Package: CausalGPS (via r-universe)

October 26, 2026

Type Package

Title Matching on Generalized Propensity Scores with Continuous Exposures

Version 0.5.1

Maintainer Naeem Khoshnevis <nkhoshnevis@g.harvard.edu>

Description Provides a framework for estimating causal effects of a continuous exposure using observational data, and implementing matching and weighting on the generalized propensity score. Wu, X., Mealli, F., Kioumourtzoglou, M.A., Dominici, F. and Braun, D., 2022. Matching on generalized propensity scores with continuous exposures. Journal of the American Statistical Association, pp.1-29.

License GPL-3

Language en-US

URL https://github.com/NSAPH-Software/CausalGPS

BugReports https://github.com/NSAPH-Software/CausalGPS/issues

Copyright Harvard University

Imports parallel, data.table, SuperLearner, xgboost, gam, MASS, polycor, wCorr, stats, ggplot2, rlang, logger, Rcpp, gnm, locpol, Ecume, KernSmooth, cowplot

Encoding UTF-8

RoxygenNote 7.3.3

Suggests covr, knitr, rmarkdown, ranger, earth, testthat, gridExtra

VignetteBuilder knitr

Depends R (>= 3.5.0)

LinkingTo Rcpp

NeedsCompilation yes

Author Naeem Khoshnevis [aut, cre] (ORCID: <https://orcid.org/0000-0003-4315-1426>, AFFILIATION: Kempner), Xiao Wu [aut] (ORCID: <https://orcid.org/0000-0002-4884-657X>,

AFFILIATION: CUMC), Danielle Braun [aut] (ORCID:
<https://orcid.org/0000-0002-5177-8598>, AFFILIATION: HSPH)

Repository https://cran.r-universe.dev

Date/Publication 2026-01-11 20:39:26 UTC

RemoteUrl https://github.com/cran/CausalGPS

RemoteRef HEAD

RemoteSha 3a837d66e5cb791a310a18b63919abe04b4b290f

Contents

CausalGPS-package . . . 3
absolute_corr_fun . . . 3
absolute_weighted_corr_fun . . . 4
check_covar_balance . . . 5
compile_pseudo_pop . . . 7
compute_counter_weight . . . 9
estimate_erf . . . 10
estimate_gps . . . 11
generate_pseudo_pop . . . 12
generate_syn_data . . . 14
get_logger . . . 15
plot.cgps_cw . . . 16
plot.cgps_erf . . . 17
plot.cgps_gps . . . 17
plot.cgps_pspop . . . 18
print.cgps_cw . . . 18
print.cgps_erf . . . 19
print.cgps_gps . . . 19
print.cgps_pspop . . . 20
set_logger . . . 20
summary.cgps_cw . . . 21
summary.cgps_erf . . . 21
summary.cgps_gps . . . 22
summary.cgps_pspop . . . 22
synthetic_us_2010 . . . 23
trim_it . . . 25

Index **27**

CausalGPS-package *The 'CausalGPS' package.*

Description

An R package for implementing matching and weighting on generalized propensity scores with continuous exposures.

Details

We developed an innovative approach for estimating causal effects using observational data in settings with continuous exposures, and introduce a new framework for GPS caliper matching.

Author(s)

Naeem Khoshnevis

Xiao Wu

Danielle Braun

References

Wu, X., Mealli, F., Kioumourtzoglou, M.A., Dominici, F. and Braun, D., 2022. Matching on generalized propensity scores with continuous exposures. Journal of the American Statistical Association, pp.1-29.

Kennedy, E.H., Ma, Z., McHugh, M.D. and Small, D.S., 2017. Non-parametric methods for doubly robust estimation of continuous treatment effects. Journal of the Royal Statistical Society. Series B (Statistical Methodology), 79(4), pp.1229-1245.

See Also

Useful links:

- https://github.com/NSAPH-Software/CausalGPS
- Report bugs at https://github.com/NSAPH-Software/CausalGPS/issues

absolute_corr_fun *Check covariate balance using absolute approach*

Description

Checks covariate balance based on absolute correlations for given data sets.

Usage

```
absolute_corr_fun(w, c)
```

Arguments

w	A vector of observed continuous exposure variable.
c	A data.frame of observed covariates variable.

Value

The function returns a list including:

- absolute_corr: the absolute correlations for each pre-exposure covariates;
- mean_absolute_corr: the average absolute correlations for all pre-exposure covariates.

Examples

```
set.seed(291)
n <- 100
mydata <- generate_syn_data(sample_size=100)
year <- sample(x=c("2001","2002","2003","2004","2005"),size = n,
 replace = TRUE)
region <- sample(x=c("North", "South", "East", "West"),size = n,
 replace = TRUE)
mydata$year <- as.factor(year)
mydata$region <- as.factor(region)
mydata$cf5 <- as.factor(mydata$cf5)
cor_val <- absolute_corr_fun(mydata[,2], mydata[, 3:length(mydata)])
print(cor_val$mean_absolute_corr)
```

---

absolute_weighted_corr_fun

Check Weighted Covariate Balance Using Absolute Approach

Description

Checks covariate balance based on absolute weighted correlations for given data sets.

Usage

```
absolute_weighted_corr_fun(w, vw, c)
```

Arguments

w	A vector of observed continuous exposure variable.
vw	A vector of weights.
c	A data.table of observed covariates variable.

Value

The function returns a list saved the measure related to covariate balance `absolute_corr`: the absolute correlations for each pre-exposure covairates; `mean_absolute_corr`: the average absolute correlations for all pre-exposure covairates.

Examples

```
set.seed(639)
n <- 100
mydata <- generate_syn_data(sample_size=100)
year <- sample(x=c("2001","2002","2003","2004","2005"),size = n,
               replace = TRUE)
region <- sample(x=c("North", "South", "East", "West"),size = n,
                 replace = TRUE)
mydata$year <- as.factor(year)
mydata$region <- as.factor(region)
mydata$cf5 <- as.factor(mydata$cf5)
cor_val <- absolute_weighted_corr_fun(mydata[,2],
                                      runif(n),
                                      mydata[, 3:length(mydata)])
print(cor_val$mean_absolute_corr)
```

---

check_covar_balance *Check covariate balance*

Description

Checks the covariate balance of original population or pseudo population.

Usage

```
check_covar_balance(
  w,
  c,
  ci_appr,
  counter_weight = NULL,
  covar_bl_method = "absolute",
  covar_bl_trs = 0.1,
  covar_bl_trs_type = "mean"
)
```

Arguments

`w`	A vector of observed continuous exposure variable.
`c`	A data.frame of observed covariates variable.
`ci_appr`	The causal inference approach.

counter_weight A weight vector in different situations. If the matching approach is selected, it is an integer data.table of counters. In the case of the weighting approach, it is weight data.table.

covar_bl_method
Covariate balance method. Available options: - 'absolute'

covar_bl_trs Covariate balance threshold.

covar_bl_trs_type
Covariate balance type (mean, median, maximal).

Value

output object:

- corr_results
  - absolute_corr
  - mean_absolute_corr
- pass (TRUE,FALSE)

Examples

```
set.seed(422)
n <- 100
mydata <- generate_syn_data(sample_size=n)
year <- sample(x=c("2001","2002","2003","2004","2005"),size = n,
               replace = TRUE)
region <- sample(x=c("North", "South", "East", "West"),size = n,
                 replace = TRUE)
mydata$year <- as.factor(year)
mydata$region <- as.factor(region)
mydata$cf5 <- as.factor(mydata$cf5)

m_xgboost <- function(nthread = 1,
                      ntrees = 35,
                      shrinkage = 0.3,
                      max_depth = 5,
                      ...) {SuperLearner::SL.xgboost(
                        nthread = nthread,
                        ntrees = ntrees,
                        shrinkage=shrinkage,
                        max_depth=max_depth,
                        ...)}

data_with_gps <- estimate_gps(.data = mydata,
                              .formula = w ~ cf1 + cf2 + cf3 + cf4 + cf5 +
                                             cf6 + year + region,
                              sl_lib = c("m_xgboost"),
                              gps_density = "kernel")


cw_object_matching <- compute_counter_weight(gps_obj = data_with_gps,
```

```
                                          ci_appr = "matching",
                                          bin_seq = NULL,
                                          nthread = 1,
                                          delta_n = 0.1,
                                          dist_measure = "l1",
                                          scale = 0.5)

pseudo_pop <- generate_pseudo_pop(.data = mydata,
                                  cw_obj = cw_object_matching,
                                  covariate_col_names = c("cf1", "cf2", "cf3",
                                                          "cf4", "cf5", "cf6",
                                                          "year", "region"),
                                  covar_bl_trs = 0.1,
                                  covar_bl_trs_type = "maximal",
                                  covar_bl_method = "absolute")


adjusted_corr_obj <- check_covar_balance(w = pseudo_pop$.data[, c("w")],
                                         c = pseudo_pop$.data[ ,
                                         pseudo_pop$params$covariate_col_names],
                                         counter = pseudo_pop$.data[,
                                                    c("counter_weight")],
                                         ci_appr = "matching",
                                         covar_bl_method = "absolute",
                                         covar_bl_trs = 0.1,
                                         covar_bl_trs_type = "mean")
```

`compile_pseudo_pop` *Compile pseudo population*

Description

Compiles pseudo population based on the original population and estimated GPS value.

Usage

```
compile_pseudo_pop(
  data_obj,
  ci_appr,
  gps_density,
  exposure_col_name,
  nthread,
  ...
)
```

Arguments

`data_obj`	A S3 object including the following:

- Original data set + GPS values
- e_gps_pred
- e_gps_std_pred
- w_resid
- gps_mx (min and max of gps)
- w_mx (min and max of w).

`ci_appr` Causal inference approach.

`gps_density` Model type which is used for estimating GPS value, including `normal` and `kernel`.

`exposure_col_name` Exposure data column name.

`nthread` An integer value that represents the number of threads to be used by internal packages.

`...` Additional parameters.

Details

For matching approach, use an extra parameter, `bin_seq`, which is sequence of w (treatment) to generate pseudo population. If `NULL` is passed the default value will be used, which is `seq(min(w)+delta_n/2,max(w), by=delta_n)`.

Value

`compile_pseudo_pop` returns the pseudo population data that is compiled based on the selected causal inference approach.

Examples

```
set.seed(112)
m_d <- generate_syn_data(sample_size = 100)

m_xgboost <- function(nthread = 1,
                      ntrees = 35,
                      shrinkage = 0.3,
                      max_depth = 5,
                      ...) {SuperLearner::SL.xgboost(
                        nthread = nthread,
                        ntrees = ntrees,
                        shrinkage=shrinkage,
                        max_depth=max_depth,
                        ...)}

data_with_gps <- estimate_gps(.data = m_d,
                              .formula = w ~ cf1 + cf2 + cf3 +
                                             cf4 + cf5 + cf6,
                              gps_density = "normal",
                              sl_lib = c("m_xgboost")
                             )
```

```
pd <- compile_pseudo_pop(data_obj = data_with_gps,
                         ci_appr = "matching",
                         gps_density = "normal",
                         bin_seq = NULL,
                         exposure_col_name = c("w"),
                         nthread = 1,
                         dist_measure = "l1",
                         covar_bl_method = 'absolute',
                         covar_bl_trs = 0.1,
                         covar_bl_trs_type= "mean",
                         delta_n = 0.5,
                         scale = 1)
```

`compute_counter_weight` *Compute counter or weight of data samples*

Description

Computes counter (for matching approach) or weight (for weighting) approach.

Usage

```
compute_counter_weight(gps_obj, ci_appr, nthread = 1, ...)
```

Arguments

`gps_obj`	A gps object that is generated with `estimate_gps` function. If it is provided, the number of iteration will forced to 1 (Default: NULL).
`ci_appr`	The causal inference approach. Possible values are: • "matching": Matching by GPS • "weighting": Weighting by GPS
`nthread`	An integer value that represents the number of threads to be used by internal packages.
`...`	Additional arguments passed to different models.

Details

Additional parameters:

Causal Inference Approach (ci_appr):

- if ci_appr = 'matching':
  - *bin_seq*: A sequence of w (treatment) to generate pseudo population. If `NULL` is passed the default value will be used, which is `seq(min(w)+delta_n/2,max(w), by=delta_n)`.
  - *dist_measure*: Matching function. Available options:

    * l1: Manhattan distance matching
  - *delta_n*: caliper parameter.
  - *scale*: a specified scale parameter to control the relative weight that is attributed to the distance measures of the exposure versus the GPS.

Value

Returns a counter_weight (cgps_cw) object that includes `.data` and `params` attributes.

- `.data`: includes `id` and `counter_weight` columns. In case of `matching` the `counter_weight` column is integer values, which represent how many times the provided observational data was mached during the matching process. In case of `weighting` the column is double values.
- `params`: Include related parameters that is used for the process.

Examples

```
m_d <- generate_syn_data(sample_size = 100)
gps_obj <- estimate_gps(.data = m_d,
                        .formula = w ~ cf1 + cf2 + cf3 + cf4 + cf5 + cf6,
                        gps_density = "normal",
                        sl_lib = c("SL.xgboost"))

cw_object <- compute_counter_weight(gps_obj = gps_obj,
                                    ci_appr = "matching",
                                    bin_seq = NULL,
                                    nthread = 1,
                                    delta_n = 0.1,
                                    dist_measure = "l1",
                                    scale = 0.5)
```

`estimate_erf` *Estimate Exposure Response Function*

Description

Estimates the exposure-response function (ERF) for a matched and weighted dataset using parametric, semiparametric, and nonparametric models.

Usage

```
estimate_erf(.data, .formula, weights_col_name, model_type, w_vals, ...)
```

Arguments

`.data`	A data frame containing an observed continuous exposure variable, weights, and an observed outcome variable. Includes an `id` column for future reference.
`.formula`	A formula specifying the relationship between the exposure variable and the outcome variable. For example, Y ~ w.
`weights_col_name`	A string representing the weight or counter column name in `.data`.
`model_type`	A string representing the model type based on preliminary assumptions, including `parametric`, `semiparametric`, and `nonparametric` models.
`w_vals`	A numeric vector of values at which you want to calculate the ERF.
`...`	Additional arguments passed to the model.

Value

Returns an S3 object containing the following data and parameters:

- .data_original <- result_data_original
- .data_prediction <- result_data_prediction
- params

`estimate_gps` *Estimate generalized propensity score (GPS) values*

Description

Estimates GPS value for each observation using normal or kernel approaches.

Usage

```
estimate_gps(
  .data,
  .formula,
  gps_density = "normal",
  sl_lib = c("SL.xgboost"),
  ...
)
```

Arguments

`.data`	A data frame of observed continuous exposure variable and observed covariates variable. Also includes `id` column for future references.
`.formula`	A formula specifying the relationship between the exposure variable and the covariates. For example, w ~ I(cf1^2) + cf2.
`gps_density`	Model type which is used for estimating GPS value, including `normal` (default) and `kernel`.
`sl_lib`	A vector of prediction algorithms to be used by the SuperLearner packageg.
`...`	Additional arguments passed to the model.

Value

The function returns a S3 object. Including the following:

- `.data` : `id`, `exposure_var`, `gps`, `e_gps_pred`, `e_gps_std_pred`, `w_resid`
- `params`: Including the following fields:
  - gps_mx (min and max of gps)
  - w_mx (min and max of w).
  - .formula
  - gps_density
  - sl_lib
  - fcall (function call)

Examples

```
m_d <- generate_syn_data(sample_size = 100)
data_with_gps <- estimate_gps(.data = m_d,
                              .formula = w ~ cf1 + cf2 + cf3 + cf4 + cf5 + cf6,
                              gps_density = "normal",
                              sl_lib = c("SL.xgboost")
                             )
```

---

generate_pseudo_pop *Generate pseudo population*

Description

Generates pseudo population data set based on user-defined causal inference approach. The function uses an adaptive approach to satisfies covariate balance requirements. The function terminates either by satisfying covariate balance or completing the requested number of iteration, whichever comes first.

Usage

```
generate_pseudo_pop(
  .data,
  cw_obj,
  covariate_col_names,
  covar_bl_trs = 0.1,
  covar_bl_trs_type = "maximal",
  covar_bl_method = "absolute"
)
```

Arguments

.data	A data.frame of observation data with id column.
cw_obj	An S3 object of counter_weight.
covariate_col_names	A list of covariate columns.
covar_bl_trs	Covariate balance threshold
covar_bl_trs_type	Type of the covariance balance threshold.
covar_bl_method	Covariate balance method.

Value

Returns a pseudo population (gpsm_pspop) object that is generated or augmented based on the selected causal inference approach (ci_appr). The object includes the following objects:

- params
  - ci_appr
  - params
- pseudo_pop
- adjusted_corr_results
- original_corr_results
- best_gps_used_params
- effect size of generated pseudo population

Examples

```
set.seed(967)

m_d <- generate_syn_data(sample_size = 200)
m_d$id <- seq_along(1:nrow(m_d))

m_xgboost <- function(nthread = 4,
                      ntrees = 35,
                      shrinkage = 0.3,
                      max_depth = 5,
                      ...) {SuperLearner::SL.xgboost(
                        nthread = nthread,
                        ntrees = ntrees,
                        shrinkage=shrinkage,
                        max_depth=max_depth,
                        ...)}

data_with_gps_1 <- estimate_gps(
  .data = m_d,
  .formula = w ~ I(cf1^2) + cf2 + I(cf3^2) + cf4 + cf5 + cf6,
```

```
  sl_lib = c("m_xgboost"),
  gps_density = "normal")

cw_object_matching <- compute_counter_weight(gps_obj = data_with_gps_1,
                                             ci_appr = "matching",
                                             bin_seq = NULL,
                                             nthread = 1,
                                             delta_n = 0.1,
                                             dist_measure = "l1",
                                             scale = 0.5)

pseudo_pop <- generate_pseudo_pop(.data = m_d,
                                  cw_obj = cw_object_matching,
                                  covariate_col_names = c("cf1", "cf2",
                                                          "cf3", "cf4",
                                                          "cf5", "cf6"),
                                  covar_bl_trs = 0.1,
                                  covar_bl_trs_type = "maximal",
                                  covar_bl_method = "absolute")
```

`generate_syn_data` *Generate synthetic data for the CausalGPS package*

Description

Generates synthetic data set based on different GPS models and covariates.

Usage

```
generate_syn_data(
  sample_size = 1000,
  outcome_sd = 10,
  gps_spec = 1,
  cova_spec = 1,
  vectorized_y = FALSE
)
```

Arguments

`sample_size`	A positive integer number that represents a number of data samples.
`outcome_sd`	A positive double number that represents standard deviation used to generate the outcome in the synthetic data set.
`gps_spec`	A numerical integer values ranging from 1 to 7. The complexity and form of the relationship between covariates and treatment variables are determined by the `gps_spec`. Below, you will find a concise definition for each of these values:

- *gps_spec: 1*: The treatment is generated using a normal distributionMay 24, 2023 (`stats::rnorm`) and a linear function of covariates (cf1 to cf6).
- *gps_spec: 2*: The treatment is generated using a Student's t-distribution (`stats::rt`) and a linear function of covariates, but is also truncated to be within a specific range (-5 to 25).
- *gps_spec: 3*: The treatment includes a quadratic term for the third covariate.
- *gps_spec: 4*: The treatment is calculated using an exponential function within a fraction, creating logistic-like model.
- *gps_spec: 5*: The treatment also uses logistic-like model but with different parameters.
- *gps_spec: 6*: The treatment is calculated using the natural logarithm of the absolute value of a linear combination of the covariates.
- *gps_spec: 7*: The treatment is generated similarly to `gps_spec = 2`, but without truncation.

`cova_spec` A numerical value (1 or 2) to modify the covariates. It determines how the covariates in the synthetic data set are transformed. If `cova_spec` equals 2, the function applies non-linear transformation to the covariates, which can add complexity to the relationships between covariates and outcomes in the synthetic data. See the code for more details.

`vectorized_y` A Boolean value indicates how Y internally is generated. (Default = `FALSE`). This parameter is introduced for backward compatibility. vectorized_y = `TRUE` performs better.

Value

`synthetic_data`: The function returns a data.frame saved the constructed synthetic data.

Examples

```
set.seed(298)
s_data <- generate_syn_data(sample_size = 100,
                            outcome_sd = 10,
                            gps_spec = 1,
                            cova_spec = 1)
```

---

get_logger *Get Logger Settings*

Description

Returns current logger settings.

Usage

```
get_logger()
```

Value

Returns a list that includes **logger_file_path** and **logger_level**.

Examples

```
set_logger("mylogger.log", "INFO")
log_meta <- get_logger()
```

`plot.cgps_cw` *Extend generic plot functions for cgps_cw class*

Description

A wrapper function to extend generic plot functions for cgps_cw class.

Usage

```
## S3 method for class 'cgps_cw'
plot(x, ...)
```

Arguments

`x`	A cgps_cw object.
`...`	Additional arguments passed to customize the plot.

Details

Additional parameters:

- *every_n*: Puts label to ID at every n interval (default = 10)
- *subset_id*: A vector of range of ids to be included in the plot (default = NULL)

Value

Returns a ggplot2 object, invisibly. This function is called for side effects.

plot.cgps_erf *Extend generic plot functions for cgps_cw class*

Description

A wrapper function to extend generic plot functions for cgps_cw class.

Usage

```
## S3 method for class 'cgps_erf'
plot(x, ...)
```

Arguments

x	A cgps_erf object.
...	Additional arguments passed to customize the plot.

Details

TBD

Value

Returns a ggplot2 object, invisibly. This function is called for side effects.

plot.cgps_gps *Extend generic plot functions for cgps_gps class*

Description

A wrapper function to extend generic plot functions for cgps_gps class.

Usage

```
## S3 method for class 'cgps_gps'
plot(x, ...)
```

Arguments

x	A cgps_gps object.
...	Additional arguments passed to customize the plot.

Value

Returns a ggplot2 object, invisibly. This function is called for side effects.

plot.cgps_pspop *Extend generic plot functions for cgps_pspop class*

Description

A wrapper function to extend generic plot functions for cgps_pspop class.

Usage

```
## S3 method for class 'cgps_pspop'
plot(x, ...)
```

Arguments

x	A cgps_pspop object.
...	Additional arguments passed to customize the plot.

Details

Additional parameters:

- *include_details*: If set to TRUE, the plot will include run details (Default = FALSE).

Value

Returns a ggplot2 object, invisibly. This function is called for side effects.

print.cgps_cw *Extend print function for cgps_cw object*

Description

Extend print function for cgps_cw object

Usage

```
## S3 method for class 'cgps_cw'
print(x, ...)
```

Arguments

x	A cgps_cw object.
...	Additional arguments passed to customize the results.

Value

No return value. This function is called for side effects.

print.cgps_erf *Extend print function for cgps_erf object*

Description

Extend print function for cgps_erf object

Usage

```
## S3 method for class 'cgps_erf'
print(x, ...)
```

Arguments

x	A cgps_erf object.
...	Additional arguments passed to customize the results.

Value

No return value. This function is called for side effects.

print.cgps_gps *Extend print function for cgps_gps object*

Description

Extend print function for cgps_gps object

Usage

```
## S3 method for class 'cgps_gps'
print(x, ...)
```

Arguments

x	A cgps_gps object.
...	Additional arguments passed to customize the results.

Value

No return value. This function is called for side effects.

print.cgps_pspop *Extend print function for cgps_pspop object*

Description

Extend print function for cgps_pspop object

Usage

```
## S3 method for class 'cgps_pspop'
print(x, ...)
```

Arguments

x	A cgps_pspop object.
...	Additional arguments passed to customize the results.

Value

No return value. This function is called for side effects.

set_logger *Set Logger Settings*

Description

Updates logger settings, including log level and location of the file.

Usage

```
set_logger(logger_file_path = "CausalGPS.log", logger_level = "INFO")
```

Arguments

logger_file_path	A path (including file name) to log the messages. (Default: CausalGPS.log)
logger_level	The log level. Available levels include: • TRACE • DEBUG • INFO (Default) • SUCCESS • WARN • ERROR • FATAL

Value

No return value. This function is called for side effects.

Examples

```
set_logger("Debug")
```

---

summary.cgps_cw *print summary of cgps_cw object*

Description

print summary of cgps_cw object

Usage

```
## S3 method for class 'cgps_cw'
summary(object, ...)
```

Arguments

object	A cgps_cw object.
...	Additional arguments passed to customize the results.

Value

Returns summary of data

---

summary.cgps_erf *print summary of cgps_erf object*

Description

print summary of cgps_erf object

Usage

```
## S3 method for class 'cgps_erf'
summary(object, ...)
```

Arguments

object	A cgps_erf object.
...	Additional arguments passed to customize the results.

Value

Returns summary of data

`summary.cgps_gps` *print summary of cgps_gps object*

Description

print summary of cgps_gps object

Usage

```
## S3 method for class 'cgps_gps'
summary(object, ...)
```

Arguments

`object`	A cgps_gps object.
`...`	Additional arguments passed to customize the results.

Value

Returns summary of data

`summary.cgps_pspop` *print summary of cgps_pspop object*

Description

print summary of cgps_pspop object

Usage

```
## S3 method for class 'cgps_pspop'
summary(object, ...)
```

Arguments

`object`	A cgps_pspop object.
`...`	Additional arguments passed to customize the results.

Value

Returns summary of data

synthetic_us_2010 *Public data set for air pollution and health studies, case study: 2010 county-Level data set for the contiguous United States*

Description

A dataset containing exposure, confounders, and outcome for causal inference studies. The dataset is hosted on Harvard dataverse doi:10.7910/DVN/L7YF2G. This dataset was produced from five different resources. Please see https://github.com/NSAPH-Projects/synthetic_data/ for the data processing pipelines. In the following

Exposure Data

The exposure parameter is PM2.5. Di et al. (2019) provided daily, and annual PM2.5 estimates at 1 km×1 km grid cells in the entire United States. The data can be downloaded from Di et al. (2021). Features in this category starts with *qd_* prefix.

Census Data

The main reference for getting the census data is the United States Census Bureau. There are numerous studies and surveys for different geographical resolutions. We use 2010 county level American County Survey at the county level (acs5). Features in this category starts with *cs_* prefix.

CDC Data

The Centers for Disease Control and Prevention (CDC), provides the Behavioral Risk Factor Surveillance System (Centers for Disease Control and Prevention (2021)), which is the nation's premier system of health-related telephone surveys that collect state data about U.S. residents regarding their health-related risk behaviors.

GridMET Data

Climatology Lab at the University of California, Merced, provides the GridMET data (Abatzoglou (2013)). The data set is daily surface meteorological data covering the contiguous United States.

CMS Data

The Centers for Medicare and Medicaid Services(CMS) provides synthetic data at the county level for 2008-2010 (Centers for Medicare & Medicaid Services (2021)).

The definition of each variables are provided below. All data are collected for 2010 and aggregated into the county level and in the contiguous United States.

Usage

```
data(synthetic_us_2010)
```

Format

A data frame with 3109 rows and 46 variables:

qd_mean_pm25 Mean PM2.5 (microgram/m3)

cs_poverty The proportion of below poverty level population among 65+ years old.

cs_hispanic The proportion of Hispanic or Latino population among 65+ years old.

cs_black The proportion of Black or African American population among 65+ years old.

cs_white The proportion of White population among 65 years and over.

cs_native The proportion of American Indian or Alaska native population among 65 years and over.

cs_asian The proportion of Asian population among 65 years and over.

cs_other The proportion of other races population among 65 years and over.

cs_ed_below_highschool The proportion of the population with below high school level education among 65 years and over.

cs_household_income Median Household income in the past 12 months (in 2010 inflation-adjusted dollars) where householder is 65 years and over.

cs_median_house_value Median house value (USD)

cs_total_population Total Population

cs_area Area of each county (square miles)

cs_population_density The number of the population in one square mile.

cdc_mean_bmi Body Mass Index.

cdc_pct_cusmoker The proportion of current smokers.

cdc_pct_sdsmoker The proportion of some days smokers.

cdc_pct_fmsmoker The proportion of former smokers.

cdc_pct_nvsmoker The proportion of never smokers.

cdc_pct_nnsmoker The proportion of not known smokers.

gmet_mean_tmmn Annual mean of daily minimum temperature (K)

gmet_mean_summer_tmmn The mean of daily minimum temperature during summer (K)

gmet_mean_winter_tmmn The mean of daily minimum temperature during winter (K)

gmet_mean_tmmx Annual mean of daily maximum temperature (K)

gmet_mean_summer_tmmx The mean of daily maximum temperature during summer (K)

gmet_mean_winter_tmmx The mean of daily maximum temperature during winter (K)

gmet_mean_rmn Annual mean of daily minimum relative humidity (%)

gmet_mean_summer_rmn The mean of daily minimum relative humidity during summer (%)

gmet_mean_winter_rmn The mean of daily minimum relative humidity during winter (%)

gmet_mean_rmx Annual mean of daily maximum relative humidity (%)

gmet_mean_summer_rmx The mean of daily maximum relative humidity during summer (%)

gmet_mean_winter_rmx The mean of daily maximum relative humidity during winter (%)

gmet_mean_sph Annual mean of daily mean specific humidity (kg/kg)

gmet_mean_summer_sph The mean of daily mean specific humidity during summer(kg/kg)

gmet_mean_winter_sph The mean of daily mean specific humidity during winter(kg/kg)

cms_mortality_pct The proportion of deceased patients.

cms_white_pct The proportion of White patients.

cms_black_pct The proportion of Black patients.

cms_hispanic_pct The proportion of Hispanic patients.

cms_others_pct The proportion of Other patients.

cms_female_pct The proportion of Female patients.

region The region that the county is located in.

```
NORTHEAST=("NY","MA","PA","RI","NH","ME","VT","CT","NJ")
SOUTH=("DC","VA","NC","WV","KY","SC","GA","FL","AL","TN","MS","AR","MD","DE","OK","TX","LA")
MIDWEST=c("OH","IN","MI","IA","MO","WI","MN","SD","ND","IL","KS","NE")
WEST=c("MT","CO","WY","ID","UT","NV","CA","OR","WA","AZ","NM")
```

FIPS Federal Information Processing Standards, a unique ID for each county.

NAME County, State name.

STATE State abbreviation.

STATE_CODE State numerical code.

References

Abatzoglou, John T. 2013. "Development of Gridded Surface Meteorological Data for Ecological Applications and Modelling." International Journal of Climatology 33 (1): 121–31. doi:10.1002/joc.3413.

Centers for Disease Control and Prevention. 2021. "Behavioral Risk Factor Surveillance System." https://www.cdc.gov/brfss/annual_data/annual_2010.htm/.

Centers for Medicare & Medicaid Services. 2021. "CMS 2008-2010 Data Entrepreneurs' Synthetic Public Use File (DE-SynPUF)." https://www.cms.gov/data-research/statistics-trends-and-reports/medicare-claims-synthetic-public-use-files/cms-2008-2010-data-entrepreneurs-synthetic-public-use-f

Di, Qian, Heresh Amini, Liuhua Shi, Itai Kloog, Rachel Silvern, James Kelly, M Benjamin Sabath, et al. 2019. "An Ensemble-Based Model of Pm2. 5 Concentration Across the Contiguous United States with High Spatiotemporal Resolution." Environment International 130: 104909. doi:10.1016/j.envint.2019.104909.

Di, Qian, Yaguang Wei, Alexandra Shtein, Carolynne Hultquist, Xiaoshi Xing, Heresh Amini, Liuhua Shi, et al. 2021. "Daily and Annual Pm2.5 Concentrations for the Contiguous United States, 1-Km Grids, V1 (2000 - 2016)." NASA Socioeconomic Data; Applications Center (SEDAC). doi:10.7927/0rvr4538.

---

trim_it *Trim a data frame or an S3 object*

Description

Trims a data frame or an S3 object's `.data` attributs.

Usage

```
trim_it(data_obj, trim_quantiles, variable)
```

Arguments

`data_obj`	A data frame or an S3 object containing the data to be trimmed. For a data frame, the function operates directly on it. For an S3 object, the function expects a `.data` attribute containing the data.
`trim_quantiles`	A numeric vector of length 2 specifying the lower and upper quantiles used for trimming the data.
`variable`	The name of the variable in the data on which the trimming is to be applied.

Value

Returns a trimmed data frame or an S3 object with the $.data attribute trimmed, depending on the input type.

Examples

```
# Example usage with a data frame
df <- data.frame(id = 1:10, value = rnorm(100))
trimmed_df <- trim_it(df, c(0.1, 0.9), "value")

# Example usage with an S3 object
data_obj <- list()
class(data_obj) <- "myobject"
data_obj$.data <- df
trimmed_data_obj <- trim_it(data_obj, c(0.1, 0.9), "value")
```

Index

∗ **datasets**
synthetic_us_2010, 23

absolute_corr_fun, 3
absolute_weighted_corr_fun, 4

CausalGPS *(*CausalGPS-package*)*, 3
CausalGPS-package, 3
check_covar_balance, 5
compile_pseudo_pop, 7
compute_counter_weight, 9

estimate_erf, 10
estimate_gps, 11

generate_pseudo_pop, 12
generate_syn_data, 14
get_logger, 15

plot.cgps_cw, 16
plot.cgps_erf, 17
plot.cgps_gps, 17
plot.cgps_pspop, 18
print.cgps_cw, 18
print.cgps_erf, 19
print.cgps_gps, 19
print.cgps_pspop, 20

set_logger, 20
summary.cgps_cw, 21
summary.cgps_erf, 21
summary.cgps_gps, 22
summary.cgps_pspop, 22
synthetic_us_2010, 23

trim_it, 25