Target library

Release 0.0.30

Klaus Kähler Holst

Dec 19, 2022

CONTENTS

The `target` library provides C++ classes for targeted inference and semi-parametric estimation with bindings for python and R. The library also contains implementations of various parametric models (including different discrete choice models) and model diagnostics tools.

Relevant models includes binary regression models with binary exposure and with nuisance models defined by additional covariates. Models for the relative risk and risk differences where examined by [Richardson_2018]. Various missing data estimators and causal inference models [Bang_2005], [Tsiatis_2006] also fits into this framework.

Note: This document is work in progress.

CHAPTER

ONE

CONTENTS

1.1 Installation

This program may be compiled as a shared library or as stand-alone python and R libraries.

1.1.1 C++

To build the shared library, ninja(s) are needed.

```
pip install cmake ninja
```

The following dependencies are included as submodules:

1. Armadillo http://arma.sourceforge.net/docs.html
2. doctest (unit tests only) https://github.com/onqtam/doctest/
3. pybind11++ (python bindings only) https://pybind11.readthedocs.io
4. spdlog (logging) https://github.com/gabime/spdlog

Use make to install the headers and shared libraries (here the destination is the `/opt/` directory)

```
make install INSTALL_DIR=/opt
```

1.1.2 R

The R package can be built and installed directy from CRAN

```
install.packages("targeted")
```

or installed from source

```
make r
```

1.1.3 python

The python package can be installed directly from PyPi

```
pip install targeted
```

Note: Presently binary wheels are available for Linux systems and Mac OS X running Python>=3.6. Windows installations must be built from source.

or installed from source

```
make py
```

1.2 Development

1.2.1 Contributing

Contributions to the **target** library (or any of the accompanying R or python packages) are welcome as pull requests to the develop branch. Please see the below guidelines on *coding styles*, *testing* and *documentation*.

Suggestions or bug reports can be posted here: https://github.com/kkholst/target/issues

For bug reports please include a small reproducible example which clearly demonstrates the bug.

Coding styles, unit tests and documentation

Documentation

The C++ code is documented using Doxygen.

- http://www.doxygen.nl/manual/
- http://www.doxygen.nl/manual/commands.html

The python code is documented using the numpy docstring format

- https://numpydoc.readthedocs.io/en/latest/format.html

The R code is documented using roxygen2

- https://cran.r-project.org/web/packages/roxygen2/vignettes/roxygen2.html

The main documentation of this project is based on sphinx and the in subdirectory (relative to the root of the repository) `doc/source`

- https://www.sphinx-doc.org/en/master/

The documentation should be written in ReStructuredText (rst)

- https://docutils.sourceforge.io/docs/user/rst/quickstart.html
- https://docutils.sourceforge.io/docs/user/rst/quickref.html

or alternatively in Emacs Org mode (org).

- https://orgmode.org/

Org is the *preferred format*, in which case both the exported rst and org file must be checked into the git repository (use ox-ravel to automatically export to rst). Note, that in this case only the org-file should be manually edited.

Unit tests

C++ tests are located in the subdirectory ./tests and written using doctest

- https://github.com/onqtam/doctest/blob/master/doc/markdown/tutorial.md

The unit tests can be compiled and executed from the root directory with

```
make test
```

R tests are located in the subdirectory ./R-package/targeted/tests and written using testthat

- https://r-pkgs.org/tests.html

The unit tests can be compiled and executed from the root directory with

```
make testr
```

To run the R package checks

```
make checkr
```

Python tests are located in the subdirectory ./python-package/tests

The unit tests can be compiled and executed from the root directory with

```
make testpy
```

or directly from the `python-package``directory with ``pytest-runner`:

```
pytest
```

Code coverage

Code coverage results are automatically created and posted here:

https://codecov.io/gh/kkholst/target

Reports can also be built locally by running

```
make cov
```

from either the root directory (code coverage for C++ source code) or `./python-package` (code coverage for python source codE).

Coding style

The **target** library follows the Google's C++ style guide.

The code should be checked using the `cppcheck` static code analyzer and `cclint` (which may installed from PyPi using `pip3 install cclint`).

From the root directory run

```
make check
```

Sanitizers

The Undefined Behaviour Sanitizer via `clang++` can be executed with

```
make sanitizer
```

which runs the unit-tests and examples from the directory `misc`.

1.2.2 Roadmap

1.3 Python package

1.3.1 Quickstart

Installation

Install with `pip`:

```
pip install targeted
```

Note: Presently binary wheels are available for Linux systems and Mac OS X running Python>=3.6. Windows installations must be built from source.

Risk regression

As an illustration data is loaded from the package

```
In [1]: import targeted as tg

In [2]: d = tg.getdata() # returns a Pandas DataFrame

In [3]: print(d.head())
   y  a         x         z
0  0  0 -0.626454  1.134965
1  0  0  0.183643  1.111932
2  0  0 -0.835629 -0.870778
3  1  0  1.595281  0.210732
4  1  1  0.329508  0.069396
```

Here `y` is the binary response variable, `a` binary the exposure, and `x`, `z` covariates.

Next we estimate the *risk difference* of the exposed vs the non-exposed

```
In [4]: import numpy as np

In [5]: from patsy import dmatrices

In [6]: y, X = dmatrices('y ~ x+z', d)

In [7]: a = d['a']

In [8]: ones = np.ones((y.size,1))
```

(continues on next page)

(continued from previous page)

```
In [9]: tg.riskreg(y=y, a=a, x1=ones, x2=X)
Out[9]: Riskreg. Estimate: [1.02819014]
```

Or using the formula syntax:

```
In [10]: from targeted.formula import riskreg

In [11]: riskreg(d, 'y~a')
Out[11]: Riskreg. Estimate: [1.49101228]
```

1.3.2 Suggestions

Suggestions and bug reports: https://github.com/kkholst/target/issues

1.3.3 Reference

API Reference: targeted

Subpackages

targeted.formula package

Submodules

targeted.formula.riskreg module

targeted.formula.riskreg.**riskreg**(*data: str*, *target: str*, *type='rr'*, ***kwargs*)

Risk regression with binary exposure

Parameters

- **data** (*pandas.DataFrame*) –
- **target** (*str*) – Formula describing respone and exposure/treatment of the form 'response ~ exposure'
- **type** (*str*) – Type of model (rr: relative risk, rd: risk difference)
- **nuisance** (*str, optional*) – nuisance (odds-product) regression formula (string: 'x1+x2+...')
- **propensity** (*str, optional*) – propensity model (default: same as nuisance)
- **interaction** – interactions (string: 'x1+x2+...', default: none)

Returns Riskreg object

Return type Riskreg

Module contents

Submodules

targeted.riskreg module

class targeted.riskreg.**riskreg**(*y*, *a*, ***kwargs*)

Bases: object

Documentation for Riskreg

estimate = None

mle = None

mle_coef = None

model = None

modeltype = None

propensity = None

propensity_coef = None

targeted.riskreg.**riskreg_mle**(*y*, *a*, *x2*, **args*, ***kwargs*)

Maximum Likelihood estimation of risk-regression model

The parameter of interest is either a risk difference or relative risk parameter with the effect of confounders decsribed on a log odds-product scale.

Parameters

- **y** (*list or numpy.array*) – Response vector (0,1)
- **a** (*list or numpy.array*) – Exposure vector (0,1)
- **x2** (*numpy.array*) – Design matrix for nuisance parameter regression (odds-product)
- **x1** (*numpy.array, optional*) – Design matrix for linear interactions with exposure 'a'
- **w** (*list or numpy.array, optional*) – Weights vector
- **type** (*str*) – Relative risk: rr, Risk difference: rd

Returns Riskreg object

Return type Riskreg

References

Details behind the method can be found in[1].

[1] Richardson, T. S., Robins, J. M., & Wang, L. (2017). On modeling and estimation for the relative risk and risk difference. Journal of the American Statistical Association, 112(519), 1121–1130. http://dx.doi.org/10.1080/01621459.2016.1192546

targeted module

targeted.data module

targeted.data.**getdata**(*dataset: str = 'd'*, *list: bool = False*)

Load example data from the 'targeted' package

Parameters

- **filename** (*str*) – Name of the filename of the data
- **list** (*bool*) –

Examples

```
>>> # Surgical unit data
>>> targeted.getdata('surgunit').head()
   bloodclot  prognostic  enzyme  liver function  survival
0        6.7          62      81            2.59       200
1        5.1          59      66            1.70       101
2        7.4          57      83            2.16       204
3        6.5          73      41            2.01       101
4        7.8          65     115            4.30       509
>>> # Example data for risk-regression model
>>> targeted.getdata().head()
   y  a         x         z
0  0  0 -0.626454  1.134965
1  0  0  0.183643  1.111932
2  0  0 -0.835629 -0.870778
3  1  0  1.595281  0.210732
4  1  1  0.329508  0.069396
```

targeted.data.**sim_bin**(*n=100.0*, *exposure=1*, *binary=True*, *p=1*, *rho=0.5*, *gamma=1.0*, *outcome_intercept=- 1.0*, *exposure_intercept=- 1.0*)

Simulate data from linear model or logistic regression model with binary exposure

Parameters

- **n** (*int*) – number of samples
- **exposure** (*float*) – direct exposure effect (linear predictor scale)
- **p** (*int*) – number of covariates/auxiliary variables
- **rho** (*float*) – correlation between covariates
- **gamma** (*float*) – effect of covariates on exposure (linear predictor scale)
- **outcome_intercept** (*float*) – intercept of outcome model
- **exposure_intercept** (*float*) – intercept of exposure model

Returns DataFrame with: response `y`, exposure `a`, and covariates `x1`, `x2`, ...

Return type pandas.DataFrame

Examples

```
>>> import targeted
>>> from numpy import mean
>>> d = targeted.sim_bin(n=1e4, gamma=0)
>>> ey = lambda x: mean(d[d['a']==x]['y'])
>>> ey(1)-ey(0)  # Causal effect
-0.1973030984022366  # random
```

Notes

The model for the outcome is given by

$$g(\mathbb{E}[Y \mid A, X]) = \mu_Y + \theta A + \beta^T X$$

where g is either the identify function (`binary=False`) or the logit function (`binary=True`). The exposure effect θ is controlled by the argument `exposure`.

The covariates are $X \sim \mathcal{N}_p(0, \Sigma)$ where the covariance matrix Σ is a compound symmetry matrix with correlation ρ controlled by the argument `rho`.

For the exposure the model is given by

$$\text{logit}(\mathbb{E}[A \mid X]) = \mu_A + \gamma^T X$$

The intercept terms μ_Y and μ_A are controlled by the arguments `outcome_intercept` and `exposure_intercept`.

- genindex
- modindex

1.4 R package

Contents

1.4.1 Quickstart

```
m <- lvm(y ~ a+x, a~x)
distribution(m,~ a+y) <- binomial.lvm()
d <- sim(m,1e3,seed=1)

head(d)
```

```
  y a          x
1 0 1 -0.6264538
2 1 0  0.1836433
3 1 1 -0.8356286
4 1 1  1.5952808
5 0 1  0.3295078
6 1 0 -0.8204684
```

```
library(targeted)

a <- ace(y ~ a, nuisance=~x, data=d)
summary(a)
```

```
Augmented Inverse Probability Weighting estimator
  Response y (Outcome model: logistic regression):
     y ~ x
  Exposure a (Propensity model: logistic regression):
     a ~ x

                   Estimate Std.Err    2.5%   97.5%    P-value
 a=0                0.48506 0.02626  0.4336  0.5365  3.458e-76
 a=1                0.67794 0.02225  0.6343  0.7215 6.005e-204
Outcome model:
 (Intercept)        0.44427 0.07306  0.3011  0.5875  1.196e-09
 x                  1.06929 0.08537  0.9020  1.2366  5.408e-36
Propensity model:
 (Intercept)        0.06214 0.09258 -0.1193  0.2436  5.021e-01
 x                 -0.92905 0.15311 -1.2291 -0.6289  1.297e-09

Average Causal Effect (constrast: 'a=0' vs. 'a=1'):

   Estimate Std.Err    2.5%   97.5%   P-value
RR   0.7155 0.04356  0.6301  0.8009 1.259e-60
OR   0.4475 0.06268  0.3246  0.5703 9.383e-13
RD  -0.1929 0.03295 -0.2575 -0.1283 4.791e-09
```

Note: This document is work in progress

Important: This document is work in progress.

Tip: This document is work in progress.

Warning: This document is work in progress.

Suggestions and bug reports: https://github.com/kkholst/targeted/issues

CRAN: https://cran.r-project.org/package=targeted

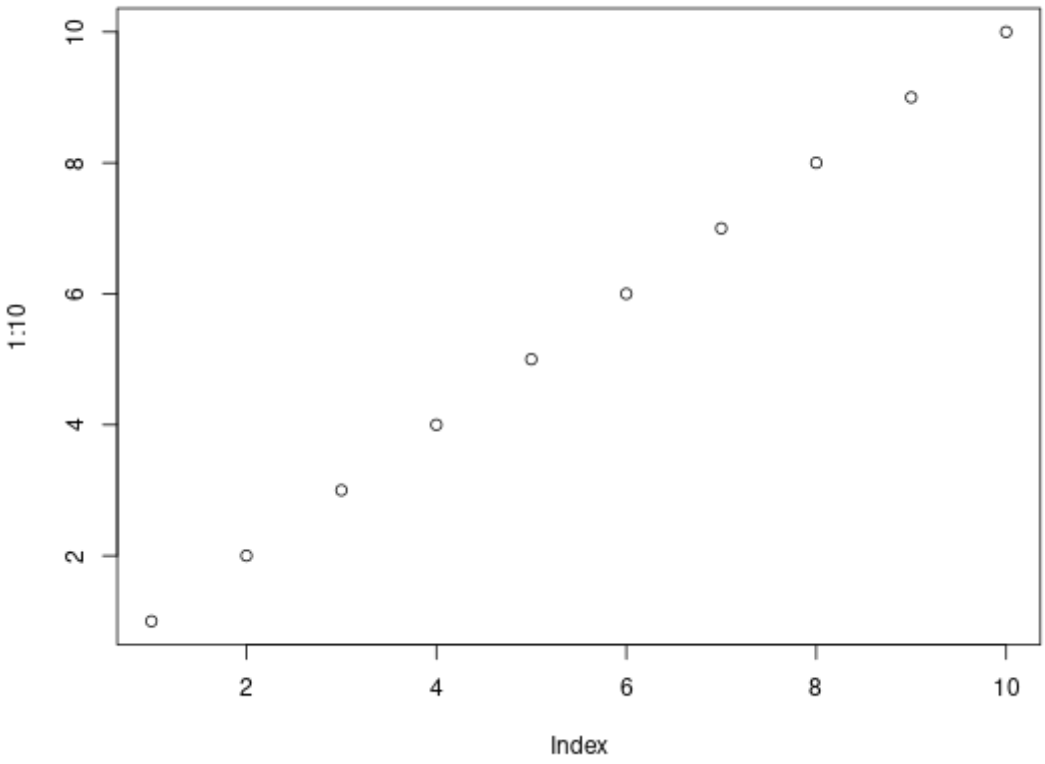


Fig. 1: Caption goes here

1.5 Target C++ Library

C++ interface documentation

Suggestions and bug reports: https://github.com/kkholst/target/issues

i

Chapter 1

Main Page

Introduction

This library provides C++ classes for targeted inference and semi-parametric efficient estimators as well as bindings for python and R. The library also contains implementation of parametric models (including different discrete choice models) and model diagnostics tools.

Relevant models includes binary regression models with binary exposure and with nuisance models defined by additional covariates. Models for the relative risk and risk differences where examined by (Richardson et al 2017). Various missing data estimators and causal inference models (Bang & Robins 2005, Tsiatis 2006) also fits into this framework.

Documentation

The main documentation can be found here: https://targetlib.org/ (PDF version)

Examples

R

https://kkholst.github.io/targeted

```
library('targeted')

# Simulation
m <- lvm(y[-2] ~ 1*x,
         lp.target[1] ~ 1,
         lp.nuisance[-1] ~ 2*x)
distribution(m, ~a) <- binomial.lvm('logit')
m <- binomial.rr(m, 'y', 'a', 'lp.target', 'lp.nuisance')
dd <- sim(m, 5e2, seed=1)

# Double-robust estimator
summary(fit <- targeted::riskreg(y ~ a | 1 | x | x, data=dd, type="rr"))
#
#  Relative risk model
#   Response:  y
#   Exposure:  a
#
#             Estimate Std.Err    2.5%    97.5%   P-value
# log(RR):
#  (Intercept)  0.86136 0.11574  0.6345  1.08820 9.895e-14
# log(OP):
#  (Intercept) -0.88518 0.22802 -1.3321 -0.43827 1.036e-04
#  x            2.35193 0.28399  1.7953  2.90854 1.213e-16
# logit(Pr):
#  (Intercept) -0.07873 0.08857 -0.2523  0.09485 3.740e-01
#  x            0.02894 0.08291 -0.1336  0.19145 7.270e-01
```

Python

https://pypi.org/project/targeted/

```
import targeted as tg
from targeted.formula import riskreg

d = tg.getdata()
val = riskreg(d, 'y~a', interaction='x', nuisance='x+z')

print(val)
## Riskreg. Estimate: [ 1.17486406 -0.23623467]
```

C++

https://www.targetlib.org/cppapi/

```
#include <target/target.h>
using namespace arma;

void main() {

  ...

  targeted::RR<double> model(y, a, x1, x2, x3, p, w);
  mat pp0 = model.TargetedBinary::pa();
  mat U = model.score(false);
  mat res = model.est(alpha);
}
```

Installation

This program may be compiled as a shared library or as stand-alone python and R libraries.

To compile and run the unit tests:

```
make test
```

Syntax checks (requires `cppcheck` and `cclint`), code coverage, and check for memory leaks

```
make check coverage
make valgrind
```

R

The R package can be built and installed with

```
make r
```

Python

The python package can be installed directly from PyPi (wheels available for Mac OS X and Linux):

```
pip install targeted
```

or installed from source

```
make py
```

Dependencies

The following dependencies are included as submodules:

1. Armadillo http://arma.sourceforge.net/docs.html
2. doctest (unit tests only) https://github.com/onqtam/doctest/
3. pybind11++ (python bindings only) https://pybind11.readthedocs.io
4. spdlog (logging) https://github.com/gabime/spdlog

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

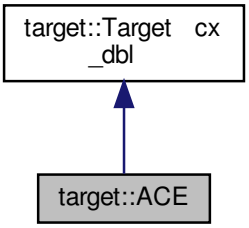
Here is a list of all documented files with brief descriptions:

Chapter 5

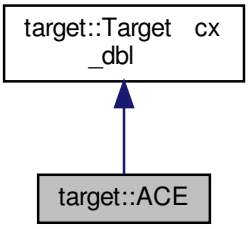
Class Documentation

5.1 target::ACE Class Reference

Inheritance diagram for target::ACE:



Collaboration diagram for target::ACE:



Public Member Functions

- ACE (const arma::cx_vec &y, const arma::cx_mat &a, const arma::cx_mat &x2, const arma::cx_mat &x3, const arma::cx_vec &parameter, const arma::cx_vec &weights, bool binary=true)

  ACE AIPW for Average Causal Effects.
- **ACE** (const arma::vec &y, const arma::vec &a, const arma::mat &x2, const arma::mat &x3, const arma::vec &parameter, const arma::vec &weights, bool binary=true)
- void **calculate** (bool target=true, bool nuisance=true, bool propensity=true) override
- void **update_par** (arma::cx_vec par)
- void **update_par** (arma::vec par)
- arma::cx_mat **est** (arma::cx_vec par, bool indiv=false, const cx_dbl &value=1)
- arma::cx_mat **est** (bool indiv=false, const cx_dbl &value=1)
- arma::mat deriv (const cx_dbl &value=1)

  deriv Calculate derivative of estimating function for AIPW estimator

Protected Attributes

- bool **binary**

Additional Inherited Members

5.1.1 Detailed Description

Definition at line 152 of file target.hpp.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 ACE()

```
target::ACE::ACE (
            const arma::cx_vec & y,
            const arma::cx_mat & a,
            const arma::cx_mat & x2,
            const arma::cx_mat & x3,
            const arma::cx_vec & parameter,
            const arma::cx_vec & weights,
            bool bin = true )
```

ACE AIPW for Average Causal Effects.

Assume we have iid observations (Y_i, A_i, X_i)

Parameters

y	Response variable
a	Exposure variable
x2	Design matrix for outcome regression. Note must be calculated for fixed A=a (to derive distribution of potential outcome $Y^{\triangle}*$(a))
x3	Design matrix for propensity model
parameter	Parameters from outcome regression and propensity model
weights	Weight vector
binary	If TRUE outcome regression is assumed to be a logistic regression model

Returns

ACE

Definition at line 357 of file target.cpp.

5.1.3 Member Function Documentation

5.1.3.1 deriv()

```
arma::mat target::ACE::deriv (
            const cx_dbl & value = 1 )
```

deriv Calculate derivative of estimating function for AIPW estimator

Complex step derivative.

Parameters

value	Matrix with partial derivatives of estimating function

Returns

arma::mat

Definition at line 423 of file target.cpp.

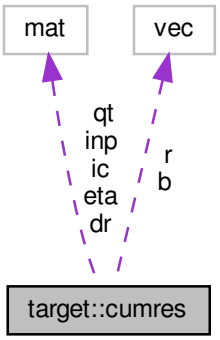
The documentation for this class was generated from the following files:

- target.hpp
- target.cpp

5.2 target::cumres Class Reference

```
#include <cumres.hpp>
```

Collaboration diagram for target::cumres:



Public Member Functions

- cumres (const arma::vec &r, const arma::mat &dr, const arma::mat &ic)

  Constructor.
- void order (const arma::mat &inp, arma::vec b=arma::vec())

  Set variables to order after and bandwidth.
- arma::vec rnorm ()

  Draw n samples from standard normal distribution.
- arma::mat obs ()

  Calculate observed cumulative residual process.
- arma::mat sample (const arma::umat &idx=arma::umat())

  Simulate one process under the null hypothesis of a correctly specified model.
- arma::mat sample (unsigned R, const arma::umat &idx=arma::umat(), bool quantiles=true)

  Sample R processes and calculate test statistics under the null (Supremum and L2 statistics)

Public Attributes

- unsigned n

  Sample size.
- arma::vec r

  Residuals.
- arma::umat ord

  Stores order of observations to cumulate after.
- arma::mat dr

  Derivative of residuals wrt model parameters.
- arma::mat ic

  Influence curve.
- arma::mat inp

  Variable to order residuals after.
- arma::vec b

Bandwidth of moving average.

- arma::mat qt

  Stores data for calculations of quantiles.

- arma::mat eta

  Cumulative derivative of residuals.

5.2.1 Detailed Description

The `cumres` class provides a data structure for calculating goodness-of-fit statistics based on aggregation of residuals (cumulative residuals) of a statistical model.

Definition at line 21 of file cumres.hpp.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 cumres()

```
target::cumres::cumres (
            const arma::vec & r,
            const arma::mat & dr,
            const arma::mat & ic )
```

Constructor.

Constructor for the cumres class.

Parameters

r	column vector of residuals
dr	matrix of partial deriatives of the residuals wrt to the parameter vector
ic	matrix with the estimated influence functions for the parametric model

Definition at line 22 of file cumres.cpp.

5.2.3 Member Function Documentation

5.2.3.1 obs()

```
arma::mat target::cumres::obs ( )
```

Calculate observed cumulative residual process.

Calculate the observed cumulative residual process

$$W(t) = n^{-1/2} \sum_{i=1}^{n} 1\{t - b < X_i \leq t\} r_i,$$

where r_i is the the residual corresponding to the ith observation and X_i is the variable which the process is ordered against (as defined by the `inp` argument to cumres::order).

When `b` is not set (i.e., an empty vector) the standard cumulative residual process is calculated (corresponding to $b = \infty$):

$$W(t) = n^{-1/2} \sum_{i=1}^{n} 1\{X_i \leq t\} r_i.$$

Definition at line 89 of file cumres.cpp.

5.2.3.2 rnorm()

```
arma::vec target::cumres::rnorm ( )
```

Draw n samples from standard normal distribution.

Sample n independent standard normal distributed variables.

Definition at line 64 of file cumres.cpp.

5.2.3.3 sample() [1/2]

```
arma::mat target::cumres::sample (
            const arma::umat & idx = arma::umat() )
```

Simulate one process under the null hypothesis of a correctly specified model.

Obtain a single sample of the residual process under the null hypothesis (true model).

Parameters

idx	indices in which to evaluate the process. If this is an empty vector the process is evaluated in all observed points.

Definition at line 117 of file cumres.cpp.

5.2.3.4 sample() [2/2]

```
arma::mat target::cumres::sample (
            unsigned R,
            const arma::umat & idx = arma::umat(),
            bool quantiles = true )
```

Sample R processes and calculate test statistics under the null (Supremum and L2 statistics)

Draw R samples from the cumulative residual process under the null hypothesis (true model)

Parameters

R	Number of process to sample
idx	subset of indices to evalute the process in
quantiles	Boolean that defines whether quantiles of the sampled process is to be estimated

Returns

arma::mat $R \times 2p$ matrix with Supremum and L2 test statistics for each of the p variables (columns in the `inp` variable defined in cumres::order)

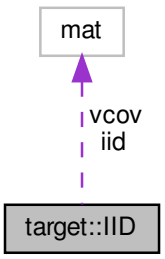
Definition at line 162 of file cumres.cpp.

The documentation for this class was generated from the following files:

- cumres.hpp
- cumres.cpp

5.3 target::IID Class Reference

Collaboration diagram for target::IID:



Public Member Functions

- **IID** (arma::mat score, arma::mat v)

Public Attributes

- arma::mat **iid**
- arma::mat **vcov**

5.3.1 Detailed Description

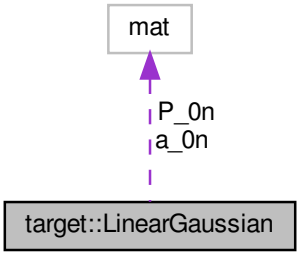
Definition at line 36 of file glm.hpp.

The documentation for this class was generated from the following file:

- glm.hpp

5.4 target::LinearGaussian Class Reference

Collaboration diagram for target::LinearGaussian:



Public Member Functions

- void **UpdateData** (const arma::cube Y, const arma::mat T, const arma::cube Z, const arma::mat H, const arma::mat Q, const arma::vec a0, const arma::mat P0, const arma::mat K, const arma::mat L, const arma::cube x, const arma::vec c, const arma::vec d)
- **LinearGaussian** (const arma::cube Y, const arma::mat T, const arma::cube Z, const arma::mat H, const arma::mat Q, const arma::vec a0, const arma::mat P0, const arma::mat K, const arma::mat L, const arma::cube x, const arma::vec c, const arma::vec d)
- void **filter** ()
- void **smoother** ()

Public Attributes

- unsigned **n**
- unsigned **p**
- unsigned **m**
- double **llh**
- arma::cube **a_t**
- arma::cube **P_t**
- arma::cube **a_tt**
- arma::cube **P_tt**
- arma::cube **v_t**
- arma::cube **K_t**
- arma::cube **F_t**
- arma::cube **F_t_inv**
- arma::cube **a_tn**
- arma::cube **P_tn**
- arma::cube **J_t**
- arma::cube **P_tt1n**
- arma::mat **a_0n**
- arma::mat **P_0n**

5.4.1 Detailed Description

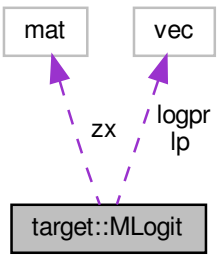
Definition at line 17 of file kalman.hpp.

The documentation for this class was generated from the following files:

- kalman.hpp
- kalman.cpp

5.5 target::MLogit Class Reference

Collaboration diagram for target::MLogit:



Public Member Functions

- **MLogit** (const arma::uvec &choice, const arma::uvec &alt, const arma::uvec &id_idx, const arma::mat &z1, const arma::mat &z2, const arma::mat &x, unsigned nalt=0, arma::vec weights=arma::vec())
- arma::mat **hessian** (bool update=false)
- arma::mat **score** (bool update=false, bool indiv=true)
- double **loglik** ()
- void **updateZX** ()
- void **updateRef** (unsigned basealt)
- void **updatePar** (arma::vec theta)
- arma::vec **getPar** ()
- void **updateProb** ()
- void **updateProb** (arma::vec theta)
- const arma::vec ∗ **Weights** ()
- double **Weights** (unsigned i)
- const arma::uvec ∗ **Choice** ()
- unsigned **Choice** (unsigned i)
- const arma::uvec ∗ **Alt** ()
- unsigned **Alt** (unsigned i)
- const arma::mat ∗ **X** ()
- arma::rowvec **X** (unsigned row)
- const arma::mat ∗ **Z1** ()
- arma::rowvec **Z1** (unsigned row)
- const arma::mat ∗ **Z2** ()
- arma::rowvec **Z2** (unsigned row)
- const arma::uvec ∗ **cluster** ()
- unsigned **cluster** (unsigned index)
- void **updateData** (const arma::uvec &choice, const arma::uvec &alt, const arma::uvec &id_idx, const arma↩ ::mat &z1, const arma::mat &z2, const arma::mat &x, const arma::vec &weights)

Public Attributes

- unsigned **n**
- unsigned **ncl**
- unsigned **J**
- unsigned **basealt** = 0
- arma::vec **lp**
- arma::mat **zx**
- arma::vec **logpr**
- unsigned **p_z1**
- unsigned **p_z2**
- unsigned **p_x**

5.5.1 Detailed Description

Definition at line 18 of file mlogit.hpp.

The documentation for this class was generated from the following files:

- mlogit.hpp
- mlogit.cpp

5.6 QuadRule Class Reference

Public Member Functions

- arma::vec **Weight** ()
- arma::vec **Abscissa** ()
- **QuadRule** (int n=1)

5.6.1 Detailed Description

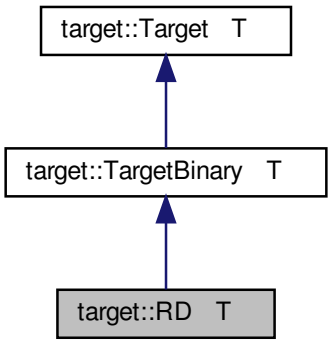
Definition at line 7 of file quadrule.h.

The documentation for this class was generated from the following file:

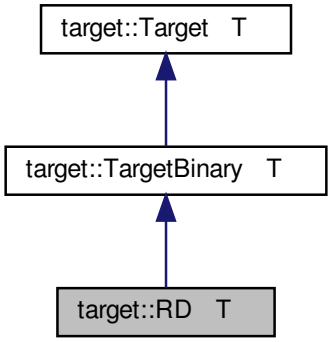
- quadrule.h

5.7 target::RD< T > Class Template Reference

Inheritance diagram for target::RD< T >:



Collaboration diagram for target::RD< T >:



Public Member Functions

- **RD** (const arma::Col< T > &y, const arma::Mat< T > &a, const arma::Mat< T > &x1, const arma::Mat< T > &x2, const arma::Mat< T > &x3, const arma::Col< T > &parameter, const arma::Col< T > &weights)
- arma::Col< T > **rd** ()
- arma::Col< T > **op** ()
- void **calculate** (bool target=true, bool nuisance=true, bool propensity=false) override

Additional Inherited Members

5.7.1 Detailed Description

template<class T>
class target::RD< T >

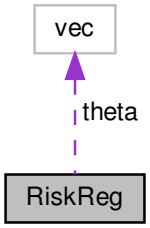
Definition at line 109 of file target.hpp.

The documentation for this class was generated from the following files:

- target.hpp
- target.cpp

5.8 RiskReg Class Reference

Collaboration diagram for RiskReg:



Public Types

- using **cx_dbl** = target::cx_dbl
- using **cx_func** = target::cx_func

Public Member Functions

- **RiskReg** (const arma::vec &y, const arma::vec &a, const arma::mat &x1, const arma::mat &x2, const arma::mat &x3, const arma::vec &weights, std::string model)
- void **setData** (const arma::vec &y, const arma::vec &a, const arma::mat &x1, const arma::mat &x2, const arma::mat &x3, const arma::vec &weights)
- void **weights** (const arma::vec &weights)
- void **update** (arma::vec &par)
- double **logl** ()
- arma::mat **dlogl** (bool indiv=false)
- arma::mat **pr** ()
- arma::cx_mat **score** (arma::cx_vec theta)
- arma::mat **esteq** (arma::vec &alpha, arma::vec &pr)
- arma::mat **hessian** ()
- arma::mat **operator()** (Data index) const

Protected Attributes

- std::unique_ptr< target::TargetBinary< double > > **model**
- std::unique_ptr< target::TargetBinary< cx_dbl > > **model_c**
- arma::vec **theta**
- std::string **type**

5.8.1 Detailed Description

Definition at line 22 of file riskreg.hpp.

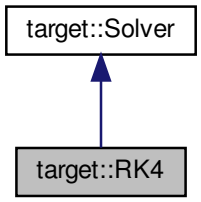
The documentation for this class was generated from the following file:

- riskreg.hpp

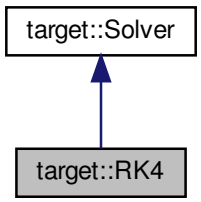
5.9 target::RK4 Class Reference

`#include <odesolver.hpp>`

Inheritance diagram for target::RK4:



Collaboration diagram for target::RK4:



Public Member Functions

- arma::mat **solve** (const arma::mat &input, arma::mat init, arma::mat theta)

Additional Inherited Members

5.9.1 Detailed Description

Runge-Kutta solver

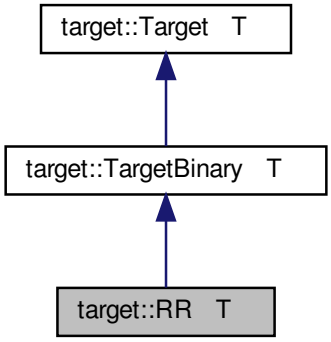
Definition at line 52 of file odesolver.hpp.

The documentation for this class was generated from the following files:

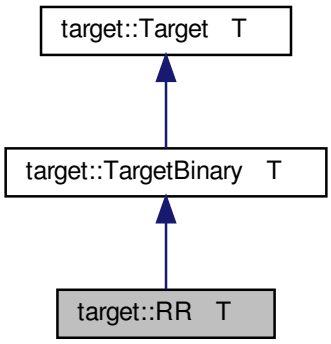
- odesolver.hpp
- odesolver.cpp

5.10 target::RR< T > Class Template Reference

Inheritance diagram for target::RR< T >:



Collaboration diagram for target::RR< T >:



Public Member Functions

- **RR** (const arma::Col< T > &y, const arma::Mat< T > &x, const arma::Mat< T > &z1, const arma::Mat< T > &z2, const arma::Mat< T > &z3, const arma::Col< T > &parameter, const arma::Col< T > &weights)
- arma::Col< T > **rr** ()
- arma::Col< T > **op** ()
- void **calculate** (bool target=true, bool nuisance=true, bool propensity=false) override

Additional Inherited Members

5.10.1 Detailed Description

template<**class T**>
class target::RR< T >

Definition at line 128 of file target.hpp.

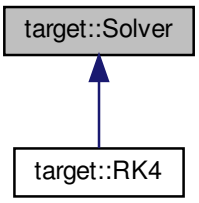
The documentation for this class was generated from the following files:

- target.hpp
- target.cpp

5.11 target::Solver Class Reference

`#include <odesolver.hpp>`

Inheritance diagram for target::Solver:



Public Member Functions

- **Solver** (odefunc F)
- virtual arma::mat **solve** (const arma::mat &input, arma::mat init, arma::mat theta)=0
- arma::mat **solveint** (const arma::mat &input, arma::mat init, arma::mat theta, double tau=1.0e-1, bool reduce=true)

Protected Attributes

- odefunc **F**

5.11.1 Detailed Description

Abstract class for ODE Solver

Definition at line 29 of file odesolver.hpp.

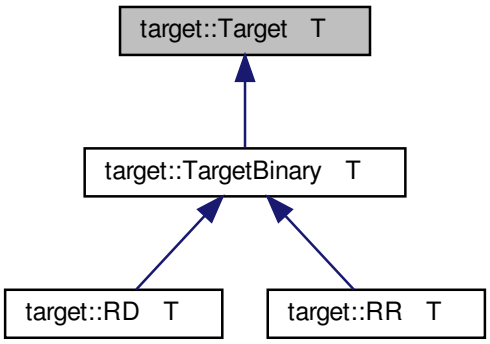
The documentation for this class was generated from the following files:

- odesolver.hpp
- odesolver.cpp

5.12 target::Target< T > Class Template Reference

```
#include <target.hpp>
```

Inheritance diagram for target::Target< T >:



Public Member Functions

- Target ()

  Constructor.
- Target (const arma::Col< T > &y, const arma::Mat< T > &a, const arma::Mat< T > &x1, const arma::Mat< T > &x2, const arma::Mat< T > &x3, const arma::Col< T > &parameter, const arma::Col< T > &weights)

  Target class constructur.
- **Target** (const arma::Col< T > &y, const arma::Mat< T > &a, const arma::Mat< T > &x1, const arma::Mat< T > &x2, const arma::Col< T > &parameter, const arma::Col< T > &weights)

- **Target** (const arma::Col< T > &y, const arma::Mat< T > &a, const arma::Mat< T > &x1, const arma::Mat< T > &x2, const arma::Mat< T > &x3, const arma::Col< T > &parameter)
- **Target** (const arma::Col< T > &y, const arma::Mat< T > &a, const arma::Mat< T > &x1, const arma::Mat< T > &x2, const arma::Col< T > &parameter)
- void **weights** (const arma::Col< T > &weights)
- arma::Col< T > **weights** ()
- arma::Col< T > **A** ()
- arma::Col< T > **Y** ()
- arma::Mat< T > **X1** ()
- arma::Mat< T > **X2** ()
- arma::Mat< T > **X3** ()
- void **update_data** (const arma::Col< T > &y, const arma::Mat< T > &a, const arma::Mat< T > &x1, const arma::Mat< T > &x2, const arma::Mat< T > &x3)
- virtual void **calculate** (bool target=true, bool nuisance=true, bool propensity=false)
- void update_par (const arma::Col< T > &parameter)

  update_par -

Public Attributes

- arma::Col< T > **alpha**
- arma::Col< T > **beta**
- arma::Col< T > **gamma**

Protected Attributes

- arma::Col< T > **nuisance**
- arma::Col< T > **target**
- arma::Col< T > **propensity**
- arma::Col< T > **_response**
- arma::Mat< T > **_exposure**
- arma::Mat< T > **_x1**
- arma::Mat< T > **_x2**
- arma::Mat< T > **_x3**
- arma::Col< T > **_weights**

5.12.1 Detailed Description

template<**class T**>
class target::Target< T >

Abstract class for Targeted Learning problems

Definition at line 19 of file target.hpp.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 Target()

```
template<typename T>
target::Target< T >::Target (
            const arma::Col< T > & y,
            const arma::Mat< T > & a,
            const arma::Mat< T > & x1,
            const arma::Mat< T > & x2,
            const arma::Mat< T > & x3,
            const arma::Col< T > & parameter,
            const arma::Col< T > & weights )
```

Target class constructur.

Parameters

y	- Response vector
a	- Exposure/treatment vector
x1	- Design matrix for target model
x2	- Design matrix for nuisance model
x3	- Design matrix for exposure propensity
parameter	- Parameter vector (order: `target`, `nuisance`, and optionally `propensity`)
weights	- Optional weights

Returns

Target class

Definition at line 34 of file target.cpp.

5.12.3 Member Function Documentation

5.12.3.1 update_par()

```
template<typename T>
void target::Target< T >::update_par (
            const arma::Col< T > & parameter )
```

update_par -

Parameters

parameter	- Description of parameter

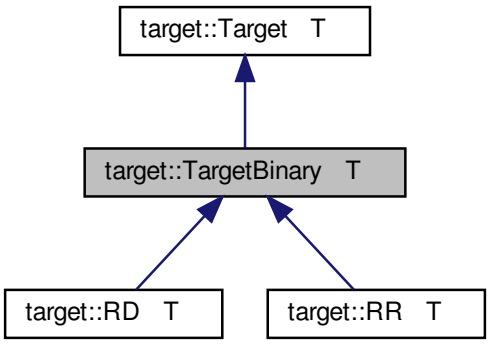
Definition at line 51 of file target.cpp.

The documentation for this class was generated from the following files:

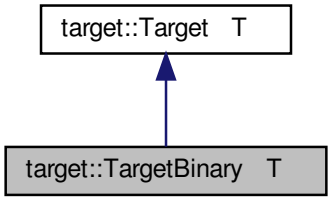
- target.hpp
- target.cpp

5.13 target::TargetBinary< T > Class Template Reference

Inheritance diagram for target::TargetBinary< T >:



Collaboration diagram for target::TargetBinary< T >:



Public Member Functions

- virtual arma::Mat< T > pa ()

  Calculate risk probabilities conditional on exposure.
- virtual arma::Mat< T > **p** (bool exposure=0)
- virtual arma::Col< T > loglik (bool indiv=false)

  log likelihood
- virtual arma::Mat< T > score (bool indiv=false)

  score function

- virtual arma::Mat< T > est (arma::Col< T > alpha, const arma::Col< T > &propensity)

  Estimating function for double robust estimator.
- virtual arma::Mat< T > **est** (arma::Col< T > alpha)
- void **calculate** (bool target=true, bool nuisance=true, bool propensity=false) override

Protected Member Functions

- virtual arma::Col< T > **H** ()=0
- virtual arma::Mat< T > **dp** ()=0

Protected Attributes

- arma::Mat< T > **pr**

Additional Inherited Members

5.13.1 Detailed Description

template<class T>
class target::TargetBinary< T >

Definition at line 87 of file target.hpp.

5.13.2 Member Function Documentation

5.13.2.1 est()

```
template<typename T >
arma::Mat< T > target::TargetBinary< T >::est (
            arma::Col< T > alpha,
            const arma::Col< T > & propensity )  [virtual]
```

Estimating function for double robust estimator.

$$U(\alpha;\widehat{\alpha},\widehat{\beta},\widehat{\gamma}) = \omega(V)\Big(A - e(V;\widehat{\gamma})\Big)\Big(H(\alpha) - p_0(V;\widehat{\alpha},\widehat{\beta})\Big)$$

1. Inital estimates $\widehat{\alpha},\widehat{\beta}$ obtained from MLE.
2. Similarly, $\widehat{\gamma}$ obtained from regular asymptotic linear model (e.g., logistic regression MLE).
3. Plugin estimates to obtain $\widehat{\omega}_{\text{eff}}$. Also, note that p_0 is calculated wrt initial MLE.

Parameters

alpha	target parameter
propensity	exposure propensity weights

Returns

arma::Mat<T>

Definition at line 170 of file target.cpp.

5.13.2.2 loglik()

```
template<typename T >
arma::Col< T > target::TargetBinary< T >::loglik (
            bool indiv = false )  [virtual]
```

log likelihood

Parameters

indiv	- if `true` return individual log likelihood contributions

Returns

vector of type `arma::Vec<T>`

Definition at line 142 of file target.cpp.

5.13.2.3 pa()

```
template<typename T >
arma::Mat< T > target::TargetBinary< T >::pa ( )  [virtual]
```

Calculate risk probabilities conditional on exposure.

$p_a(V) = E(Y \mid A = a, V), a \in \{0, 1\}$

Returns

arma::mat

Definition at line 125 of file target.cpp.

5.13.2.4 score()

```
template<typename T >
arma::Mat< T > target::TargetBinary< T >::score (
            bool indiv = false )  [virtual]
```

score function

Parameters

indiv	- If `true` the individual score contributions are returned, otherwise the sum is returned

Returns

arma::Mat

Definition at line 200 of file target.cpp.

The documentation for this class was generated from the following files:

- target.hpp
- target.cpp

Chapter 6

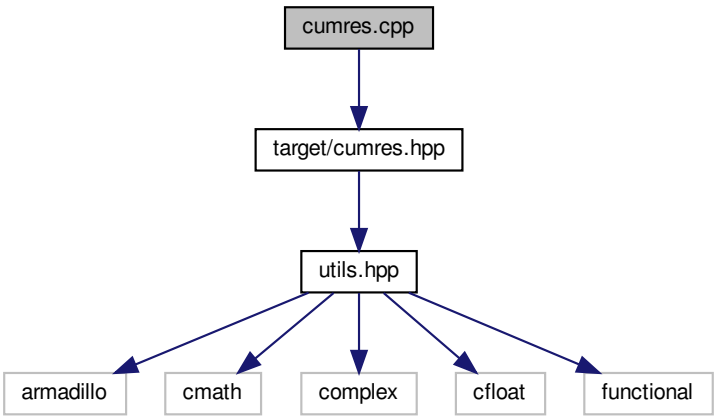
File Documentation

6.1 cumres.cpp File Reference

Generic function for calculating cumulative residuals.

```
#include <target/cumres.hpp>
```

Include dependency graph for cumres.cpp:



6.1.1 Detailed Description

Generic function for calculating cumulative residuals.

Author

Klaus K. Holst

Copyright

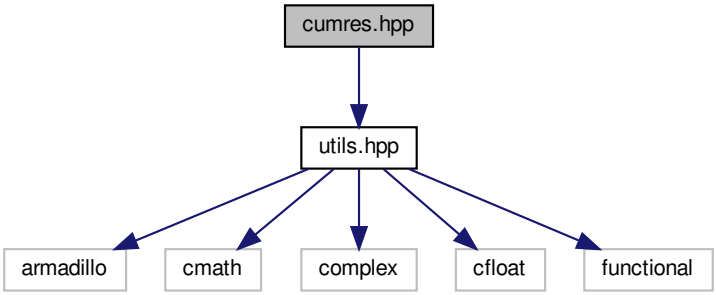
2020-2022, Klaus Kähler Holst

Test statistics

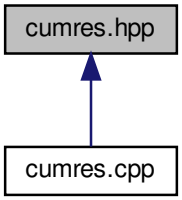
6.2 cumres.hpp File Reference

Generic class for cumulative residuals.

`#include "utils.hpp"`
Include dependency graph for cumres.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class target::cumres

6.2.1 Detailed Description

Generic class for cumulative residuals.

Author

Klaus K. Holst

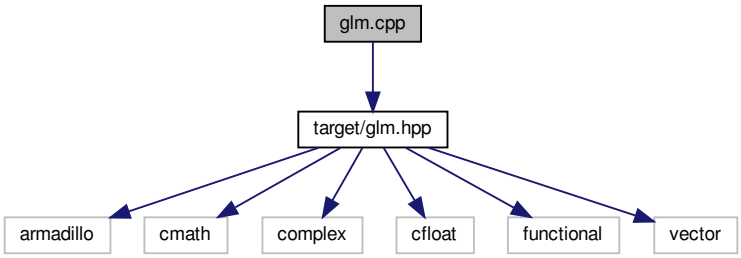
Copyright

2020-2021, Klaus Kähler Holst

6.3 glm.cpp File Reference

Utility functions for Generalized Linear Models.

`#include <target/glm.hpp>`
Include dependency graph for glm.cpp:



Functions

- arma::mat **target::softmax** (arma::mat lp, bool ref, bool log)
- arma::vec **target::softmax** (arma::vec u)
- arma::mat **target::expit** (arma::mat x)
- arma::cx_mat **target::expit** (arma::cx_mat x)
- IID **target::logistic_iid** (const arma::vec &y, const arma::vec &p, const arma::mat &x, const arma::vec &w)
- IID **target::linear_iid** (const arma::vec &y, const arma::vec &p, const arma::mat &x, const arma::vec &w)

6.3.1 Detailed Description

Utility functions for Generalized Linear Models.

Author

Klaus K. Holst

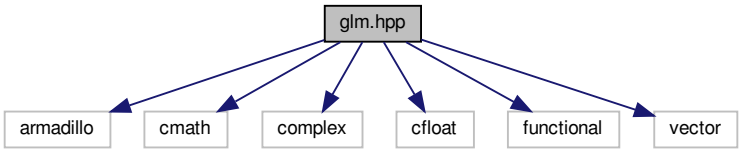
Copyright

6.4 glm.hpp File Reference

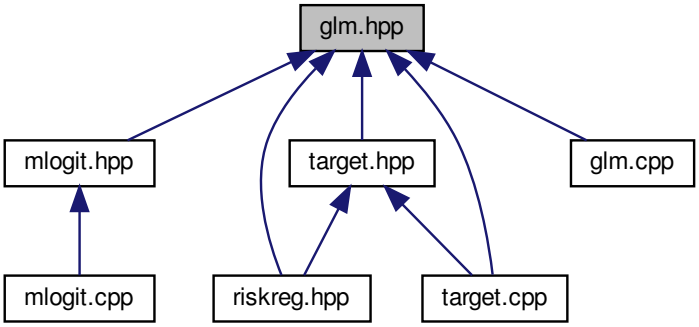
Utility functions for Generalized Linear Models.

```
#include <armadillo>
#include <cmath>
#include <complex>
#include <cfloat>
#include <functional>
#include <vector>
```

Include dependency graph for glm.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class target::IID

Typedefs

- using **cx_dbl** = std::complex< double >
- using **cx_func** = std::function< arma::cx_mat(arma::cx_vec theta)>
- using **matlist** = std::vector< arma::mat >

Functions

- arma::mat **target::expit** (arma::mat x)
- arma::cx_mat **target::expit** (arma::cx_mat x)
- arma::vec **target::softmax** (arma::vec u)
- arma::mat **target::softmax** (arma::mat lp, bool ref, bool log)
- IID **target::logistic_iid** (const arma::vec &y, const arma::vec &p, const arma::mat &x, const arma::vec &w)
- IID **target::linear_iid** (const arma::vec &y, const arma::vec &p, const arma::mat &x, const arma::vec &w)

6.4.1 Detailed Description

Utility functions for Generalized Linear Models.

Author

Klaus K. Holst

Copyright

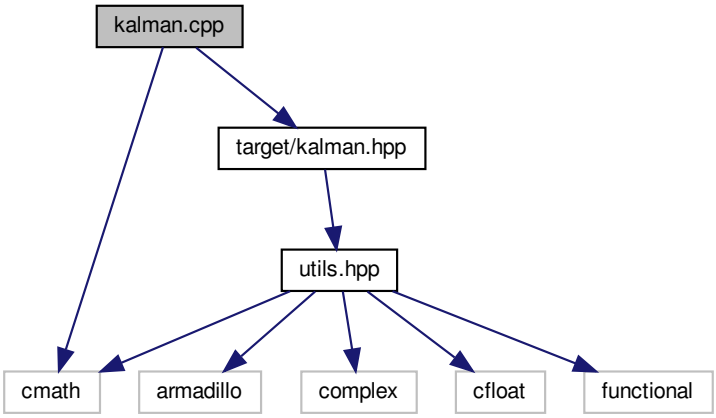
2018-2021, Klaus Kähler Holst

6.5 kalman.cpp File Reference

Kalman filter.

```
#include <target/kalman.hpp>
#include <cmath>
```

Include dependency graph for kalman.cpp:



6.5.1 Detailed Description

Kalman filter.

Author

Klaus K. Holst

Copyright

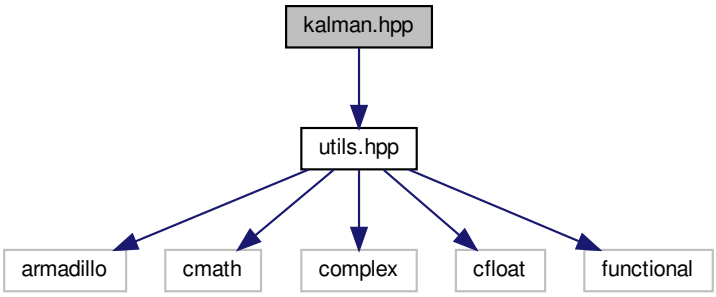
2020-2021, Klaus Kähler Holst

6.6 kalman.hpp File Reference

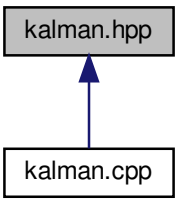
Kalman filter.

```
#include "utils.hpp"
```

Include dependency graph for kalman.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class target::LinearGaussian

6.6.1 Detailed Description

Kalman filter.

Author

Klaus K. Holst

Copyright

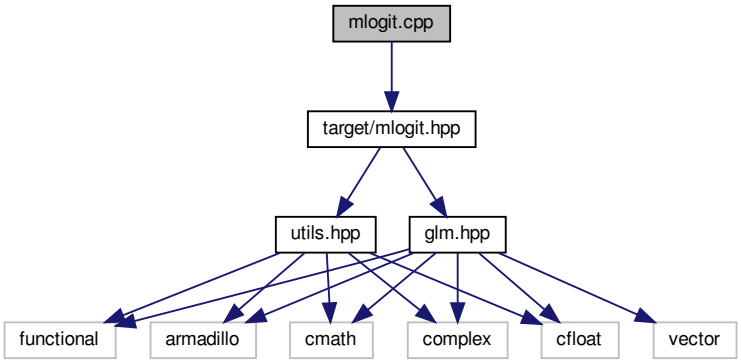
2020-2021, Klaus

6.7 mlogit.cpp File Reference

Conditional logit models.

```
#include <target/mlogit.hpp>
```

Include dependency graph for mlogit.cpp:



6.7.1 Detailed Description

Conditional logit models.

Author

Klaus K. Holst

Copyright

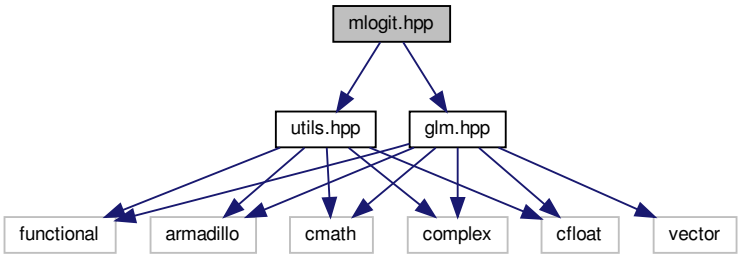
2020-2022, Klaus Kähler Holst

6.8 mlogit.hpp File Reference

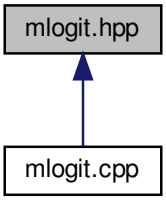
Multinomial logit models.

```
#include "glm.hpp"
#include "utils.hpp"
```

Include dependency graph for mlogit.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class target::MLogit

6.8.1 Detailed Description

Multinomial logit models.

Author

Klaus K. Holst

Copyright

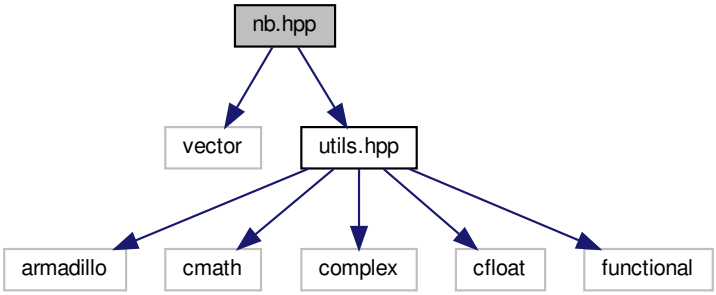
2020-2021, Klaus Kähler Holst

6.9 nb.hpp File Reference

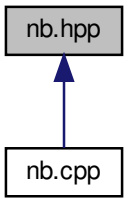
Weighted Naive Bayes.

```
#include <vector>
#include "utils.hpp"
```

Include dependency graph for nb.hpp:



This graph shows which files directly or indirectly include this file:



Typedefs

- using **target::raggedArray** = std::vector< arma::vec >

Functions

- raggedArray **target::pcond** (const arma::uvec &idx, const arma::mat &x, const arma::uvec &xlev, const arma::vec &weights, double laplacesmooth)
- std::vector< raggedArray > **target::nb** (arma::vec y, arma::mat x, arma::uvec xlev=arma::uvec(), arma::vec ylev=arma::vec(), arma::vec weights=arma::vec(), double laplacesmooth=1.0)
- arma::mat **target::prednb** (arma::mat const &X, raggedArray const &condprob, raggedArray const &xord, arma::uvec multinomial, arma::vec prior=arma::vec(), double threshold=1E-3)

6.9.1 Detailed Description

Weighted Naive Bayes.

Author

Klaus K. Holst

Copyright

2020-2021, Klaus Kähler Holst

Author

Klaus K. Holst

Copyright

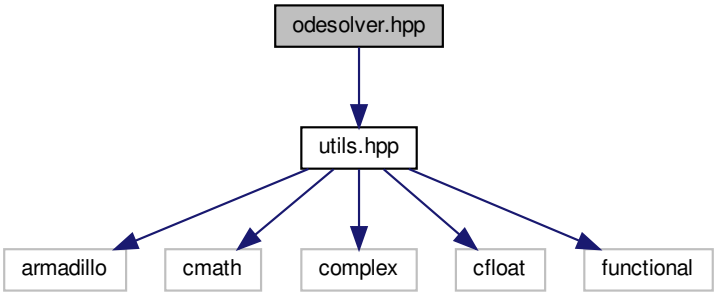
2020-2022, Klaus Kähler Holst

6.10 odesolver.hpp File Reference

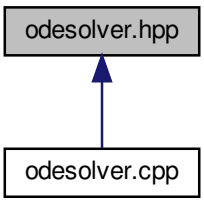
Classes for Ordinary Differential Equation Solvers.

```
#include "utils.hpp"
```

Include dependency graph for odesolver.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class target::Solver
- class target::RK4

Typedefs

- using **target::odefunc** = std::function< arma::mat(arma::mat input, arma::mat x, arma::mat theta)>

Functions

- arma::mat **target::interpolate** (const arma::mat &input, double tau, bool locf=false)

6.10.1 Detailed Description

Classes for Ordinary Differential Equation Solvers.

Author

Klaus K. Holst

Copyright

2019, Klaus Kähler Holst

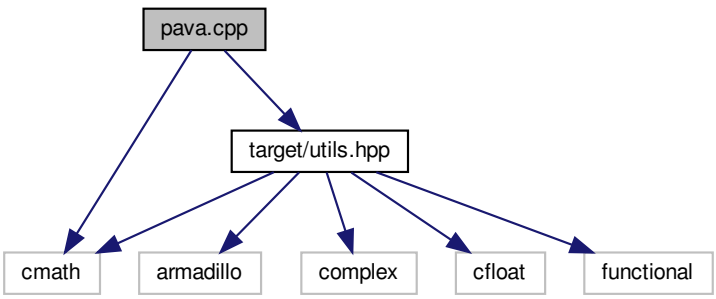
4th order Runge-Kutta

6.11 pava.cpp File Reference

Pooled Adjacent Violator Algorithms.

```
#include <cmath>
#include <target/utils.hpp>
```

Include dependency graph for pava.cpp:



Functions

- arma::mat target::pava (arma::vec y, const arma::vec &x=arma::vec(), arma::vec w=arma::vec())
  Weighted Pooled Adjacent Violator Algorithm.

6.11.1 Detailed Description

Pooled Adjacent Violator Algorithms.

Author

Klaus K. Holst

Copyright

6.11.2 Function Documentation

6.11.2.1 pava()

```
arma::mat target::pava (
            arma::vec y,
            const arma::vec & x,
            arma::vec w )
```

Weighted Pooled Adjacent Violator Algorithm.

Pooled Adjacent Violator Algorithm for

Parameters

y	Response variable
x	Optional covariate to order the response variable after
w	Optional weight vector

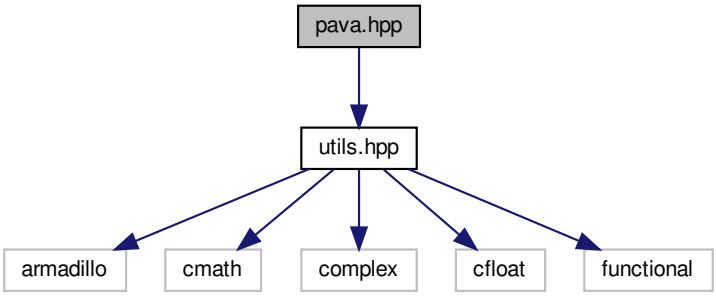
Definition at line 22 of file pava.cpp.

6.12 pava.hpp File Reference

Pooled adjacent violator algorithm.

```
#include "utils.hpp"
```

Include dependency graph for pava.hpp:



Functions

- arma::mat target::pava (arma::vec y, const arma::vec &x=arma::vec(), arma::vec w=arma::vec())

  Weighted Pooled Adjacent Violator Algorithm.

6.12.1 Detailed Description

Pooled adjacent violator algorithm.

Author

Klaus K. Holst

Copyright

2020-2021, Klaus Kähler Holst

6.12.2 Function Documentation

6.12.2.1 pava()

```
arma::mat target::pava (
            arma::vec y,
            const arma::vec & x,
            arma::vec w )
```

Weighted Pooled Adjacent Violator Algorithm.

Pooled Adjacent Violator Algorithm for

Parameters

y	Response variable
x	Optional covariate to order the response variable after
w	Optional weight vector

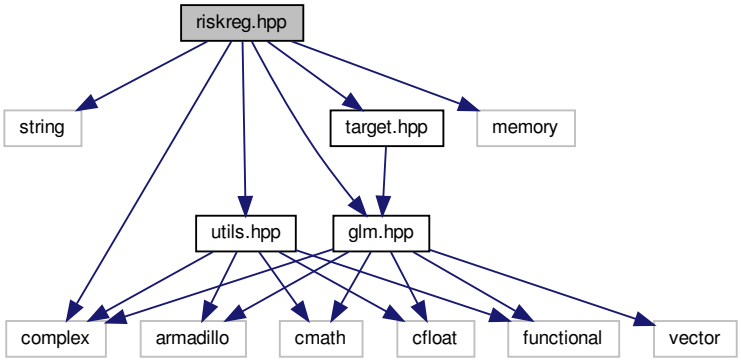
Definition at line 22 of file pava.cpp.

6.13 riskreg.hpp File Reference

Risk regression model.

```
#include <string>
#include <complex>
#include <memory>
#include "target.hpp"
#include "glm.hpp"
#include "utils.hpp"
```

Include dependency graph for riskreg.hpp:



Classes

- class RiskReg

Enumerations

- enum **Data** {
  Y, **A**, **X1**, **X2**,
  X3, **W** }

6.13.1 Detailed Description

Risk regression model.

Author

Klaus K. Holst

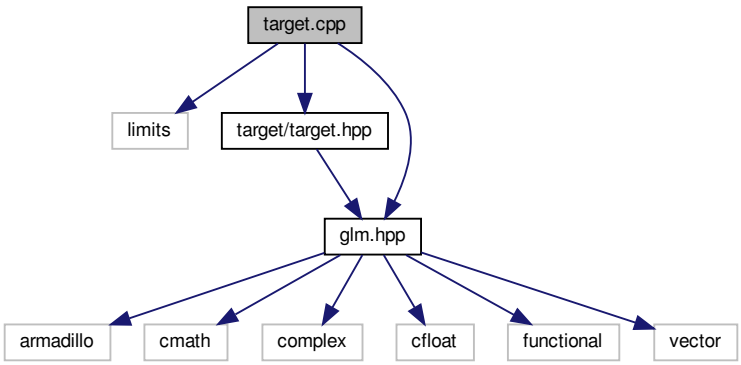
Copyright

6.14 target.cpp File Reference

Classes for targeted inference models.

```
#include <limits>
#include <target/target.hpp>
#include <target/glm.hpp>
```

Include dependency graph for target.cpp:



Functions

- template<typename T >
  arma::Mat< T > target::rd2prob (const arma::Col< T > &rd, const arma::Col< T > &op)

  rd2prob - Computes risk probabilities given exposure 0 or 1.
- template<typename T >
  arma::Mat< T > target::rr2prob (const arma::Col< T > &rr, const arma::Col< T > &op)

  rr2prob - Computes risk probabilities given exposure 0 or 1.

6.14.1 Detailed Description

Classes for targeted inference models.

Author

Klaus K. Holst

Copyright

2019-2022, Klaus Kähler Holst

Includes implementation of risk-difference estimator with nuisance model for log odds-product (MLE and DR estimator).

6.14.2 Function Documentation

6.14.2.1 rd2prob()

```
template<typename T >
arma::Mat< T > target::rd2prob (
             const arma::Col< T > & rd,
             const arma::Col< T > & op )
```

rd2prob - Computes risk probabilities given exposure 0 or 1.

Parameters

rd	- Risk difference vector
op	- Odds-product vetor

Returns

arma::Mat<T> with two columns with probabilities given exposure 0 or 1

Definition at line 307 of file target.cpp.

6.14.2.2 rr2prob()

```
template<typename T >
arma::Mat< T > target::rr2prob (
             const arma::Col< T > & rr,
             const arma::Col< T > & op )
```

rr2prob - Computes risk probabilities given exposure 0 or 1.

Parameters

rr	- Relative risk vector
op	- Odds-product vetor

Returns

arma::Mat<T> with two columns with probabilities given exposure 0 or 1

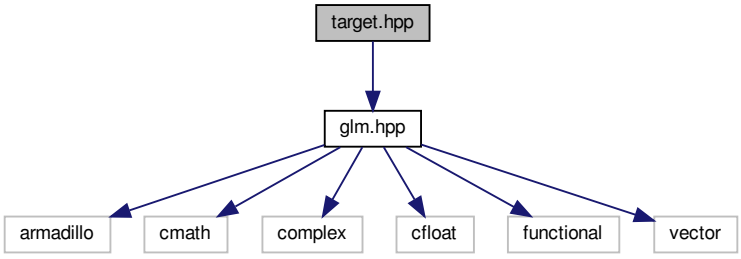
Definition at line 328 of file target.cpp.

6.15 target.hpp File Reference

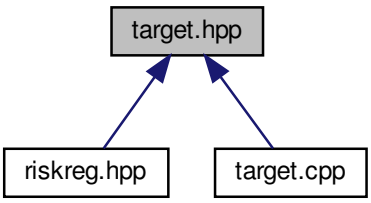
Classes for targeted inference models.

```
#include "glm.hpp"
```

Include dependency graph for target.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class target::Target< T >
- class target::TargetBinary< T >
- class target::RD< T >
- class target::RR< T >
- class target::ACE

Functions

- template<typename T >
  arma::Mat< T > target::rd2prob (const arma::Col< T > &rd, const arma::Col< T > &op)

  rd2prob - Computes risk probabilities given exposure 0 or 1.
- template<typename T >
  arma::Mat< T > target::rr2prob (const arma::Col< T > &rr, const arma::Col< T > &op)

  rr2prob - Computes risk probabilities given exposure 0 or 1.

6.15.1 Detailed Description

Classes for targeted inference models.

Author

Klaus K. Holst

Copyright

2019-2021, Klaus Kähler Holst

6.15.2 Function Documentation

6.15.2.1 rd2prob()

```
template<typename T >
arma::Mat< T > target::rd2prob (
            const arma::Col< T > & rd,
            const arma::Col< T > & op )
```

rd2prob - Computes risk probabilities given exposure 0 or 1.

Parameters

rd	- Risk difference vector
op	- Odds-product vetor

Returns

arma::Mat<T> with two columns with probabilities given exposure 0 or 1

Definition at line 307 of file target.cpp.

6.15.2.2 rr2prob()

```
template<typename T >
arma::Mat< T > target::rr2prob (
            const arma::Col< T > & rr,
            const arma::Col< T > & op )
```

rr2prob - Computes risk probabilities given exposure 0 or 1.

Parameters

rr	- Relative risk vector
op	- Odds-product vetor

Returns

arma::Mat<T> with two columns with probabilities given exposure 0 or 1

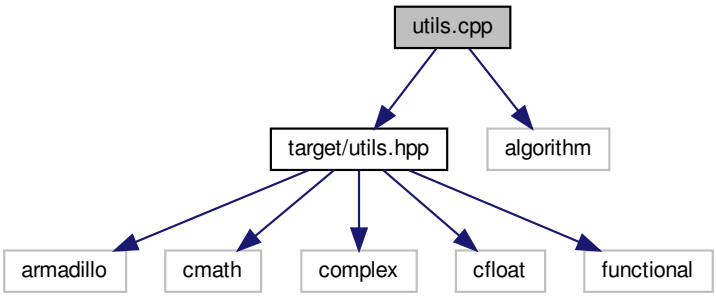
Definition at line 328 of file target.cpp.

6.16 utils.cpp File Reference

Various utility functions and constants.

```
#include <target/utils.hpp>
#include <algorithm>
```

Include dependency graph for utils.cpp:



Functions

- arma::mat **target::deriv** (cx_func f, arma::vec theta)
- arma::umat **target::clusterid** (const arma::uvec &id)
- arma::mat **target::groupsum** (const arma::mat &x, const arma::uvec &cluster, bool reduce=true)
- arma::mat **target::interpolate** (const arma::mat &input, double tau, bool locf=false)
- void **target::fastpattern** (const arma::umat &y, arma::umat &pattern, arma::uvec &group, unsigned categories=2)
- arma::umat **target::fastapprox** (arma::vec time, const arma::vec &newtime, bool equal=false, unsigned type=0)
- double **target::SupTest** (const arma::vec &D)
- double **target::L2Test** (const arma::vec &D, const arma::vec &t)
- double **target::CramerVonMises** (const arma::vec &x, arma::vec G)

6.16.1 Detailed Description

Various utility functions and constants.

Author

Klaus K. Holst

Copyright

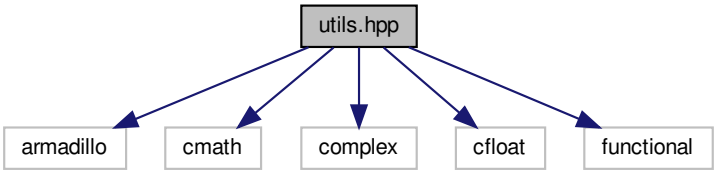
2020-2022, Klaus Kähler Holst

6.17 utils.hpp File Reference

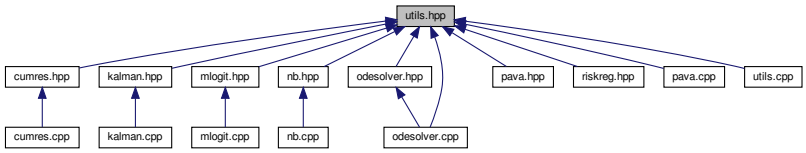
Various utility functions and constants.

```
#include <armadillo>
#include <cmath>
#include <complex>
#include <cfloat>
#include <functional>
```

Include dependency graph for utils.hpp:



This graph shows which files directly or indirectly include this file:



Typedefs

- using **target::cx_dbl** = std::complex< double >
- using **target::cx_func** = std::function< arma::cx_mat(arma::cx_vec theta)>

Functions

- arma::mat **target::deriv** (cx_func f, arma::vec theta)
- arma::umat **target::clusterid** (const arma::uvec &id)
- arma::mat **target::groupsum** (const arma::mat &x, const arma::uvec &cluster, bool reduce=true)
- arma::mat **target::interpolate** (const arma::mat &input, double tau, bool locf=false)
- void **target::fastpattern** (const arma::umat &y, arma::umat &pattern, arma::uvec &group, unsigned categories=2)
- arma::umat **target::fastapprox** (arma::vec time, const arma::vec &newtime, bool equal=false, unsigned type=0)
- double **target::SupTest** (const arma::vec &D)
- double **target::L2Test** (const arma::vec &D, const arma::vec &t)
- double **target::CramerVonMises** (const arma::vec &x, const arma::vec &G)

Variables

- arma::mat const **target::EmptyMat** = arma::mat()
- arma::vec const **target::EmptyVec** = arma::vec()
- const char * **target::COL_RESET** = "\x1b[0m"
- const char * **target::COL_DEF** = "\x1b[39m"
- const char * **target::BLACK** = "\x1b[30m"
- const char * **target::RED** = "\x1b[31m"
- const char * **target::MAGENTA** = "\x1b[35m"
- const char * **target::YELLOW** = "\x1b[33m"
- const char * **target::GREEN** = "\x1b[32m"
- const char * **target::BLUE** = "\x1b[34m"
- const char * **target::CYAN** = "\x1b[36m"
- const char * **target::WHITE** = "\x1b[37m"
- const char * **target::GRAY** = "\x1b[90m"
- const char * **target::LRED** = "\x1b[91m"
- const char * **target::LGREEN** = "\x1b[92m"
- const char * **target::LYELLOW** = "\x1b[93m"
- const char * **target::LBLUE** = "\x1b[94m"
- const char * **target::LMAGENTA** = "\x1b[95m"
- const char * **target::LCYAN** = "\x1b[96m"
- const char * **target::LWHITE** = "\x1b[97m"

6.17.1 Detailed Description

Various utility functions and constants.

Author

Klaus K. Holst

Copyright

Index

1.6 References

BIBLIOGRAPHY

[Bang_2005] H. Bang, & J. M. Robins (2005). Doubly robust estimation in missing data and causal inference models. Biometrics, 61(4), 962–973. http://dx.doi.org/10.1111/j.1541-0420.2005.00377.x

[Tsiatis_2006] A. Tsiatis (2006). Semiparametric theory and missing data. Springer New York.

[Richardson_2018] T. S. Richardson, J. M. Robins, & L. Wang (2017). On modeling and estimation for the relative risk and risk difference. Journal of the American Statistical Association, 112(519), 1121–1130. http://dx.doi.org/10.1080/01621459.2016.1192546

PYTHON MODULE INDEX

t

INDEX