzione)
- TODO: grandezza punti phi in multicube

Credits

Author
Massimo Pierini

Institution
Universitas Mercatorum

Affiliation
Graduand in Statistics & Big Data (L41)

Date
2023-24

Credit
Domenico Piccolo, Rosaria Simone

Contacts
cub@maxpierini.it

Classes and Functions

class cubmods.smry.**CUBres**(*model*, *df*, *formula*, *m*, *n*, *sample*, *f*, *theoric*, *diss*, *est_names*, *estimates*, *e_types*, *varmat*, *stderrs*, *pval*, *wald*, *loglike*, *muloglik*, *loglikuni*, *AIC*, *BIC*, *seconds*, *time_exe*, *logliksat=None*, *dev=None*, *logliksatcov=None*, *niter=None*, *maxiter=None*, *tol=None*, *sh=None*, *rho=None*, *gen_pars=None*)

Bases: `object`

Default Class for MLE results; each model module extends this Class to an ad hoc Class with specific functions. An instance of the extended Class is returned by `.mle()` functions of model modules.

Methods

as_dataframe()	DataFrame of estimated parameters
as_txt()	Print the summary.
save(fname)	Save a CUBresult object to file named `fname + .cub.fit`
summary()	Call `as_txt()`

as_dataframe()

DataFrame of estimated parameters

as_txt()

Print the summary.

save(*fname*)

Save a CUBresult object to file named `fname + .cub.fit`

summary()

Call `as_txt()`

class cubmods.smry.**CUBsample**(*rv*, *m*, *pars*, *model*, *df*, *formula*, *diss*, *theoric*, *par_names*, *sh=None*, *seed=None*)

Bases: `object`

An instance of this Class is returned by `.draw()` functions. See the corresponding model's function for details.

Methods

as_dataframe()	The parameters' values specified.
plot([figsize, kind, ax, saveas])	Basic plot function.
save(fname)	Save a CUBsample object to file named `fname + cub.sample`
summary()	Print the summary of the drawn sample.

as_dataframe()

The parameters' values specified.

Returns
a DataFrame with parameters' names and values

Return type
DataFrame

plot(*figsize=(7, 5)*, *kind='bar'*, *ax=None*, *saveas=None*)

Basic plot function.

Parameters

- **figsize** (*tuple of float*) – tuple of `(length, height)` for the figure (useful only if `ax` is not None)
- **kind** (*str*) – choose a barplot (`'bar'` default) of a scatterplot (`'scatter'`)
- **ax** (*matplolib ax, optional*) – matplotlib axis, if None a new figure will be created, defaults to None

- **saveas** (*str*) – if provided, name of the file to save the plot

Returns
`ax` or a tuple `(fig, ax)`

save(*fname*)

Save a CUBsample object to file named `fname` + `cub.sample`

summary()

Print the summary of the drawn sample.

3.1.22 Module contents

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

BIBLIOGRAPHY

[CSDI+21] Giovanni Cerulli, R Simone, Francesca Di Iorio, D Piccolo, CF Baum, and others. The stata module for cub models for rating data analysis. In *London Stata Conference 2021*, number 16. Stata Users Group, 2021.

[IPS22] Maria Iannario, Domenico Piccolo, and Maintainer Rosaria Simone. Package 'cub'. *CRAN*, 2022.

[P+03] Domenico Piccolo and others. On the moments of a mixture of uniform and shifted binomial random variables. *Quaderni di Statistica*, 5(1):85–104, 2003.

[P+06] Domenico Piccolo and others. Observed information matrix for mub models. *Quaderni di Statistica*, 8(1):33–78, 2006.

[PS19] Domenico Piccolo and Rosaria Simone. The class of cub models: statistical foundations, inferential issues and empirical evidence. *Statistical Methods & Applications*, 28:389–435, 2019.

[Pie24] Massimo Pierini. Modelli della classe cub in python. *Universitas Mercatorum, Rome, IT*, pages 16–20, June 2024. (Bachelor's thesis L-41).

[PL20] W Stephen Pittard and Shuzhao Li. The essential toolbox of data science: python, r, git, and docker. *Computational Methods and Data Analysis for Metabolomics*, pages 265–311, 2020.

[SDIL19] Rosaria Simone, Francesca Di Iorio, and Riccardo Lucchetti. Cub for gretl. *GNU Regression, Econometrics and Time Series Library*, pages 147, 2019.

PYTHON MODULE INDEX

C