Package 'misclassifyr'

July 2, 2024

Title Estimation and inference for misclassification models.

Version 0.0.0.9000

Description This package provides tools for estimation and inference of simple misclassification models, as described in Mattheis (2024).

License `use_mit_license()`

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

LinkingTo Rcpp

Imports Rcpp

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

R topics documented:

`loglikelihood` *Returns the log likelihood of the data.*

Description

Longer description of what it does... It's assumed that`tab`, `J`, and `K` are defined in the environment.

Usage

```
loglikelihood(theta)
```

Arguments

theta	A numeric vector of length Jx(J^2+K) describing the joint distribution of the data.

Details

some details.

Value

the log likelihood of the data given theta, i.e. Pi and Delta.

Examples

```
## Not run:
some example code # Should return something

## End(Not run)
```

`misclassifyr` *misclassifyr*

Description

This package provides a menu of options for estimation and inference of misclassification models in which the analyst has access to two noisy measures, `Y1` and `Y2` of a latent outcome `Y*`, a correctly measured covariate `X`, and discrete controls `W`.

Usage

```
misclassifyr(
  tab,
  J,
  K,
  model_to_Pi,
  model_to_Delta,
  phi_0 = NA,
  psi_0 = NA,
  split_eta = NA,
  lambda_pos = NA,
  lambda_dd = NA,
  optim_maxit = 1000,
  optim_tol = 1e-09,
  optim_stepsize = NA,
  check_stability = F,
  stability_sd = 0.1,
  estimation_options = NA,
  formula = NA
)
```

Arguments

tab	A dataframe or a list of dataframes containing tabulated data or a list of tabulated data split by controls. The columns should be numeric with names Y1, Y2, X, and n where Y1 and Y2 take each value between 1 and J, X takes each value between 1 and K, and
J	An integer corresponding to the number of unique values of Y1 and Y2.
K	An integer less than or equal to J corresponding to the number of unique values of X.
model_to_Pi	A function mapping the parameters of a model for the joint distribution to the joint distribution \Pi
model_to_Delta	A function mapping the parameters of a model to the conditional distribution Y1, Y2 | Y*, \Delta
phi_0	A numeric vector providing the starting location for optimization for the argument to model_to_Pi.
psi_0	A numeric vector providing the starting location for optimization for the argument to model_to_Delta.
split_eta	An integer indicating where to split the vector eta in phi and psi, the arguments to model_to_Pi and model_to_Delta respectively.
lambda_pos	scales the penalty for violations of positivity (i.e. all probabilities should be positive).
lambda_dd	scales the penalty for violations of diagonal dominance.
optim_maxit	An integer for the maximum number of iterations in numerical optimization, passed to optim()
optim_tol	A positive number defining convergence in numerical optimization, passed to optim()
optim_stepsize	A positive number for the step size in the numerical gradient, passed to optim()
check_stability	A logical value indicating whether to perform a stability test for the numerical optimizer.
estimation_options	Options for downstream estimates.
formula	A regression formula

Value

An object that includes estimates and information from the estimation process

---

model_to_Delta_NP *Maps model parameters, psi, to the conditional distribution Y1, Y2 | Y*, Delta.*

Description

Longer description of what it does...

Usage

```
model_to_Delta_NP(psi)
```

Arguments

psi	A numeric vector containing Pi, Delta^(1), and Delta^(2).

Details

some details.

Value

something

Examples

```
## Not run:
some example code # Should return something

## End(Not run)
```

---

model_to_Delta_NP_ind *Maps model parameters, psi, to the conditional distribution Y1, Y2 | Y*, Delta.*

Description

Longer description of what it does...

Usage

```
model_to_Delta_NP_ind(psi)
```

Arguments

psi	A numeric vector of length 2xJx(J-1) containing Delta^(1) and Delta^(2).

Details

some details.

Value

something

Examples

```
## Not run:
some example code # Should return something

## End(Not run)
```

model_to_Delta_RL_ind *Maps model parameters, psi, to the joint distribution of the data, theta.*

Description

Longer description of what it does...

Usage

```
model_to_Delta_RL_ind(psi)
```

Arguments

psi	A numeric vector of length 2(J-1)+2J corresponding to the column and row scales of the record linkage.

Details

some details.

Value

something

Examples

```
## Not run:
some example code # Should return something

## End(Not run)
```

model_to_Pi_NP *Maps model parameters, phi, to the joint distribution of X and Y*, Pi.*

Description

Longer description of what it does...

Usage

```
model_to_Pi_NP(phi)
```

Arguments

phi	A numeric vector.
J	An integer corresponding to the dimension of Y.
K	An integer corresponding to the dimension of X.

Details

some details.

Value

something

Examples

```
## Not run:
some example code # Should return something

## End(Not run)
```

`softlog` *Logarithm with a lower bound*

Description

If `x` is greater than `1e-20`, it returns `log(x)`. Otherwise, it returns `log(1e-20)`.

Usage

```
softlog(x)
```

Arguments

`x`	A numeric vector.

Details

The arguments to the `ll` function are the log value of the probabilities in Delta and Pi of the misclassification model. To prevent convergence issues, `ll` enforces a lower bound on these probabilities of `1e-20`. This function is useful for mapping probabilities that may be zero to `ll`. It will throw an error if any element of `x` is negative.

Value

A numeric vector composed of the elements of `log(x)` or `log(1e-20)` for element is less than `1e-20`.

Examples

```
## Not run:
softlog(c(0.5, 0.1, 0, -1)) # Should return the log values including log(1e-20) for 0 and -1

## End(Not run)
```

Index