

Package ‘postcard’

May 9, 2026

Type Package

Title Estimating Marginal Effects with Prognostic Covariate Adjustment

Version 1.1.0

Description Conduct power analyses and inference of marginal effects.

Uses plug-in estimation and influence functions to perform robust inference, optionally leveraging historical data to increase precision with prognostic covariate adjustment. The methods are described in Højbjerg-Frandsen et al. (2025) <[doi:10.48550/arXiv.2503.22284](https://doi.org/10.48550/arXiv.2503.22284)>.

License MIT + file LICENSE

URL <https://novonordisk-opensource.github.io/postcard/>,
<https://github.com/NovoNordisk-OpenSource/postcard>

BugReports <https://github.com/NovoNordisk-OpenSource/postcard/issues>

Imports cli, Deriv, dplyr, earth, generics, gggrid, ggplot2, options, parsnip, rlang, rsample, scales, stats, stringr, tune, utils, workflowsets, xgboost, yardstick

Suggests knitr, LiblineaR, MASS, ranger, rmarkdown, testthat (>= 3.0.0), vdiff, withr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Mathias Lerbech Jeppesen [aut, cre],
Emilie Højbjerg-Frandsen [aut],
Novo Nordisk A/S [cph]

Maintainer Mathias Lerbech Jeppesen <mathiasljeppesen@outlook.com>

Repository CRAN

Date/Publication 2025-09-17 12:50:02 UTC

Contents

| | |
|---------------------------------------|-----------|
| default_learners | 2 |
| fit_best_learner | 3 |
| glm_data | 4 |
| options | 6 |
| power_linear | 7 |
| power_marginaleffect | 10 |
| prog | 14 |
| rctglm | 15 |
| rctglm_methods | 19 |
| rctglm_with_prognosticscore | 21 |
| repeat_power_linear | 24 |
| repeat_power_marginaleffect | 26 |
| Index | 29 |

| | |
|------------------|-----------------------------------|
| default_learners | <i>Creates a list of learners</i> |
|------------------|-----------------------------------|

Description

This function creates a list of learners compatible with the learners argument of [fit_best_learner](#), which is used as the default argument.

Usage

```
default_learners()
```

Value

a named list of learners, where each element consists of a

- model: A parsnip model specification
- grid: A data.frame with columns corresponding to tuning parameters

Examples

```
default_learners()
```

| | |
|------------------|---|
| fit_best_learner | <i>Find the best learner in terms of RMSE among specified learners using cross validation</i> |
|------------------|---|

Description

Find the best learner in terms of RMSE among specified learners using cross validation

Usage

```
fit_best_learner(  
  preproc,  
  data,  
  cv_folds = 5,  
  learners = default_learners(),  
  verbose = options::opt("verbose")  
)
```

Arguments

| | |
|----------|---|
| preproc | A list (preferably named) with preprocessing objects: formulas, recipes, or workflows::workflow_variables() . Passed to workflowsets::workflow_set() . |
| data | A data frame. |
| cv_folds | a numeric with the number of cross-validation folds used when fitting and evaluating models |
| learners | a list (preferably named) containing named lists of elements model and optionally grid. The model element should be a parsnip model specification, which is passed to workflowsets::workflow_set as the model argument, while the grid element is passed as the grid argument of workflowsets::option_add |
| verbose | numeric verbosity level. Higher values means more information is printed in console. A value of 0 means nothing is printed to console during execution (Defaults to 2, overwritable using option 'postcard.verbose' or environment variable 'R_POSTCARD_VERBOSE') |

Details

Ensure data compatibility with the learners.

Value

a trained workflow

See Also

See [rctglm_with_prognosticscore\(\)](#) for a function that utilises this function to perform prognostic covariate adjustment.

Examples

```

# Generate some synthetic 2-armed RCT data along with historical controls
n <- 100
dat_rct <- glm_data(
  Y ~ 1+2*x1+3*a,
  x1 = rnorm(n, 2),
  a = rbinom(n, 1, .5),
  family = gaussian()
)
dat_hist <- glm_data(
  Y ~ 1+2*x1,
  x1 = rnorm(n, 2),
  family = gaussian()
)

# Fit a learner to the historical control data
learners <- list(
  mars = list(
    model = parsnip::set_engine(
      parsnip::mars(
        mode = "regression", prod_degree = 3
      ),
      "earth"
    )
  )
)
fit <- fit_best_learner(
  preproc = list(mod = Y ~ .),
  data = dat_hist,
  learners = learners
)

# Use it fx. to predict the "control outcome" in the 2-armed RCT
predict(fit, new_data = dat_rct)

```

glm_data

Generate data simulated from a GLM

Description

Provide a formula, variables and a family to generate a linear predictor using the formula and provided variables before using the inverse link of the family to generate the GLM modelled mean, μ , which is then used to simulate the response with this mean from the generating function according to the chosen family.

Usage

```
glm_data(formula, ..., family = gaussian(), family_args = NULL)
```

Arguments

| | |
|-------------|---|
| formula | an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details' in the glm documentation. |
| ... | a data.frame with columns corresponding to variables used in formula, a named list of those variables, or individually provided named arguments of variables |
| family | the family of the response. this can be a character string naming a family function, a family function or the result of a call to a family function |
| family_args | a named list with values of arguments passed to family relevant <code>r<family_name></code> function for simulating the data |

Value

a data.frame

Examples

```
# Generate a gaussian response from a single covariate
glm_data(Y ~ 1+2*x1,
         x1 = rnorm(10))

# Generate a gaussian response from a single covariate with non-linear
# effects. Specify that the response should have standard deviation sqrt(3)
glm_data(Y ~ 1+2*log(x1),
         x1 = runif(10, min = 1, max = 10),
         family_args = list(sd = sqrt(3)))

# Generate a negative binomial response
glm_data(Y ~ 1+2*x1-x2,
         x1 = rnorm(10),
         x2 = rgamma(10, shape = 2),
         family = MASS::negative.binomial(2))

# Provide variables as a list/data.frame and pass a link to the negative.binomial
# function
glm_data(resp ~ 1+2*x1-x2,
         data.frame(
           x1 = rnorm(10),
           x2 = rgamma(10, shape = 2)
         ),
         family = MASS::negative.binomial(2),
         family_args = list(link = "identity"))
```

| | |
|---------|-------------------------|
| options | <i>postcard Options</i> |
|---------|-------------------------|

Description

Internally used, package-specific options. All options will prioritize R options() values, and fall back to environment variables if undefined. If neither the option nor the environment variable is set, a default value is used.

Arguments

verbose numeric verbosity level. Higher values means more information is printed in console. A value of 0 means nothing is printed to console during execution (Defaults to 2, overwritable using option 'postcard.verbose' or environment variable 'R_POSTCARD_VERBOSE')

Checking Option Values

Option values specific to postcard can be accessed by passing the package name to env.

```
options::opts(env = "postcard")
```

```
options::opt(x, default, env = "postcard")
```

Options

verbose numeric verbosity level. Higher values means more information is printed in console. A value of 0 means nothing is printed to console during execution

default: 2

option: postcard.verbose

envvar: R_POSTCARD_VERBOSE (evaluated if possible, raw string otherwise)

See Also

options getOption Sys.setenv Sys.getenv

power_linear

*Power and sample size estimation for linear models***Description**

variance_ancova provides a convenient function for estimating a variance to use for power and sample size approximation.

The power_gs and samplesize_gs functions calculate the Guenther-Schouten power approximation for ANOVA or ANCOVA. The approximation is based in (Guenther WC. Sample Size Formulas for Normal Theory T Tests. The American Statistician. 1981;35(4):243–244) and (Schouten HJA. Sample size formula with a continuous outcome for unequal group sizes and unequal variances. Statistics in Medicine. 1999;18(1):87–91).

The function power_nc calculates the power for ANOVA or ANCOVA based on the non-centrality parameter and the exact t-distributions.

See more details about each function in Details and in sections after Value.

Usage

```
variance_ancova(formula, data, inflation = 1, deflation = 1, ...)
```

```
power_gs(variance, ate, n, r = 1, margin = 0, alpha = 0.05, ...)
```

```
samplesize_gs(variance, ate, r = 1, margin = 0, power = 0.9, alpha = 0.05, ...)
```

```
power_nc(variance, df, ate, n, r = 1, margin = 0, alpha = 0.05, ...)
```

Arguments

| | |
|-----------|--|
| formula | an object of class "formula" (or one that can be coerced to that class): a symbolic description used in <code>stats::model.frame()</code> to create a data.frame with response and covariates. This data.frame is used to estimate the R^2 , which is then used to find the variance. See more in details. |
| data | a data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame), containing the variables in formula. Neither a matrix nor an array will be accepted. |
| inflation | a numeric to multiply the marginal variance of the response by. Default is 1 which estimates the variance directly from data. Use values above 1 to obtain a more conservative estimate of the marginal response variance. |
| deflation | a numeric to multiply the R^2 by. Default is 1 which means the estimate of R^2 is unchanged. Use values below 1 to obtain a more conservative estimate of the coefficient of determination. See details about how R^2 related to the estimation. |
| ... | Not currently used. Here to accomodate implementation of ... for the <code>repeat_power_linear()</code> function. |
| variance | a numeric variance to use for the approximation. See more details in documentation sections of each power approximating function. |

| | |
|--------|---|
| ate | a numeric minimum effect size that we should be able to detect. |
| n | a numeric with number of participants in total. From this number of participants in the treatment group is $n1 = (r/(1+r))n$ and the control group is $n0 = (1/(1+r))n$. |
| r | a numeric allocation ratio $r = n1/n0$. For one-to-one randomisation $r=1$. |
| margin | a numeric superiority margin (for non-inferiority margin, a negative value can be provided). |
| alpha | a numeric significance level. The critical value used for testing corresponds to using the significance level for a two-sided test. I.e. we use the quantile $1 - \alpha/2$ of the null distribution as the critical value. |
| power | a numeric giving the desired power when calculating the sample size |
| df | a numeric degrees of freedom to use in the t-distribution. |

Details

This details section provides information about relation between arguments to functions and the formulas described in sections below for each power approximation formula.

Note that all entities that carry the same name as an argument and in the formula will not be mentioned below, as they are obviously linked (n, r, alpha)

- ate: $\beta_1 - \beta_0$
- margin: Δ_s
- variance: $\hat{\sigma}^2(1 - \hat{R}^2)$

Finding the variance to use for approximation:

The `variance_ancova` function estimates $\sigma^2(1 - R^2)$ in data and returns it as a numeric that can be passed directly as the variance in `power_gs`. Corresponds to estimating the power from using an `lm` with the same formula as specified in `variance_ancova`.

The user can estimate the variance any way they see fit.

Value

All functions return a numeric. `variance_ancova` returns a numeric with a variance estimated from data to used for power estimation and sample size estimation. `power_xx` and `samplesize_xx` functions return a numeric with the power or sample size approximation.

Guenther-Schouten power approximation

The estimation formula in the case of an ANCOVA model with multiple covariate adjustment is (see description for reference):

$$n = \frac{(1+r)^2}{r} \frac{(z_{1-\alpha/2} + z_{1-\beta})^2 \hat{\sigma}^2 (1 - \hat{R}^2)}{(\beta_1 - \beta_0 - \Delta_s)^2} + \frac{(z_{1-\alpha/2})^2}{2}$$

where $\hat{R}^2 = \frac{\hat{\sigma}_{XY}^\top \hat{\Sigma}_X^{-1} \hat{\sigma}_{XY}}{\hat{\sigma}^2}$, we denote by $\hat{\sigma}^2$ an estimate of the variance of the outcome, $\hat{\Sigma}_X$ and estimate of the covariance matrix of the covariates, and $\hat{\sigma}_{XY}$ a p -dimensional column vector consisting of an estimate of the covariance between the outcome variable and each covariate. In the univariate case R^2 is replaced by ρ^2

Power approximation using non-centrality parameter

The prospective power estimations are based on (Kieser M. Methods and Applications of Sample Size Calculation and Recalculation in Clinical Trials. Springer; 2020). The ANOVA power is calculated based on the non-centrality parameter given as

$$nc = \sqrt{\frac{r}{(1+r)^2}} \cdot n \cdot \frac{\beta_1 - \beta_0 - \Delta_s}{\sigma},$$

where we denote by σ^2 the variance of the outcome, such that the power can be estimated as

$$1 - \beta = 1 - F_{t,n-2,nc} \left(F_{t,n-2,0}^{-1}(1 - \alpha/2) \right).$$

The power of ANCOVA with univariate covariate adjustment and no interaction is calculated based on the non-centrality parameter given as

$$nc = \sqrt{\frac{rn}{(1+r)^2}} \frac{\beta_1 - \beta_0 - \Delta_s}{\sigma \sqrt{1 - \rho^2}},$$

such that the power can be estimated as

$$1 - \beta = 1 - F_{t,n-3,nc} \left(F_{t,n-3,0}^{-1}(1 - \alpha/2) \right).$$

The power of ANCOVA with either univariate covariate adjustment and interaction or multiple covariate adjustment with or without interaction is calculated based on the non-centrality parameter given as

$$nc = \frac{\beta_1 - \beta_0 - \Delta_s}{\sqrt{\left(\frac{1}{n_1} + \frac{1}{n_0} + X_d^\top ((n-2)\Sigma_X)^{-1} X_d \right) \sigma^2 (1 - \widehat{R}^2)}}.$$

where $X_d = (\overline{X}_1^1 - \overline{X}_0^1, \dots, \overline{X}_1^p - \overline{X}_0^p)^\top$, $\widehat{R}^2 = \frac{\widehat{\sigma}_{XY}^\top \widehat{\Sigma}_X^{-1} \widehat{\sigma}_{XY}}{\widehat{\sigma}^2}$, we denote by $\widehat{\sigma}^2$ an estimate of the variance of the outcome, $\widehat{\Sigma}_X$ and estimate of the covariance matrix of the covariates, and $\widehat{\sigma}_{XY}$ a p -dimensional column vector consisting of an estimate of the covariance between the outcome variable and each covariate. Since we are in the case of randomized trials the expected difference between the covariate values between the two groups is 0. Furthermore, the elements of Σ_X^{-1} will be small, unless the variances are close to 0, or the covariates exhibit strong linear dependencies, so that the correlations are close to 1. These scenarios are excluded since they could lead to potentially serious problems regarding inference either way. These arguments are used by Zimmermann et. al (Zimmermann G, Kieser M, Bathke AC. Sample Size Calculation and Blinded Recalculation for Analysis of Covariance Models with Multiple Random Covariates. Journal of Biopharmaceutical Statistics. 2020;30(1):143–159.) to approximate the non-centrality parameter as in the univariate case where ρ^2 is replaced by R^2 .

Then the power for ANCOVA with d degrees of freedom can be estimated as

$$1 - \beta = 1 - F_{t,d,nc} \left(F_{t,d,0}^{-1}(1 - \alpha/2) \right).$$

See Also

See [power_marginaleffect](#) for a power approximation function that works for a larger class of models.

Examples

```
# Generate a data set to use as an example
n_train <- 1e3
n_test <- 100

dat_gaus <- glm_data(Y ~ 1+2*X1-X2+3*A,
                    X1 = rnorm(n_train),
                    X2 = rgamma(n_train, shape = 2),
                    A = rbinom(n_train, size = 1, prob = 0.5),
                    family = gaussian())

# Approximate the power using no adjustment covariates
va_nocov <- var(dat_gaus$Y)
power_gs(n = n_test, variance = va_nocov, ate = 1)

# Approximate the power with a model adjusting for both variables in the
# data generating process

## First estimate the variance  $\sigma^2 * (1-R^2)$ 
va_cov <- variance_ancova(Y ~ X1 + X2 + A, dat_gaus)
## Then estimate the power using this variance
power_gs(n = n_test, variance = va_cov, ate = 1.8, margin = 1, r = 2)

# Approximate the sample size needed to obtain 90% power with same model as
# above
samplesize_gs(
  variance = va_cov, ate = 1.8, power = 0.9, margin = 1, r = 2
)

# No adjustment covariates
power_nc(n = n_test, variance = va_nocov, df = 199, ate = 1)
# Adjusting for all covariates in data generating process
power_nc(n = n_test, variance = va_cov, df = 196, ate = 1.8, margin = 1, r = 2)
```

power_marginaleffect *Power approximation for estimating marginal effects in GLMs*

Description

The functions implements the algorithm for power estimation described in [Powering RCTs for marginal effects with GLMs using prognostic score adjustment](#) by Højbjerg-Frandsen et. al (2025). See a bit more context in details and all details in reference.

Usage

```
power_marginaleffect(
  response,
  predictions,
  target_effect,
  exposure_prob,
  var1 = NULL,
  kappa1_squared = NULL,
  estimand_fun = "ate",
  estimand_fun_deriv0 = NULL,
  estimand_fun_deriv1 = NULL,
  inv_estimand_fun = NULL,
  margin = estimand_fun(1, 1),
  alpha = 0.05,
  tolerance = 0.001,
  verbose = options::opt("verbose"),
  ...
)
```

Arguments

| | |
|---------------------|---|
| response | a vector of mode numeric with the response variable from comparator participants |
| predictions | a vector of mode numeric with predictions of the response |
| target_effect | a numeric minimum effect size that we should be able to detect. See more in details. |
| exposure_prob | a numeric with the probability of being in "group 1" (rather than group 0). See more in details. |
| var1 | a numeric variance of the potential outcome corresponding to group 1, or a function with a single argument meant to obtain var1 as a transformation of the variance of the potential outcome corresponding to group 0. See more in details. |
| kappa1_squared | a numeric mean-squared error from predicting potential outcome corresponding to group 1, or a function with a single arguments meant to obtain kappa1_squared as a transformation of the MSE in group 0. See more in details. |
| estimand_fun | a function with arguments psi1 and psi0 specifying the estimand. Alternative, specify "ate" or "rate_ratio" as a character to use one of the default estimand functions. See more details in the "Estimand" section of rctglm . |
| estimand_fun_deriv0 | a function specifying the derivative of estimand_fun wrt. psi0. As a default the algorithm will use symbolic differentiation to automatically find the derivative from estimand_fun |
| estimand_fun_deriv1 | a function specifying the derivative of estimand_fun wrt. psi1. As a default the algorithm will use symbolic differentiation to automatically find the derivative from estimand_fun |

| | |
|------------------|---|
| inv_estimand_fun | (optional) a function with arguments psi0 and target_effect, so estimand_fun(psi1 = y, psi0 = x) = z and inv_estimand_fun(psi0 = x, target_effect = z) = y for all x, y, z. If left as NULL, the inverse will be found numerically. |
| margin | a numeric superiority margin. As a default, the estimand_fun is evaluated with the same counterfactual means psi1 and psi0, corresponding to a superiority margin assuming no difference (fx. 0 for ATE and 1 for rate ratio). |
| alpha | a numeric significance level. The critical value used for testing corresponds to using the significance level for a two-sided test. I.e. we use the quantile $1 - \alpha/2$ of the null distribution as the critical value. |
| tolerance | passed to all.equal when comparing calculated target_effect from derivations and given target_effect during numeric derivation of inv_estimand_fun. Thus only relevant when inv_estimand_fun is NULL. |
| verbose | numeric verbosity level. Higher values means more information is printed in console. A value of 0 means nothing is printed to console during execution (Defaults to 2, overwritable using option 'postcard.verbose' or environment variable 'R_POSTCARD_VERBOSE') |
| ... | arguments passed to <code>stats::uniroot</code> , which is used to find the inverse of estimand_fun, when inv_estimand_fun is NULL. |

Details

The reference in the description shows in its "Prospective power" section a derivation of a variance bound

$$v_{\infty}^{\uparrow 2} = r_0'^2 \sigma_0^2 + r_1'^2 \sigma_1^2 + \pi_0 \pi_1 \left(\frac{|r_0'| \kappa_0}{\pi_0} + \frac{|r_1'| \kappa_1}{\pi_1} \right)^2$$

where r_a' is the derivative of the estimand_fun with respect to Ψ_a , σ_a^2 is the variance of the potential outcome corresponding to group a , π_a is the probability of being assigned to group a , and κ_a is the expected mean-squared error when predicting the potential outcome corresponding to group a .

The variance bound is then used for calculating a lower bound of the power using the distributions corresponding to the null and alternative hypotheses $\mathcal{H}_0 : \hat{\Psi} \sim F_0 = \mathcal{N}(\Delta, v_{\infty}^{\uparrow 2}/n)$ and $\mathcal{H}_1 : \hat{\Psi} \sim F_1 = \mathcal{N}(\Psi, v_{\infty}^{\uparrow 2}/n)$. See more details in the reference.

Relating arguments to formulas:

- response: Used to obtain both σ_0^2 (by taking the sample variance of the response) and κ_0 .
- predictions: Used when calculating the MSE κ_0 .
- var1: σ_1^2 . As a default, chosen to be the same as σ_0^2 . Can specify differently through this argument fx. by
 - Inflating or deflating the value of σ_0^2 by a scalar according to prior beliefs. Fx. specify `var1 = function(x) 1.2 * x` to inflate σ_0^2 by 1.2.
 - If historical data is available for group 1, an estimate of the variance from that data can be provided here.
- kappa1_squared: κ_1 . Same as for var1, default is to assume the same value as kappa0_squared, which is found by using the response and predictions vectors. Adjust the value according to prior beliefs if relevant.
- target_effect: Ψ .
- exposure_prob: π_1

Value

a numeric with the estimated power.

See Also

See [power_linear](#) for power approximation functionalities specific to linear models.

Examples

```
# Generate a data set to use as an example
n <- 100
exposure_prob <- .5

dat_gaus <- glm_data(Y ~ 1+2*X1-X2+3*A+1.6*A*X2,
  X1 = rnorm(n),
  X2 = rgamma(n, shape = 2),
  A = rbinom(n, size = 1, prob = exposure_prob),
  family = gaussian())

# -----
# Obtain out-of-sample (OOS) prediction using glm model
# -----
gaus1 <- dat_gaus[1:(n/2), ]
gaus2 <- dat_gaus[(n/2+1):n, ]

glm1 <- glm(Y ~ X1 + X2 + A, data = gaus1)
glm2 <- glm(Y ~ X1 + X2 + A, data = gaus2)
preds_glm1 <- predict(glm2, newdata = gaus1, type = "response")
preds_glm2 <- predict(glm1, newdata = gaus2, type = "response")
preds_glm <- c(preds_glm1, preds_glm2)

# Obtain power
power_marginaleffect(
  response = dat_gaus$Y,
  predictions = preds_glm,
  target_effect = 2,
  exposure_prob = exposure_prob
)

# -----
# Get OOS predictions using discrete super learner and adjust variance
# -----
learners <- list(
  mars = list(
    model = parsnip::set_engine(
      parsnip::mars(
        mode = "regression", prod_degree = 3
      ),
      "earth"
    )
  ),
  lm = list(
```

```

      model = parsnip::set_engine(
        parsnip::linear_reg(),
        "lm"
      )
    )
  )
  lrnr1 <- fit_best_learner(preproc = list(mod = Y ~ X1 + X2 + A),
    data = gaus1,
    learners = learners)
  lrnr2 <- fit_best_learner(preproc = list(mod = Y ~ X1 + X2 + A),
    data = gaus2,
    learners = learners)
  preds_lrnr1 <- dplyr::pull(predict(lrnr2, new_data = gaus1))
  preds_lrnr2 <- dplyr::pull(predict(lrnr1, new_data = gaus2))
  preds_lrnr <- c(preds_lrnr1, preds_lrnr2)

  # Estimate the power AND adjust the assumed variance in the "unknown"
  # group 1 to be 20% larger than in group 0
  power_marginaleffect(
    response = dat_gaus$Y,
    predictions = preds_lrnr,
    target_effect = 2,
    exposure_prob = exposure_prob,
    var1 = function(var0) 1.2 * var0
  )

```

 prog

Extract information about the fitted prognostic model

Description

Extracts the `prognostic_info` list element from an `rctglm_prog` object. See 'Value' at [rctglm_with_prognosticscore](#) for more details.

Usage

```

prog(x)

## S3 method for class 'rctglm_prog'
prog(x)

```

Arguments

`x` an object of class `rctglm_prog` (returned by [rctglm_with_prognosticscore](#))

Value

a list with the structure described of `prognostic_info` in the Value section of [rctglm_with_prognosticscore](#).

See Also

The generic `rctglm_with_prognosticscore()` for which this method works.

Examples

```
# Generate some data
n <- 100
b0 <- 1
b1 <- 1.5
b2 <- 2
W1 <- runif(n, min = 1, max = 10)
exposure_prob <- .5

dat_treat <- glm_data(
  Y ~ b0+b1*log(W1)+b2*A,
  W1 = W1,
  A = rbinom(n, 1, exposure_prob)
)

dat_notreat <- glm_data(
  Y ~ b0+b1*log(W1),
  W1 = W1
)

learners <- list(
  mars = list(
    model = parsnip::set_engine(
      parsnip::mars(
        mode = "regression", prod_degree = 3
      ),
      "earth"
    )
  )
)
ate <- rctglm_with_prognosticscore(
  formula = Y ~ .,
  exposure_indicator = A,
  exposure_prob = exposure_prob,
  data = dat_treat,
  family = gaussian(),
  estimand_fun = "ate",
  data_hist = dat_notreat,
  learners = learners)

prog(ate)
```

Description

The procedure uses plug-in-estimation and influence functions to perform robust inference of any specified estimand in the setting of a randomised clinical trial, even in the case of heterogeneous effect of covariates in randomisation groups. See [Powering RCTs for marginal effects with GLMs using prognostic score adjustment](#) by Højbjerg-Frandsen et. al (2025) for more details on methodology.

Usage

```
rctglm(
  formula,
  exposure_indicator,
  exposure_prob,
  data,
  family = gaussian,
  estimand_fun = "ate",
  estimand_fun_deriv0 = NULL,
  estimand_fun_deriv1 = NULL,
  cv_variance = FALSE,
  cv_variance_folds = 10,
  verbose = options::opt("verbose"),
  ...
)
```

Arguments

| | |
|--------------------|---|
| formula | an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details' in the glm documentation. |
| exposure_indicator | (name of) the <i>binary</i> variable in data that identifies randomisation groups. The variable is required to be binary currently. |
| exposure_prob | a numeric with the probability of being in "group 1" (rather than group 0) in groups defined by exposure_indicator. |
| data | an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which the function is called. |
| family | a description of the error distribution and link function to be used in the model. For <code>glm</code> this can be a character string naming a family function, a family function or the result of a call to a family function. For <code>glm.fit</code> only the third option is supported. (See family for details of family functions.) |
| estimand_fun | a function with arguments <code>psi1</code> and <code>psi0</code> specifying the estimand. Alternative, specify "ate" or "rate_ratio" as a character to use one of the default estimand functions. See more details in the "Estimand" section of rctglm . |

| | |
|----------------------------------|--|
| <code>estimand_fun_deriv0</code> | a function specifying the derivative of <code>estimand_fun</code> wrt. <code>psi0</code> . As a default the algorithm will use symbolic differentiation to automatically find the derivative from <code>estimand_fun</code> |
| <code>estimand_fun_deriv1</code> | a function specifying the derivative of <code>estimand_fun</code> wrt. <code>psi1</code> . As a default the algorithm will use symbolic differentiation to automatically find the derivative from <code>estimand_fun</code> |
| <code>cv_variance</code> | a logical determining whether to estimate the variance using cross-validation (see details of rctglm). |
| <code>cv_variance_folds</code> | a numeric with the number of folds to use for cross validation if <code>cv_variance</code> is TRUE. |
| <code>verbose</code> | numeric verbosity level. Higher values means more information is printed in console. A value of 0 means nothing is printed to console during execution (Defaults to 2, overwritable using option <code>'postcard.verbose'</code> or environment variable <code>'R_POSTCARD_VERBOSE'</code>) |
| <code>...</code> | Additional arguments passed to <code>stats::glm()</code> |

Details

The procedure assumes the setup of a randomised clinical trial with observations grouped by a binary `exposure_indicator` variable, allocated randomly with probability `exposure_prob`. A GLM is fit and then used to predict the response of all observations in the event that the `exposure_indicator` is 0 and 1, respectively. Taking means of these predictions produce the *counterfactual means* `psi0` and `psi1`, and an estimand $r(\psi_0, \psi_1)$ is calculated using any specified `estimand_fun`.

The variance of the estimand is found by taking the variance of the influence function of the estimand. If `cv_variance` is TRUE, then the counterfactual predictions for each observation (which are used to calculate the value of the influence function) is obtained as out-of-sample (OOS) predictions using cross validation with number of folds specified by `cv_variance_folds`. The cross validation splits are performed using stratified sampling with `exposure_indicator` as the `strata` argument in [`rsample::vfold_cv`](#).

Read more in `vignette("model-fit")`.

Value

`rctglm` returns an object of class inheriting from `"rctglm"`.

An object of class `rctglm` is a list containing the following components:

- `estimand`: A `data.frame` with plug-in estimate of estimand, standard error (SE) estimate and variance estimate of estimand
- `estimand_funs`: A list with
 - `f`: The `estimand_fun` used to obtain an estimate of the estimand from counterfactual means
 - `d0`: The derivative with respect to `psi0`
 - `d1`: The derivative with respect to `psi1`

- `means_counterfactual`: A `data.frame` with counterfactual means `psi0` and `psi1`
- `fitted.values_counterfactual`: A `data.frame` with counterfactual mean values, obtained by transforming the linear predictors for each group by the inverse of the link function.
- `glm`: A `glm` object returned from running `stats::glm` within the procedure
- `call`: The matched call

Estimands

As noted in the description, `psi0` and `psi1` are the counterfactual means found by prediction using a fitted GLM in the binary groups defined by `exposure_indicator`.

Default estimand functions can be specified via `"ate"` (which uses the function `function(psi1, psi0) psi1-psi0`) and `"rate_ratio"` (which uses the function `function(psi1, psi0) psi1/psi0`). See more information on specifying the `estimand_fun` in `vignette("model-fit")`.

As a default, the `Deriv` package is used to perform symbolic differentiation to find the derivatives of the `estimand_fun`.

See Also

See how to extract information using methods in [rctglm_methods](#).

Use `rctglm_with_prognosticscore()` to include prognostic covariate adjustment.

See vignettes

Examples

```
# Generate some data to showcase example
n <- 100
exp_prob <- .5

dat_gaus <- glm_data(
  Y ~ 1+1.5*X1+2*A,
  X1 = rnorm(n),
  A = rbinom(n, 1, exp_prob),
  family = gaussian()
)

# Fit the model
ate <- rctglm(formula = Y ~ .,
              exposure_indicator = A,
              exposure_prob = exp_prob,
              data = dat_gaus,
              family = gaussian)

# Pull information on estimand
estimand(ate)

## Another example with different family and specification of estimand_fun
dat_binom <- glm_data(
  Y ~ 1+1.5*X1+2*A,
  X1 = rnorm(n),
```

```

  A = rbinom(n, 1, exp_prob),
  family = binomial()
)

rr <- rctglm(formula = Y ~ .,
             exposure_indicator = A,
             exposure_prob = exp_prob,
             data = dat_binom,
             family = binomial(),
             estimand_fun = "rate_ratio")

odds_ratio <- function(psi1, psi0) (psi1*(1-psi0))/(psi0*(1-psi1))
or <- rctglm(formula = Y ~ .,
             exposure_indicator = A,
             exposure_prob = exp_prob,
             data = dat_binom,
             family = binomial,
             estimand_fun = odds_ratio)

```

rctglm_methods

Methods for objects of class rctglm

Description

Methods mostly to extract information from model fit and inference. See details for more information on each method.

Usage

```

estimand(object)

## S3 method for class 'rctglm'
estimand(object)

est(object)

## S3 method for class 'rctglm'
coef(object, ...)

## S3 method for class 'rctglm'
predict(object, ...)

## S3 method for class 'rctglm'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

```

Arguments

| | |
|--------|--|
| object | an object of class rctglm |
| ... | additional arguments passed to methods |
| x | an object of class rctglm |
| digits | a numeric with the number of digits to display when printing |

Details

The function `estimand` (or short-hand version `est`) can be used to extract a `data.frame` with an estimated value and standard error of the estimand.

A method for the generic `coef()` has been added for class `rctglm`, which extracts coefficient information from the underlying `glm` fit in the procedure. The same is true for the method defined for the `predict()` generic. The method for an `rctglm` class object calls `predict.glm()` on the `glm` fit contained within an `rctglm` object.

Value

`estimand/est` returns a `data.frame` with columns `Estimate` and `Std. Error` with the estimate and standard error of the estimand.

See `coef()` and `predict.glm()` for details on what is returned by the corresponding methods.

See Also

The generic `rctglm()` which produces `rctglm` class objects.

Examples

```
# Generate some data to showcase example
n <- 100
exposure_prob <- .5
dat_binom <- glm_data(
  Y ~ 1+1.5*X1+2*A,
  X1 = rnorm(n),
  A = rbinom(n, 1, exposure_prob),
  family = binomial()
)

# Fit the model
ate <- rctglm(formula = Y ~ .,
              exposure_indicator = A,
              exposure_prob = exposure_prob,
              data = dat_binom,
              family = binomial,
              estimand_fun = "ate")

print(ate)
estimand(ate)
coef(ate)
```

 rctglm_with_prognosticscore

Use prognostic covariate adjustment when fitting an [rctglm](#)

Description

The procedure uses [fit_best_learner](#) to fit a prognostic model to historical data and uses the model to produce counterfactual predictions as a prognostic score that is then adjusted for as a covariate in the [rctglm](#) procedure. See [Powering RCTs for marginal effects with GLMs using prognostic score adjustment](#) by Højbjerg-Frandsen et. al (2025) for more details.

Usage

```
rctglm_with_prognosticscore(
  formula,
  exposure_indicator,
  exposure_prob,
  data,
  family = gaussian,
  estimand_fun = "ate",
  estimand_fun_deriv0 = NULL,
  estimand_fun_deriv1 = NULL,
  cv_variance = FALSE,
  cv_variance_folds = 10,
  ...,
  data_hist,
  prog_formula = NULL,
  cv_prog_folds = 5,
  learners = default_learners(),
  verbose = options::opt("verbose")
)
```

Arguments

| | |
|--------------------|--|
| formula | an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details' in the glm documentation. |
| exposure_indicator | (name of) the <i>binary</i> variable in data that identifies randomisation groups. The variable is required to be binary currently. |
| exposure_prob | a numeric with the probability of being in "group 1" (rather than group 0) in groups defined by exposure_indicator. |
| data | an optional data frame, list or environment (or object coercible by <code>as.data.frame</code> to a data frame) containing the variables in the model. If not found in data, the variables are taken from <code>environment(formula)</code> , typically the environment from which the function is called. |

| | |
|---------------------|--|
| family | a description of the error distribution and link function to be used in the model. For <code>glm</code> this can be a character string naming a family function, a family function or the result of a call to a family function. For <code>glm.fit</code> only the third option is supported. (See family for details of family functions.) |
| estimand_fun | a function with arguments <code>psi1</code> and <code>psi0</code> specifying the estimand. Alternative, specify "ate" or "rate_ratio" as a character to use one of the default estimand functions. See more details in the "Estimand" section of rctglm . |
| estimand_fun_deriv0 | a function specifying the derivative of <code>estimand_fun</code> wrt. <code>psi0</code> . As a default the algorithm will use symbolic differentiation to automatically find the derivative from <code>estimand_fun</code> |
| estimand_fun_deriv1 | a function specifying the derivative of <code>estimand_fun</code> wrt. <code>psi1</code> . As a default the algorithm will use symbolic differentiation to automatically find the derivative from <code>estimand_fun</code> |
| cv_variance | a logical determining whether to estimate the variance using cross-validation (see details of rctglm). |
| cv_variance_folds | a numeric with the number of folds to use for cross validation if <code>cv_variance</code> is TRUE. |
| ... | Additional arguments passed to <code>stats::glm()</code> |
| data_hist | a <code>data.frame</code> with historical data on which to fit a prognostic model |
| prog_formula | an object of class "formula" (or one that can be coerced to that class): a symbolic description of the prognostic model to be fitted to <code>data_hist</code> . Passed in a list as the <code>preproc</code> argument in <code>fit_best_learner()</code> . As a default, the formula is created by modelling the response (assumed to have the same name as in <code>formula</code>) using all columns in <code>data_hist</code> . |
| cv_prog_folds | a numeric with the number of cross-validation folds used when fitting and evaluating models |
| learners | a list (preferably named) containing named lists of elements <code>model</code> and optionally <code>grid</code> . The <code>model</code> element should be a <code>parsnip</code> model specification, which is passed to <code>workflowsets::workflow_set</code> as the <code>model</code> argument, while the <code>grid</code> element is passed as the <code>grid</code> argument of <code>workflowsets::option_add</code> |
| verbose | numeric verbosity level. Higher values means more information is printed in console. A value of 0 means nothing is printed to console during execution (Defaults to 2, overwritable using option 'postcard.verbose' or environment variable 'R_POSTCARD_VERBOSE') |

Details

Prognostic covariate adjustment involves training a prognostic model (using `fit_best_learner`) on historical data (`data_hist`) to predict the response in that data.

Assuming that the historical data is representative of the comparator group in a "new" data set (group 0 of the binary `exposure_indicator` in `data`), we can use the prognostic model to predict the counterfactual outcome of all observations (including the ones in the comparator group for which the prediction of counterfactual outcome coincides with a prediction of actual outcome).

This prediction, which is called the prognostic score, is then used as an adjustment covariate in the GLM (the prognostic score is added to formula before calling `rctglm` with the modified formula).

See much more details in the reference in the description.

Value

`rctglm_with_prognosticscore` returns an object of class `rctglm_prog`, which inherits from `rctglm`.

An `rctglm_prog` object is a list with the same components as an `rctglm` object (see the Value section of `rctglm` for a breakdown of the structure), but with an additional list element of:

- `prognostic_info`: List with information about the fitted prognostic model on historical data. It has components:
 - `formula`: The formula with symbolic description of how the response is modelled as function of covariates in the models
 - `model_fit`: A trained workflow - the result of `fit_best_learner`
 - `learners`: A list of learners used for the discrete super learner
 - `cv_folds`: The amount of folds used for cross validation
 - `data`: The historical data used for cross validation when fitting and testing models

See Also

Method to extract information of the prognostic model in `prog`. Function used to fit the prognostic model is `fit_best_learner()`.

See `rctglm()` for the function and class this inherits from.

Examples

```
# Generate some data
n <- 100
b0 <- 1
b1 <- 1.5
b2 <- 2
W1 <- runif(n, min = 1, max = 10)
exp_prob <- .5

dat_treat <- glm_data(
  Y ~ b0+b1*log(W1)+b2*A,
  W1 = W1,
  A = rbinom(n, 1, exp_prob)
)

dat_notreat <- glm_data(
  Y ~ b0+b1*log(W1),
  W1 = W1
)

learners <- list(
  mars = list(
```

```

    model = parsnip::set_engine(
      parsnip::mars(
        mode = "regression", prod_degree = 3
      ),
      "earth"
    )
  ),
  lm = list(
    model = parsnip::set_engine(
      parsnip::linear_reg(),
      "lm"
    )
  )
)
ate <- rctglm_with_prognosticscore(
  formula = Y ~ .,
  exposure_indicator = A,
  exposure_prob = exp_prob,
  data = dat_treat,
  family = gaussian(),
  estimand_fun = "ate",
  data_hist = dat_notreat,
  learners = learners)

# Pull information on estimand
estimand(ate)

```

repeat_power_linear *Create data and plot power curves calculated using functions in `power_linear()` for a list of formulas/models*

Description

Estimate a variance for power approximation using `variance_ancova()` for each formula in `formula_list` on `train_data`. Then calculate power using the function with name specified in `power_fun` across a number of sample sizes `ns` for an assumed average treatment effect of `ate`.

Usage

```

repeat_power_linear(
  ate,
  formula_list,
  train_data,
  power_fun = c("power_gs", "power_nc"),
  ns = 10:400,
  desired_power = 0.9,
  ...
)

```

```
## S3 method for class 'postcard_rpl'
plot(x, cols = NULL, ...)
```

Arguments

| | |
|---------------|---|
| ate | Passed to power_gs() or power_nc() |
| formula_list | a named list of formulas that are element wise passed to variance_ancova() |
| train_data | Passed as the data argument in variance_ancova() |
| power_fun | a character string with value "power_gs" or "power_nc", specifying what function in the power_linear() topic to use |
| ns | a numeric vector of sample sizes |
| desired_power | a numeric between 0 and 1 indicating the desired power level |
| ... | Arguments passed to variance_ancova() and power_gs() or power_nc() |
| x | an object of class <code>postcard_rpl</code> created by <code>repeat_power_linear()</code> |
| cols | a (potentially named) character vector of colors for the different models in <code>formula_list</code> |

Value

`repeat_power_linear` returns an object of class `postcard_rpl`, which is just a `data.frame` with a `plot` method defined. The `plot` method returns a `ggplot2` object.

See Also

[repeat_power_marginaleffect\(\)](#) for a similar implementation to iterate the process of approximating power with the [power_marginaleffect\(\)](#)

Examples

```
train_data <- glm_data(
  Y ~ 1+1.5*log(W)+2*X,
  W = runif(1e3, min = 1, max = 10),
  X = rnorm(1e3, sd = 3)
)
rpl <- repeat_power_linear(
  ate = 0.5,
  formula_list = list("ANCOVA 1 covariate" = Y ~ X, "ANCOVA 2 covariates" = Y ~ W + X),
  train_data = train_data)

rpl_nc <- repeat_power_linear(
  ate = 0.5,
  formula_list = list("ANCOVA 1 covariate" = Y ~ X, "ANCOVA 2 covariates" = Y ~ W + X),
  train_data = train_data,
  power_fun = "power_nc",
  df = 1e3-3,
  deflation = 0.95,
  margin = -0.2,
```

```

    r = 2)

## Not run:
plot(rpl)

plot(rpl_nc)

## End(Not run)

```

```
repeat_power_marginaleffect
```

Create data and plot power curves calculated using `power_marginaleffect()` for a list of models

Description

Iterate a process of simulating test data from `test_data_fun`, making predictions using models in `model_list`, and calculating power using `power_marginaleffect()` across a number of sample sizes `ns` and iterations `n_iter`. The results are averaged and used to create a plot of the resulting power curves.

Usage

```

repeat_power_marginaleffect(
  target_effect,
  exposure_prob,
  model_list = default_power_model_list(),
  test_data_fun = function(n) {
    glm_data(Y ~ 1 + 3 * log(W), W = stats::runif(n, min
      = 1, max = 50))
  },
  ns = seq(10, 300, 10),
  desired_power = 0.9,
  n_iter = 1,
  ...
)

## S3 method for class 'postcard_rpm'
plot(x, cols = NULL, ...)

```

Arguments

| | |
|----------------------------|--|
| <code>target_effect</code> | Passed to <code>power_marginaleffect()</code> |
| <code>exposure_prob</code> | Passed to <code>power_marginaleffect()</code> |
| <code>model_list</code> | a named list of models used to get predictions on generated test data sets that are then passed to <code>power_marginaleffect()</code> as predictions. The elements of <code>model_list</code> need to have an existing <code>predict()</code> method. The default is an |

ANCOVA and a prognostic model fitted with `fit_best_learner()` to a simple data set of 1000 observations generated with a non-linear effect of a single covariate using `glm_data()`.

`test_data_fun` a function with a single argument `n` that generates test data sets for the sample sizes `ns` specified. The default generates data using `glm_data()` with the same data generating process as the training data used to fit the default models in `model_list`.

`ns` a numeric vector of sample sizes

`desired_power` a numeric between 0 and 1 indicating the desired power level

`n_iter` a numeric indicating a number of iterations to process and average over

`...` additional arguments passed to `power_marginaleffect()`

`x` an object of class `postcard_rpm` created by `repeat_power_marginaleffect()`

`cols` a (potentially named) character vector of colors for the different models in `model_list`

Value

`repeat_power_marginal` returns an object of class `postcard_rpm`, which is just a `data.frame` with a `plot` method defined. The `plot` method returns a `ggplot2` object.

See Also

`repeat_power_linear()` for a similar implementation to iterate the process of approximating power with the functions in `power_linear()`

Examples

```
# Note everything is wrapped in dontrun to avoid long runtimes of examples (tests are
# still in place). Reduce the number of sample sizes and/or iterations to avoid long
# runtimes
## Not run:
# A simple use case with default models and test data (we run only with a few sample
# sizes to reduce runtime of examples)
rpm <- repeat_power_marginaleffect(
  target_effect = 0.9,
  exposure_prob = 0.5
)
plot(rpm)

#####
# Create model from a poisson family and estimate the power of rate ratio with
# several arguments passed to power_marginaleffect
#####
b1 <- 0.9
b2 <- 0.2
b3 <- -0.4
b4 <- -0.6

train_pois <- glm_data(
```

```

Y ~ b1*log(X1)+b2*X2+b3*X3+b4*X2*X3,
X1 = runif(1e3, min = 1, max = 10),
X2 = rnorm(1e3),
X3 = rgamma(1e3, shape = 1),
family = poisson()
)

# Define models to compare fit to training data
ancova_prog_list <- list(
ANCOVA = glm(Y ~ X1 + X2 + X3, data = train_pois, family = poisson),
"ANCOVA with prognostic score" = fit_best_learner(list(mod = Y ~ X1 + X2 + X3), data = train_pois)
)

# Create a function that produces data to predict on
test_pois_fun <- function(n) {
  glm_data(
    Y ~ b1*log(X1)+b2*X2+b3*X3+b4*X2*X3,
    X1 = runif(n, min = 1, max = 10),
    X2 = rnorm(n),
    X3 = rgamma(n, shape = 1),
    family = poisson()
  )
}

# Specify a bunch of different arguments that are passed to power_marginaleffect()
## Run for 2 sample sizes to reduce runtime
rpm_rr <- repeat_power_marginaleffect(
  model_list = ancova_prog_list,
  test_data_fun = test_pois_fun,
  ns = seq(100, 200), n_iter = 1,
  var1 = function(var0) 1.1 * var0,
  kappa1_squared = function(kap0) 1.1 * kap0,
  estimand_fun = "rate_ratio",
  target_effect = 1.4,
  exposure_prob = 1/2,
  margin = 0.8
)
plot(rpm_rr2)

## End(Not run)

```

Index

`all.equal`, 12
`as.data.frame`, 7

`coef()`, 20
`coef.rctglm(rctglm_methods)`, 19

`default_learners`, 2

`est`, 20
`est(rctglm_methods)`, 19
`estimand`, 20
`estimand(rctglm_methods)`, 19

`family`, 16, 22
`fit_best_learner`, 2, 3, 21–23
`fit_best_learner()`, 22, 23, 27

`glm`, 5, 16, 21
`glm_data`, 4
`glm_data()`, 27

`options`, 6

`plot.postcard_rpl`
 (`repeat_power_linear`), 24
`plot.postcard_rpm`
 (`repeat_power_margineffect`),
 26

`power_gs(power_linear)`, 7
`power_gs()`, 25
`power_linear`, 7, 13
`power_linear()`, 24, 25, 27
`power_margineffect`, 10, 10
`power_margineffect()`, 25–27
`power_nc(power_linear)`, 7
`power_nc()`, 25
`predict()`, 20
`predict.glm()`, 20
`predict.rctglm(rctglm_methods)`, 19
`print.rctglm(rctglm_methods)`, 19
`prog`, 14, 23

`rctglm`, 11, 15, 16, 17, 21–23
`rctglm()`, 20, 23
`rctglm_methods`, 18, 19
`rctglm_with_prognosticscore`, 14, 21
`rctglm_with_prognosticscore()`, 3, 15, 18
`repeat_power_linear`, 24
`repeat_power_linear()`, 7, 27
`repeat_power_margineffect`, 26
`repeat_power_margineffect()`, 25
`rsample::vfold_cv`, 17

`samplesize_gs(power_linear)`, 7
`stats::glm`, 18
`stats::glm()`, 17, 22
`stats::model.frame()`, 7
`stats::uniroot`, 12

`variance_ancova(power_linear)`, 7
`variance_ancova()`, 24, 25

`workflows::workflow_variables()`, 3
`workflowsets::option_add`, 3, 22
`workflowsets::workflow_set`, 3, 22
`workflowsets::workflow_set()`, 3