

Package ‘adaptMT’

July 22, 2025

Type Package

Title Adaptive P-Value Thresholding for Multiple Hypothesis Testing
with Side Information

Version 1.0.0

Maintainer Lihua Lei <lihua.lei@berkeley.edu>

Description Implementation of adaptive p-value thresholding (AdaPT), including both a framework that allows the user to specify any algorithm to learn local false discovery rate and a pool of convenient functions that implement specific algorithms. See Lei, Lihua and Fithian, William (2016) <[doi:10.48550/arXiv.1609.06035](https://doi.org/10.48550/arXiv.1609.06035)>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

URL <https://arxiv.org/abs/1609.06035>,
<https://github.com/lihualai71/adaptMT>

BugReports <https://github.com/lihualai71/adaptMT/issues>

Suggests glmnet, HDtweedie, mgcv, splines, testthat, knitr, rmarkdown,
dplyr

RoxygenNote 6.0.1

Imports methods

VignetteBuilder knitr

NeedsCompilation no

Author Lihua Lei [aut, cre]

Repository CRAN

Date/Publication 2018-07-31 12:00:03 UTC

Contents

adapt	2
adapt_gam	5
adapt_glm	6
adapt_glmnet	8
corr_lfdr	9
ctgm_lfdr	10
estrogen	12
gen_adapt_model	13
gen_exp_family	15
plot_1d	16
plot_2d	17

Index	19
--------------	-----------

adapt	<i>Adaptive P-value Thresholding</i>
-------	--------------------------------------

Description

adapt is a framework allowing for arbitrary exponential families for computing E-steps and arbitrary algorithms for fitting M-steps.

Usage

```
adapt(x, pvals, models, dist = beta_family(), s0 = rep(0.45, length(pvals)),
      alphas = seq(0.01, 1, 0.01), params0 = list(pix = NULL, mux = NULL),
      nfits = 20, nms = 1, niter_fit = 10, tol = 1e-04, niter_ms = 20,
      cr = "BIC", verbose = list(print = TRUE, fit = FALSE, ms = TRUE))
```

Arguments

x	covariates (i.e. side-information). Should be compatible to models. See Details
pvals	a vector of values in [0, 1]. P-values
models	an object of class "adapt_model" or a list of objects of class "adapt_model". See Details
dist	an object of class "gen_exp_family". beta_family() as default
s0	a vector of values in [0, 0.5). Initial threshold.
alphas	a vector of values in (0, 1). Target FDR levels.
params0	a list in the form of list(pix = , mux =). Initial guess of pi(x) and mu(x). NULL as default
nfits	a positive integer. Number of model-fitting steps. See Details
nms	a non-negative integer. Number of model selection steps. See Details
niter_fit	a positive integer. Number of EM iterations in model fitting

<code>tol</code>	a positive scalar. EM algorithm stops when $\pi(x)$ and $\mu(x)$ in consecutive steps differ by at most 'tol' for each element
<code>niter_ms</code>	a positive integer. Number of EM iterations in model selection
<code>cr</code>	a string. The criterion for model selection with BIC as default. Also support AIC, AICC and HIC
<code>verbose</code>	a list of logical values in the form of <code>list(print = , fit = , ms =)</code> . Each element indicates whether the relevant information is outputted to the console. See Details

Details

`x` should have a type compatible to the fitting functions in `models`. For GLM and GAM, `x` should be a `data.frame`. For `glmnet`, `x` should be a matrix.

`models` could either be an `adapt_model` object, if a single model is used, or a list of `adapt_model` objects, each of which corresponding to a model. Each element should be generated by [gen_adapt_model](#). For `glm/gam/glmnet`, one can use the shortcut by running [gen_adapt_model](#) with `name = "glm"` or `"gam"` or `"glmnet"` but without specifying `pifun`, `mufun`, `pifun_init` and `mufun_init`. See examples below.

`nfits` is the number of model fitting steps plus `nms`, the model selection steps, if `models` contains multiple `adapt_model` objects. Suppose M is the number of masked p-values at the initial step, then the model is updated at the initial step and at every time when $\lceil M/nfits \rceil$ more p-values are revealed. If `nms` > 0, model selection is performed at the initial step and at every time when $\lceil M/nms \rceil$ more p-values are revealed. Between two consecutive model selection steps, the model selected from the last step is used for model fitting. For example, when $M = 10000$, `nfits` = 10 and `nms` = 2, model selection will be performed at the initial step and when 5000 p-values are revealed, while the model fitting will be performed when 1000, 2000, 3000, 4000, 6000, 7000, 8000, 9000 p-values are revealed.

`verbose` has three elements: `print`, `fit` and `ms`. If `print = TRUE`, the progress of the main procedure is outputted to the console, in the form of "alpha = 0.05: FDP_{hat} 0.0333, Number of Rej. 30" (where the numbers are made up for illustration). If `fit = TRUE`, a progress bar for the model fitting is outputted to the console. Similarly, if `ms = TRUE`, a progress bar for the model selection is outputted to the console.

For ultra-large scale problems ($n > 10^5$), it is recommended to keep `alphas` short because the output `s` is of size $n \times \text{length}(\text{alphas})$.

The output `qvals` gives the q-values of each hypothesis. `qvals[i]` is defined as the minimum target FDR level such that `pvals[i]` is rejected. For hypotheses with p-values above `s0`, the q-values are set to be `Inf` because they are never rejected by AdaPT for whatever alpha.

The output `order` gives the order of (the indices of) p-values being revealed, i.e. being in the region $(s, 1-s)$. The latter hypotheses appeared in order have smaller q-values (i.e. are more likely to be rejected).

Value

<code>nrejs</code>	a vector of integers. Number of rejections for each alpha
<code>rejs</code>	a list of vector of integers. The set of indices of rejections for each alpha

s	a matrix of size $\text{length}(\text{pvals}) \times \text{length}(\text{alphas})$. Threshold curves for each alpha
params	a list. Each element is a list in the form of <code>list(pix = , mux = , alpha = , nmask =)</code> , recording the parameter estimates, the achieved alpha and the number of masked p-values. To avoid massive storage cost, it only contains the information when a new target FDR level is achieved. As a result, it might be shorter than <code>nfits</code> .
qvals	a vector of values in $[0, 1] \cup \text{Inf}$. Q-values. See Details
order	a permutation of $1 : \text{length}(\text{pvals})$. Indices of hypotheses arranged in the order of reveal. See Details
alphas	same as the input alphas
dist	same as the input dist
models	a list of <code>adapt_model</code> objects of length <code>params</code> . The model used in each fitting step. As in <code>params</code> , it only contains the model when a new target FDR level is achieved and each element corresponds to an element of <code>params</code> .
info	a list of length <code>nfits</code> . Each element is a list recording extra information in each fitting step, e.g. degree of freedom (<code>df</code>) and variable importance (<code>vi</code>). As in <code>params</code> , it only contains the model information when a new target FDR level is achieved and each element corresponds to an element of <code>params</code> .
args	a list including the other inputs <code>nfits</code> , <code>nms</code> , <code>niter_fit</code> , <code>niter_ms</code> , <code>tol</code> , <code>cr</code>
.	.

Examples

```
# Load estrogen data
data(estrogen)
pvals <- as.numeric(estrogen$pvals)
x <- data.frame(x = as.numeric(estrogen$ord_high))
dist <- beta_family()

# Subsample the data for convenience
inds <- (x$x <= 5000)
pvals <- pvals[inds]
x <- x[inds,,drop = FALSE]

# Generate models for function adapt
library("splines")
formulas <- paste0("ns(x, df = ", 6:10, ")")
models <- lapply(formulas, function(formula){
  piargs <- muargs <- list(formula = formula)
  gen_adapt_model(name = "glm", piargs = piargs, muargs = muargs)
})

# Run adapt
res <- adapt(x = x, pvals = pvals, models = models,
            dist = dist, nfits = 10)
```

Description

adapt_gam is a wrapper of [adapt](#) that fits $\pi(x)$ and $\mu(x)$ by [gam](#) from [mgcv](#) package.

Usage

```
adapt_gam(x, pvals, pi_formulas, mu_formulas, piargs = list(),
          muargs = list(), dist = beta_family(), s0 = rep(0.45, length(pvals)),
          alphas = seq(0.01, 1, 0.01), ...)
```

Arguments

x	covariates (i.e. side-information). Should be compatible to models. See Details
pvals	a vector of values in [0, 1]. P-values
pi_formulas	a vector/list of strings/formulas. Formulas for fitting $\pi(x)$ by gam. See Details
mu_formulas	a vector/list of strings/formulas. Formulas for fitting $\mu(x)$ by gam. See Details
piargs	a list. Other arguments passed to gam for fitting $\pi(x)$
muargs	a list. Other arguments passed to gam for fitting $\mu(x)$
dist	an object of class " gen_exp_family ". beta_family() as default
s0	a vector of values in [0, 0.5). Initial threshold.
alphas	a vector of values in (0, 1). Target FDR levels.
...	other arguments passed to adapt (except models)

Details

pi_formulas and mu_formulas can either be a list or a vector with each element being a string or a formula. For instance, suppose x has a single column with name x1, the following five options are valid for the same inputs ([ns](#) forms a spline basis with df knots and [s](#) forms a spline basis with knots automatically selected by generalized cross-validation):

1. c("x1", "ns(x1, df = 8)", "s(x1)");
2. c("~ x1", "~ ns(x1, df = 8)", "s(x1)");
3. list("x1", "ns(x1, df = 8)", "s(x1)");
4. list("~ x1", "~ ns(x1, df = 8)", "s(x1)");
5. list(~ x1, ~ ns(x1, df = 8), s(x1))

There is no need to specify the name of the response variable, as this is handled in the function.

When x has a few variables, it is common to use non-parametric GLM by replacing x by a spline basis of x. In this case, [ns](#) from [library\(splines\)](#) package or [s](#) from [mgcv](#) package are suggested. When [s](#) (from [mgcv](#) package) is used, it is treated as a single model because the knots will be selected automatically.

See Also

[adapt](#), [adapt_glm](#), [adapt_glmnet](#), [gam](#), [ns](#), [s](#)

Examples

```
# Generate a 2-dim x
n <- 400
x1 <- x2 <- seq(-100, 100, length.out = 20)
x <- expand.grid(x1, x2)
colnames(x) <- c("x1", "x2")

# Generate p-values (one-sided z test)
# Set all hypotheses in the central circle with radius 30 to be
# non-nulls. For non-nulls,  $z \sim N(2,1)$  and for nulls,  $z \sim N(0,1)$ .
H0 <- apply(x, 1, function(coord){sum(coord^2) < 900})
mu <- ifelse(H0, 2, 0)
set.seed(0)
zvals <- rnorm(n) + mu
pvals <- 1 - pnorm(zvals)

# Run adapt_glm with a 2d spline basis
library("mgcv")
formula <- "s(x1, x2)"
dist <- beta_family()
res <- adapt_glm(x = x, pvals = pvals, pi_formulas = formula,
                mu_formulas = formula, dist = dist, nfits = 5)
```

adapt_glm

Adaptive P-value Thresholding with Generalized Linear Models

Description

adapt_glm is a wrapper of [adapt](#) that fits $\pi(x)$ and $\mu(x)$ by [glm](#).

Usage

```
adapt_glm(x, pvals, pi_formulas, mu_formulas, dist = beta_family(),
          s0 = rep(0.45, length(pvals)), alphas = seq(0.01, 1, 0.01),
          piargs = list(), muargs = list(), ...)
```

Arguments

x covariates (i.e. side-information). Should be compatible to models. See Details

pvals a vector of values in $[0, 1]$. P-values

pi_formulas a vector/list of strings/formulas. Formulas for fitting $\pi(x)$ by [glm](#). See Details

mu_formulas	a vector/list of strings/formulas. Formulas for fitting $\mu(x)$ by glm. See Details
dist	an object of class "gen_exp_family". <code>beta_family()</code> as default
s0	a vector of values in $[0, 0.5)$. Initial threshold.
alphas	a vector of values in $(0, 1)$. Target FDR levels.
piargs	a list. Other arguments passed to glm for fitting $\pi(x)$
muargs	a list. Other arguments passed to glm for fitting $\mu(x)$
...	other arguments passed to <code>adapt</code> (except models)

Details

`pi_formulas` and `mu_formulas` can either be a list or a vector with each element being a string or a formula. For instance, suppose `x` has a single column with name `x1`, the following five options are valid for the same inputs (`ns` forms a spline basis with `df` knots):

1. `c("x1", "ns(x1, df = 8)");`
2. `c("~ x1", "~ ns(x1, df = 8)");`
3. `list("x1", "ns(x1, df = 8)");`
4. `list("~ x1", "~ ns(x1, df = 8)");`
5. `list(~ x1, ~ ns(x1, df = 8))`

There is no need to specify the name of the response variable, as this is handled in the function.

When `x` has a few variables, it is common to use non-parametric GLM by replacing `x` by a spline basis of `x`. In this case, `ns` from `library(splines)` package is suggested.

See Also

[adapt](#), [adapt_gam](#), [adapt_glmnet](#), [glm](#), [ns](#)

Examples

```
# Load estrogen data
data(estrogen)
pvals <- as.numeric(estrogen$pvals)
x <- data.frame(x = as.numeric(estrogen$ord_high))
dist <- beta_family()

# Subsample the data for convenience
inds <- (x$x <= 5000)
pvals <- pvals[inds]
x <- x[inds, , drop = FALSE]

# Run adapt_glm
library("splines")
formulas <- paste0("ns(x, df = ", 6:10, ")")
res <- adapt_glm(x = x, pvals = pvals, pi_formulas = formulas,
                mu_formulas = formulas, dist = dist, nfits = 10)

# Run adapt by manually setting models for glm
```

```

models <- lapply(formulas, function(formula){
  piargs <- muargs <- list(formula = formula)
  gen_adapt_model(name = "glm", piargs = piargs, muargs = muargs)
})
res2 <- adapt(x = x, pvals = pvals, models = models,
             dist = dist, nfits = 10)

# Check equivalence
identical(res, res2)

```

 adapt_glmnet

Adaptive P-value Thresholding with L1/L2 Penalized Generalized Linear Models

Description

adapt_glmnet is a wrapper of [adapt](#) that fits $\pi(x)$ and $\mu(x)$ by [glmnet](#) from glmnet package.

Usage

```

adapt_glmnet(x, pvals, piargs = list(), muargs = list(),
            dist = beta_family(), s0 = rep(0.45, length(pvals)), alphas = seq(0.01,
            1, 0.01), ...)

```

Arguments

x	covariates (i.e. side-information). Should be compatible to models. See Details
pvals	a vector of values in [0, 1]. P-values
piargs	a list. Other arguments passed to glmnet for fitting $\pi(x)$
muargs	a list. Other arguments passed to glmnet for fitting $\mu(x)$
dist	an object of class " gen_exp_family ". beta_family() as default
s0	a vector of values in [0, 0.5). Initial threshold.
alphas	a vector of values in (0, 1). Target FDR levels.
...	other arguments passed to adapt (except models)

Details

adapt_glmnet by default implements LASSO on x with lambda selected by cross-validation. Specify in piargs and muargs if ridge or elastic-net penalty is needed.

See Also

[adapt](#), [adapt_glm](#), [adapt_gam](#), [glmnet](#)

Examples

```

# Generate a 100-dim covariate x
set.seed(0)
m <- 100
n <- 1000
x <- matrix(runif(n * m), n, m)

# Generate the parameters from a conditional two-group
# logistic-Gamma GLM where pi(x) and mu(x) are both
# linear in x. pi(x) has an intercept so that the average
# of pi(x) is 0.3
inv_logit <- function(x) {exp(x) / (1 + exp(x))}
pi1 <- 0.3
beta.pi <- c(3, 3, rep(0, m-2))
beta0.pi <- uniroot(function(b){
  mean(inv_logit(x %>% beta.pi + b)) - pi1
}, c(-100, 100))$root
pi <- inv_logit(x %>% beta.pi + beta0.pi)
beta.mu <- c(2, 2, rep(0, m-2))
beta0.mu <- 0
mu <- pmax(1, x %>% beta.mu + beta0.mu)

# Generate p-values
H0 <- as.logical(ifelse(runif(n) < pi, 1, 0))
y <- ifelse(H0, rexp(n, 1/mu), rexp(n, 1))
pvals <- exp(-y)

# Run adapt_glmnet
res <- adapt_glmnet(x, pvals, s0 = rep(0.15, n), nfits = 5)

```

corr_lfdr

*Quantifying Information Loss of Adaptive P-Value Thresholding***Description**

corr_lfdr computes the oracle local FDR estimate, by using revealing all p-values, and computes the Pearson correlation between it and the estimate within each step of adapt.

Usage

```
corr_lfdr(obj, x, pvals, model = NULL, niter_oracle = 100)
```

Arguments

obj	an 'adapt' object. Output of <code>adapt</code> function
x	covariates (i.e. side-information). Should be compatible to models.
pvals	a vector of values in $[0, 1]$. P-values

`model` an optional argument. If `model = NULL` then the last model in `obj$models` is used for fitting the oracle model (i.e. with all p-values revealed). Otherwise it should be an 'adapt_model' object

`niter_oracle` an positive integer. Number of iterations in EM algorithm

Value

- `corra` vector of values in $[0, 1]$. Pearson correlation of oracle local FDR estimate and the estimates within each step. Each value corresponds to an entry of `obj$params`
- `oracle_lfdra` vector of values in $[0, 1]$. Oracle local FDR estimate
- `lfdra` matrix of values in $[0, 1]$. Local FDR estimates within each step.
- `alphasa` vector of values in $[0, 1]$. The target FDR levels corresponding to each local FDR estimate
- `nmasksa` vector of integers. The number of masked p-values corresponding to each local FDR estimate

Examples

```
# Load estrogen data
data(estrogen)
pvals <- as.numeric(estrogen$pvals)
x <- data.frame(x = as.numeric(estrogen$ord_high))
dist <- beta_family()

# Subsample the data for convenience
inds <- (x$x <= 5000)
pvals <- pvals[inds]
x <- x[inds,,drop = FALSE]

# Run adapt_glm
library("splines")
formulas <- paste0("ns(x, df = ", 6:10, ")")
res <- adapt_glm(x = x, pvals = pvals, pi_formulas = formulas,
                 mu_formulas = formulas, dist = dist, nfits = 10)

# Run corr_lfdr
obj <- corr_lfdr(res, x, pvals)
obj$corr
```

Description

`ctgm_lfdr` computes the oracle local FDR estimate, by using all p-values without masking.

Usage

```
ctgm_lfdr(x, pvals, models, dist = beta_family(), type = c("over", "raw"),
  params0 = list(pix = NULL, mux = NULL), niter = 50, cr = "BIC",
  verbose = TRUE)
```

Arguments

<code>x</code>	covariates (i.e. side-information). Should be compatible to <code>models</code> . See Details
<code>pvals</code>	a vector of values in $[0, 1]$. P-values
<code>models</code>	an object of class "adapt_model" or a list of objects of class "adapt_model". See Details
<code>dist</code>	an object of class "gen_exp_family". <code>beta_family()</code> as default
<code>type</code>	a character. Either "over" or "raw" indicating the type of local FDR estimates. See Details
<code>params0</code>	a list in the form of <code>list(pix = , mux =)</code> . Initial values of $\pi_1(x)$ and $\mu_1(x)$. Both can be set as NULL
<code>niter</code>	a positive integer. Number of EM iterations.
<code>cr</code>	a string. The criterion for model selection with BIC as default. Also support AIC, AICC and HIC
<code>verbose</code>	a logical values in the form of <code>list(fit = , ms =)</code> . Indicate whether the progress of model fitting and model selection is displayed

Details

`ctgm_lfdr` implements the EM algorithm to fit $\pi_1(x)$ and $\mu_1(x)$ on unmasked p-values. Although it is not related to FDR control of AdaPT, it provides useful measures for post-hoc justification and other purposes. For instance, one can use these local FDR estimates for prioritizing the hypotheses if strict FDR control is not required.

In contrast to `adapt`, `ctgm_lfdr` does not guarantee FDR control unless the model is correctly specified. It is recommended to use `ctgm_lfdr` only when FDR control is not required.

`x` should have a type compatible to the fitting functions in `models`. For GLM and GAM, `x` should be a `data.frame`. For `glmnet`, `x` should be a matrix.

`models` could either be an `adapt_model` object, if a single model is used, or a list of `adapt_model` objects, each of which corresponding to a model. Each element should be generated by `gen_adapt_model`. For `glm/gam/glmnet`, one can use the shortcut by running `gen_adapt_model` with `name = "glm"` or `"gam"` or `"glmnet"` but without specifying `pifun`, `mufun`, `pifun_init` and `mufun_init`. See examples below.

When `type = "over"`, it yields a conservative estimate of local FDR

$$lfdr(p) = (1 - \pi_1 + \pi_1 f_1(1)) / (1 - \pi_1 + \pi_1 f_1(p)).$$

When `type = "raw"`, it yields the original local FDR.

$$lfdr(p) = (1 - \pi_1) / (1 - \pi_1 + \pi_1 f_1(p)).$$

The former is shown to be more stable and reliable because the weak identifiability in conditional mixture models.

Value

- lfdra vector of values in $[0, 1]$. Local FDR estimates of each hypothesis.
- modelan adapt_model object. The selected model if multiple models are provided.

Examples

```
# Load estrogen data
data(estrogen)
pvals <- as.numeric(estrogen$pvals)
x <- data.frame(x = as.numeric(estrogen$ord_high))
dist <- beta_family()

# Subsample the data for convenience
inds <- (x$x <= 5000)
pvals <- pvals[inds]
x <- x[inds,,drop = FALSE]

# Generate models for function adapt
library("splines")
formulas <- paste0("ns(x, df = ", 6:10, ")")
models <- lapply(formulas, function(formula){
  piargs <- muargs <- list(formula = formula)
  gen_adapt_model(name = "glm", piargs = piargs, muargs = muargs)
})

# Run ctgm_lfdr with two types of lfdr estimates
res_over <- ctgm_lfdr(x, pvals, models, type = "over")
res_raw <- ctgm_lfdr(x, pvals, models, type = "raw")

# Compare two estimates
par(mfrow = c(2, 1))
hist(res_over$lfdr)
hist(res_raw$lfdr)
```

 estrogen

Gene/Drug response dataset

Description

P-values and ordering of genes drawn from a microarray dataset, consisting of 22283 genes on breast cancer cells in response to estrogen, from NCBI Gene Expression Omnibus (GEO) through 'GEOquery' package, with index "GDS2324".

Usage

```
estrogen
```

Format

An object of class `data.frame` with 22283 rows and 3 columns.

Details

The original dataset "GDS2324" consists of gene expression measurements for $n = 22283$ genes, in response to estrogen treatments in breast cancer cells for five groups of patients, with different dosage levels and 5 trials in each. The task is to identify the genes responding to a low dosage. The p-value for gene i is obtained by a one-sided permutation test which evaluates evidence for a change in gene expression level between the control group (placebo) and the low-dose group. The p-values are then ordered according to permutation t-statistics comparing the control and low-dose data, pooled, against data from a higher dosage (with genes that appear to have a strong response at higher dosages placed earlier in the list).

Two orderings are considered: first, a stronger (more informative) ordering based on a comparison to the highest dosage; and second, a weaker (less informative) ordering based on a comparison to a medium dosage.

The variables are as follows:

- `pvals`. p-values
- `ord_high`. stronger ordering
- `ord_mod`. weaker ordering

The R code to produce the data can be found in `'/extdata/estrogen_get_pvals.R'`.

`gen_adapt_model` *adapt_model Objects for M-steps*

Description

`adapt_model` objects provide the functions and their arguments in computing the M-steps. Each object can be passed to `adapt` as a candidate model.

Usage

```
gen_adapt_model(pifun = NULL, mufun = NULL, pifun_init = NULL,
               mufun_init = NULL, piargs = list(), muargs = list(),
               piargs_init = list(), muargs_init = list(), name = "")
```

Arguments

<code>pifun</code>	a function to fit $\pi(x)$. See Details
<code>mufun</code>	a function to fit $\mu(x)$. See Details
<code>pifun_init</code>	a function to fit $\pi(x)$ at the initial step
<code>mufun_init</code>	a function to fit $\mu(x)$ at the initial step
<code>piargs</code>	a list. Arguments for "pifun". An empty list as default

muargs	a list. Arguments for "mufun". An empty list as default
piargs_init	a list. Arguments for piargs_init. An empty list as default
muargs_init	a list. Arguments for muargs_init. An empty list as default
name	a string. An optional argument for the user-specified name of the model. An empty string as default.

Details

pifun should be in the form of `pifun(formula, data, family, weights, ...)` or `pifun(x, y, family, ...)`. The former includes `glm` and `gam` and the latter includes `glmnet`. The outputs should be in the form of `list(fitv = , info = , ...)` where `fitv` gives the estimate of $\pi(x)$, as a vector with the same order of `x`, and `info` should at least contain a key `df` if model selection is used, i.e. `info = list(df = , ...)`

mufun should be in the form of `pifun(formula, data, family, weights, ...)` or `pifun(x, y, family, weights, ...)`. Note that mufun must take `weights` as an input. The outputs should be in the same form as pifun except that `fitv` should give the estimate of $\mu(x)$.

When pifun / mufun takes the form of `(formula, family, ...)`, `piargs / muargs` should at least contain a key `formula`; when pifun / mufun takes the form of `(x, y, family, ...)`, `piargs / muargs` can be empty.

For `glm/gam/glmnet`, one can use the shortcut by running `gen_adapt_model` with `name = "glm"` or `"gam"` or `"glmnet"` but without specifying `pifun`, `mufun`, `pifun_init` and `mufun_init`. See examples below.

Value

name	same as the input name
algo	a list recording pifun, mufun, pifun_init and mufun_init
args	a list recording piargs, muargs, piargs_init and muargs_init

Examples

```
# Exemplary code to generate 'adapt_model' for logistic-Gamma glm with naive initialization.
# The real implementation in the package is much more complicated.

# pifun as a logistic regression
pifun <- function(formula, data, weights, ...){
  glm(formula, data, weights = weights, family = binomial(), ...)
}
# pifun_init as a constant
pifun_init <- function(x, pvals, s, ...){
  rep(0.1, length(pvals))
}
# mufun as a Gamma GLM
mufun <- function(formula, data, weights, ...){
  glm(formula, data, weights = weights, family = Gamma(), ...)
}
# mufun_init as a constant
mufun_init <- function(x, pvals, s, ...){
```

```

    rep(1.5, length(pvals))
  }

  library("splines") # for using ns() in the formula
  piargs <- list(formula = "ns(x, df = 8)")
  muargs <- list(formula = "ns(x, df = 8)")
  name <- "glm"

  mod <- gen_adapt_model(pifun, mufun, pifun_init, mufun_init,
                        piargs, muargs, name = name)

  mod

  # Using shortcut for GLM. See the last paragraph of Details.
  mod2 <- gen_adapt_model(name = "glm", piargs = piargs, muargs = muargs)
  mod2

```

gen_exp_family

Generate exp_family Objects for Exponential Families

Description

exp_family objects contain all required information in an exponential family to perform the E-step. The exponential function is encoded by

$$h(p; \mu) = \exp\{(\eta(\mu) - \eta(\mu^*))g(p) - (A(\mu) - A(\mu^*))\}$$

where $g(p)$ is an arbitrary transformation, μ is the *mean parameter*, η is the natural parameter, and $A(\mu)$ is the partition function. The extra redundant parameter μ^* is to guarantee that $U([0, 1])$ belongs to the class.

Usage

```

gen_exp_family(g, ginv, eta, mustar, A, name = NULL, family = NULL)

beta_family()

inv_gaussian_family()

```

Arguments

g	a function. An transformation of p-values
ginv	a function. The inverse function of g
eta	a function. The natural parameter as a function of the mean parameter mu
mustar	a scalar. The mean parameter that gives $U([0, 1])$
A	a function. The partition function
name	a string. A name for the family. NULL by default
family	an object of class "family" from stats package. The family used for model fitting in glm , gam , glmnet , etc..

Details

Beta family (`beta_family()`): modeling p-values as Beta-distributed random variables, i.e. $g(p) = -\log(p)$, $\eta(\mu) = -1/\mu$, $\mu^* = 1$, $A(\mu) = \log(\mu)$, `name = "beta"` and `family = Gamma()`. Beta-family is highly recommended for general problems and used as default.

Inverse-gaussian family (`inv_gaussian_family()`): modeling p-values as transformed z-scores, i.e. $g(p) = \Phi^{-1}(p)$ (Φ is the c.d.f. of a standard normal random variable), $\eta(\mu) = \mu$, $\mu^* = 0$, $A(\mu) = \mu^2/2$, `name = "inv_gaussian"` and `family = gaussian()`.

Value

an object of class "exp_family". This includes all inputs and `h`, the density function.

plot_1d

Plotting Functions for AdaPT with 1D Covariates

Description

Plotting the outputs of `adapt` when `x` is 1-dimensional, including threshold curves and level curves of local FDR.

Usage

```
plot_1d_thresh(obj, x, pvals, alpha, title, xlab = "x", xlim = NULL,
  disp_ymax = 0.2, num_yticks = 3, rand_seed_perturb = NA, ...)
```

```
plot_1d_lfdr(obj, x, pvals, alpha, title, xlab = "x", xlim = NULL,
  disp_ymax = 0.2, num_yticks = 3, legend_pos = "topright", ...)
```

Arguments

<code>obj</code>	an 'adapt' object
<code>x</code>	covariates (i.e. side-information). Should be compatible to <code>models</code> and 1-dimensional.
<code>pvals</code>	a vector of values in $[0, 1]$. P-values
<code>alpha</code>	a positive scalar in $(0, 1)$. Target FDR level
<code>title</code>	a string. Title of the figure
<code>xlab</code>	a string. Label of the x-axis
<code>xlim</code>	a vector of length 2. Limits of x-axis
<code>disp_ymax</code>	a positive scalar in $(0, 1]$. Maximum value displayed in the y-axis
<code>num_yticks</code>	a positive integer. Number of ticks in the y-axis
<code>rand_seed_perturb</code>	random seed if jitter is added. NA if no jittering is needed
<code>...</code>	other arguments passed to <code>par</code>
<code>legend_pos</code>	a string. Position of the legend

Examples

```

# Load estrogen data
data(estrogen)
pvals <- as.numeric(estrogen$pvals)
x <- data.frame(x = as.numeric(estrogen$ord_high))
dist <- beta_family()

# Subsample the data for convenience
inds <- (x$x <= 5000)
pvals <- pvals[inds]
x <- x[inds,,drop = FALSE]

# Run adapt_glm
library("splines")
formulas <- paste0("ns(x, df = ", 6:10, ")")
res <- adapt_glm(x = x, pvals = pvals, pi_formulas = formulas,
                 mu_formulas = formulas, dist = dist, nfits = 10)

# Plots
par(mfrow = c(2, 1))
plot_1d_thresh(res, x, pvals, 0.1, "P-value Thresholds (alpha = 0.1)",
              disp_ymax = 0.5)
plot_1d_lfdr(res, x, pvals, 0.1, "Level Curves of lfdr (alpha = 0.1)",
            disp_ymax = 0.5)

```

plot_2d

Plotting Functions for AdaPT with 2D Covariates

Description

Plotting the outputs of `adapt` when `x` is 2-dimensional, including threshold curves and level curves of local FDR.

Usage

```
plot_2d_thresh(obj, x, pvals, alpha, title, xlab = NULL, ylab = NULL,
              keyaxes = list(), ...)
```

```
plot_2d_lfdr(obj, x, pvals, alpha, title, targetp, xlab = NULL, ylab = NULL,
            keyaxes = list(), ...)
```

Arguments

<code>obj</code>	an 'adapt' object
<code>x</code>	covariates (i.e. side-information). Should be compatible to <code>models</code> and 2-dimensional.
<code>pvals</code>	a vector of values in $[0, 1]$. P-values

alpha	a positive scalar in (0, 1). Target FDR level
title	a string. Title of the figure
xlab, ylab	a string. Label of x/y-axis
keyaxes	a list of arguments passed into axis. The graphical setting for the legend bar. An empty list by default
...	other arguments passed to <code>par</code>
targetp	a real in (0, 1). See Details

Details

The breaks in the legend of `plot_2d_thresh` correspond to the maximum, the 95

`plot_2d_lfdr` gives the contour plot of local FDR estimates when all p-values are equal to `targetp`. It is recommended to run `plot_2d_lfdr` for multiple `targetp`'s ranging from 0.001, 0.005, 0.01, 0.05.

Examples

```
# Generate a 2-dim x
n <- 400
x1 <- x2 <- seq(-100, 100, length.out = 20)
x <- expand.grid(x1, x2)
colnames(x) <- c("x1", "x2")

# Generate p-values (one-sided z test)
# Set all hypotheses in the central circle with radius 30 to be
# non-nulls. For non-nulls,  $z \sim N(2,1)$  and for nulls,  $z \sim N(0,1)$ .
H0 <- apply(x, 1, function(coord){sum(coord^2) < 900})
mu <- ifelse(H0, 2, 0)
set.seed(0)
zvals <- rnorm(n) + mu
pvals <- 1 - pnorm(zvals)

# Run adapt_gam with a 2d spline basis
library("mgcv")
formula <- "s(x1, x2)"
dist <- beta_family()
res <- adapt_gam(x = x, pvals = pvals, pi_formulas = formula,
                mu_formulas = formula, dist = dist, nfits = 5)

# Plots
plot_2d_thresh(res, x, pvals, 0.3, "P-value Thresholds (alpha = 0.3)")
plot_2d_lfdr(res, x, pvals, 0.3, "Local FDR Estimates (alpha = 0.3, p = 0.01)", 0.01)
```

Index

* datasets

estrogen, [12](#)

[adapt](#), [2](#), [5–9](#), [13](#)

[adapt_gam](#), [5](#), [7](#), [8](#)

[adapt_glm](#), [6](#), [6](#), [8](#)

[adapt_glmnet](#), [6](#), [7](#), [8](#)

[beta_family](#), [2](#), [5](#), [7](#), [8](#), [11](#)

[beta_family](#) ([gen_exp_family](#)), [15](#)

[corr_lfdr](#), [9](#)

[ctgm_lfdr](#), [10](#)

[estrogen](#), [12](#)

[family](#), [15](#)

[gam](#), [5](#), [6](#), [14](#), [15](#)

[gen_adapt_model](#), [3](#), [11](#), [13](#), [14](#)

[gen_exp_family](#), [2](#), [5](#), [7](#), [8](#), [11](#), [15](#)

[glm](#), [6](#), [7](#), [14](#), [15](#)

[glmnet](#), [8](#), [14](#), [15](#)

[inv_gaussian_family](#) ([gen_exp_family](#)), [15](#)

[ns](#), [5–7](#)

[par](#), [16](#), [18](#)

[plot_1d](#), [16](#)

[plot_1d_lfdr](#) ([plot_1d](#)), [16](#)

[plot_1d_thresh](#) ([plot_1d](#)), [16](#)

[plot_2d](#), [17](#)

[plot_2d_lfdr](#) ([plot_2d](#)), [17](#)

[plot_2d_thresh](#) ([plot_2d](#)), [17](#)

[s](#), [5](#), [6](#)