Package ‘CausalGPS’

February 7, 2022

Type Package

Title Matching on Generalized Propensity Scores with Continuous Exposures

Version 0.2.7

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.


License GPL-3

Language en-US

URL https://github.com/fasrc/CausalGPS

BugReports https://github.com/fasrc/CausalGPS/issues

Copyright Harvard University

Imports parallel, data.table, SuperLearner, xgboost, earth, ranger, gam, KernSmooth, MASS, polycor, wCorr, stats, ggplot2, rlang, logger, Rcpp, gnm, tidyr

Encoding UTF-8

RoxygenNote 7.1.1

Suggests covr, knitr, rmarkdown, testthat

VignetteBuilder knitr

Depends R (>= 3.5.0)

LinkingTo Rcpp

NeedsCompilation yes

Author Naeem Khoshnevis [aut, cre] (<https://orcid.org/0000-0003-4315-1426>, FASRC), Xiao Wu [aut] (<https://orcid.org/0000-0002-4884-657X>, HSPH), Danielle Braun [aut] (<https://orcid.org/0000-0002-5177-8598>, HSPH)
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
absolute_corr_fun

References


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 table 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 covairates.

Examples

```r
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)
data.table::setDT(mydata)
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)
data.table::setDT(mydata)
cor_val <- absolute_weighted_corr_fun(mydata[,2],
                           data.table::data.table(runif(n)),
                           mydata[, 3:length(mydata)])
print(cor_val$mean_absolute_corr)
check_cовар_balance  

Description

Checks the covariate balance of original population or pseudo population.

Usage

check_cовар_balance(pseudo_pop, ci_appr, nthread = 1, optimized_compile, ...)

Arguments

pseudo_pop  The generated pseudo population. In the following format:
  • 1st column: outcome (Y)
  • 2nd column: exposure (w)
  • 3rd column: gps
  • 4th column to the end: covariates (c)

ci_appr  The causal inference approach.

nthread  The number of available threads.

optimized_compile  If TRUE, use optimized compile approach.

...  Additional arguments passed to different models.

Details

Additional parameters:

• For ci_appr == matching:
  – covar_bl_method
  – covar_bl_trs

Value

output object:

• corr_results
  – absolute_corr
  – mean_absolute_corr
• pass (TRUE,FALSE)
Examples

```r
set.seed(422)
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)
pseudo_pop <- generate_pseudo_pop(mydata$Y,
                                  mydata$treat,
                                  mydata[c("cf1","cf2","cf3","cf4","cf5","cf6","year","region")],
                                  ci_appr = "matching",
                                  pred_model = "sl",
                                  gps_model = "non-parametric",
                                  trim_quantiles = c(0.01,0.99),
                                  optimized_compile = TRUE,
                                  sl_lib = c("m_xgboost"),
                                  covar_bl_method = "absolute",
                                  covar_bl_trs = 0.1,
                                  covar_bl_trs_type = "mean",
                                  max_attempt = 1,
                                  matching_fun = "matching_l1",
                                  delta_n = 1,
                                  scale = 0.5,
                                  nthread = 1)
adjusted_corr_obj <- check_covar_balance(pseudo_pop$pseudo_pop,
                                         ci_appr="matching",
                                         nthread=1,
                                         covar_bl_method = "absolute",
                                         covar_bl_trs = 0.1,
                                         covar_bl_trs_type = "mean",
                                         optimized_compile=FALSE)
```

**Description**

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

**Usage**

```r
compile_pseudo_pop(
  dataset,
  ci_appr,
  ...)```
Arguments

dataset List of size 6 including the following:
  • Original data set + GPS values (Y, w, GPS, counter, row_index, c)
  • 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_model Model type which is used for estimating GPS value, including parametric and non-parametric.

bin_seq 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).

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

trim_quantiles A numerical vector of two. Represents the trim quantile level. Both numbers should be in the range of [0,1] and in increasing order (default: c(0.01,0.99)).

optimized_compile If TRUE, uses counts to keep track of number of replicated pseudo population.

Value

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

Note

The input data set should be output of estimate_gps function with internal_use flag activated.

Examples

m_d <- generate_syn_data(sample_size = 100)
data_with_gps <- estimate_gps(m_d$Y,
  m_d$treat,
  m_d[\text{c("cf1","cf2","cf3","cf4","cf5","cf6")}],
pred_model = "sl"
```
gps_model = "parametric",
internal_use = TRUE,
params = list(xgb_max_depth = c(3,4,5),
               xgb_nrounds=c(10,20,30,40,50,60)),
nthread = 1,
sl_lib = c("m_xgboost")
)
```

```
pd <- compile_pseudo_pop(dataset = data_with_gps,
  ci_appr = "matching",
  gps_model = "parametric",
  bin_seq = NULL,
nthread = 1,
  trim_quantiles = c(0.01, 0.99),
  optimized_compile=TRUE,
  matching_fun = "matching_l1",
  covar_bl_method = 'absolute',
  covar_bl_trs = 0.1,
  covar_bl_trs_type= "mean",
  delta_n = 0.5,
  scale = 1)
```

---

**estimate_gps**  

*Estimate GPS Values*

**Description**

Estimates GPS value for each observation using parametric or non-parametric approaches.

**Usage**

```
estimate_gps(  
  Y,  
  w,  
  c,  
  pred_model,  
  gps_model = "parametric",  
  internal_use = TRUE,  
  params = list(),  
  nthread = 1,  
  ...  
)
```

**Arguments**

- **Y**  
  
  A vector of observed outcome variable.

- **w**  
  
  A vector of observed continuous exposure variable.
### estimate_gps

- **c**: A data frame of observed covariates variable.
- **pred_model**: The selected prediction model.
- **gps_model**: Model type which is used for estimating GPS value, including parametric (default) and non-parametric.
- **internal_use**: If TRUE will return helper vectors as well. Otherwise, will return original data + GPS value.
- **params**: Includes list of params that is used internally. Unrelated parameters will be ignored.
- **nthread**: An integer value that represents the number of threads to use by internal packages.
- **...**: Additional arguments passed to the model.

### Value

The function returns a list of 6 objects according to the following order:

- Original data set + GPS, counter, row_index values (Y, w, GPS, counter, row_index, c)
- e_gps_pred
- e_gps_std_pred
- w_resid
- gps_mx (min and max of gps)
- w_mx (min and max of w). If **internal_use** is set to be FALSE, only original data set + GPS will be returned.

### Examples

```r
m_d <- generate_syn_data(sample_size = 100)
data_with_gps <- estimate_gps(m_d$Y,
    m_d$treat,
    m_d[c("cf1","cf2","cf3","cf4","cf5","cf6")],
    pred_model = "sl",
    gps_model = "parametric",
    internal_use = FALSE,
    params = list(xgb_max_depth = c(3,4,5),
                  xgb_n_rounds=c(10,20,30,40,50,60)),
    nthread = 1,
    sl_lib = c("m_xgboost")
)
```
**estimate_npmetric_erf**  
*Estimate Smoothed Exposure-Response Function (ERF) for Matched Data Set.*

**Description**

Estimate smoothed exposure-response function (ERF) for matched and weighted data set using non-parametric models.

**Usage**

```r
estimate_npmetric_erf(
  matched_Y,
  matched_w,
  matched_counter = NULL,
  bw_seq = seq(0.2, 2, 0.2),
  w_vals,
  nthread
)
```

**Arguments**

- `matched_Y`  
a vector of outcome variable in the matched set.
- `matched_w`  
a vector of continuous exposure variable in the matched set.
- `matched_counter`  
a vector of counter variable in the matched set.
- `bw_seq`  
a vector of bandwidth values (Default is seq(0.2,2,0.2)).
- `w_vals`  
a vector of values that you want to calculate the values of the ERF at.
- `nthread`  
number of available cores.

**Details**

Estimate Functions Using Local Polynomial kernel regression Package: `KernSmooth`.

**Value**

The function returns a gpsm_erp object. The object includes the following attributes:

- `params`
- `matched_Y`
- `matched_w`
- `bw_seq`
- `w_vals`
- `erf`
- `fcall`
estimate_pmetric_erf

Examples

```r
m_d <- generate_syn_data(sample_size = 100)
pseudo_pop <- generate_pseudo_pop(m_d$Y,
  m_d$treat,
  m_d[c("cf1","cf2","cf3","cf4","cf5","cf6")],
  ci_appr = "matching",
  pred_model = "sl",
  sl_lib = c("m_xgboost"),
  params = list(xgb_nrounds=c(10,20,30),
                xgb_eta=c(0.1,0.2,0.3)),
  nthread = 1,
  covar_bl_method = "absolute",
  covar_bl_trs = 0.1,
  covar_bl_trs_type="mean",
  max_attempt = 1,
  matching_fun = "matching_l1",
  delta_n = 1,
  scale = 0.5)

erf_obj <- estimate_pnpmetric_erf(pseudo_pop$pseudo_pop$Y,
  pseudo_pop$pseudo_pop$w,
  bw_seq=seq(0.2,2,0.2),
  w_vals = seq(2,20,0.5),
  nthread = 1)
```

estimate_pmetric_erf

Estimate Parametric Exposure Response Function

Description

Estimate a constant effect size for matched and weighted data set using parametric models

Usage

```r
estimate_pmetric_erf(formula, family, data, ci_appr)
```

Arguments

- `formula`: a vector of outcome variable in matched set.
- `family`: a description of the error distribution (see ?gnm)
- `data`: dataset that formula is build upon
- `ci_appr`: causal inference approach (matching or weighting).

Details

This method uses generalized nonlinear model (gmm) from gnm package.
estimate_semipmetric_erf

Value

returns an object of class gnm

Examples

```r
m_d <- generate_syn_data(sample_size = 100)
pseudo_pop <- generate_pseudo_pop(m_d$Y, m_d$treat,
   m_d[,c("cf1","cf2","cf3","cf4","cf5","cf6")],
   ci_appr = "matching",
   pred_model = "sl",
   sl_lib = c("m_xgboost"),
   params = list(xgb_nrounds=c(10,20,30),
                 xgb_eta=c(0.1,0.2,0.3)),
   nthread = 1,
   covar_bl_method = "absolute",
   covar_bl_trs = 0.1,
   covar_bl_trs_type = "mean",
   max_attempt = 1,
   matching_fun = "matching_l1",
   delta_n = 1,
   scale = 0.5)
outcome_m <- estimate_pmetric_erf(formula = Y ~ w,
                                    family = gaussian,
                                    data = pseudo_pop$pseudo_pop,
                                    ci_appr = "matching")
```

estimate_semipmetric_erf


Description

Estimates the smoothed exposure-response function using a generalized additive model with splines.

Usage

```r
estimate_semipmetric_erf(formula, family, data, ci_appr)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>formula</td>
<td>a vector of outcome variable in matched set.</td>
</tr>
<tr>
<td>family</td>
<td>a description of the error distribution (see ?gam).</td>
</tr>
<tr>
<td>data</td>
<td>dataset that formula is build upon.</td>
</tr>
<tr>
<td>ci_appr</td>
<td>causal inference approach (matching or weighting).</td>
</tr>
</tbody>
</table>
**generate_pseudo_pop**

**Generate Pseudo Population**

**Details**

This approach uses Generalized Additive Model (gam) using mgcv package.

**Value**

returns an object of class gam

**Examples**

```r
m_d <- generate_syn_data(sample_size = 100)
pseudo_pop <- generate_pseudo_pop(m_d$Y,
  m_d$treat,
  m_d[c("cf1","cf2","cf3","cf4","cf5","cf6")],
  ci_appr = "matching",
  pred_model = "sl",
  sl_lib = c("m_xgboost"),
  params = list(xgb_nrounds=c(10,20,30),
                xgb_eta=c(0.1,0.2,0.3)),
  nthread = 1,
  covar_bl_method = "absolute",
  covar_bl_trs = 0.1,
  covar_bl_trs_type = "mean",
  max_attempt = 1,
  matching_fun = "matching_l1",
  delta_n = 1,
  scale = 0.5)

outcome_m <- estimate_semipmetric_erf (formula = Y ~ w,
  family = gaussian,
  data = pseudo_pop$pseudo_pop,
  ci_appr = "matching")
```

**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**

```r
generate_pseudo_pop(
  Y,
  w,
```
generate_pseudo_pop

c, 
c_i_appr, 
pred_model, 
gps_model = "parametric", 
use_cov_transform = FALSE, 
transformers = list("pow2", "pow3"), 
bin_seq = NULL, 
trim_quantiles = c(0.01, 0.99), 
optimized_compile = FALSE, 
params = list(), 
nthread = 1, 
...
)

Arguments

Y A vector of observed outcome variable.
w A vector of observed continuous exposure variable.
c A data.frame or matrix of observed covariates variable.
ci_appr The causal inference approach. Possible values are:
  • "matching": Matching by GPS
  • "weighting": Weighting by GPS
  • "adjusting": Adjusting by GPS
pred_model a prediction model (use "sl" for SuperLearner)
gps_model Model type which is used for estimating GPS value, including parametric (default) and non-parametric.
use_cov_transform If TRUE, the function uses transformer to meet the covariate balance.
transformers A list of transformers. Each transformer should be a unary function. You can pass name of customized function in the quotes. Available transformers:
  • pow2: to the power of 2
  • pow3: to the power of 3
bin_seq 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).
trim_quantiles A numerical vector of two. Represents the trim quantile level. Both numbers should be in the range of [0,1] and in increasing order (default: c(0.01,0.99)).
optimized_compile If TRUE, uses counts to keep track of number of replicated pseudo population.
params Includes list of params that is used internally. Unrelated parameters will be ignored.
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':
  - matching_fun: Matching function. Available options:
    * matching_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.
  - 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)
  - max_attempt: maximum number of attempt to satisfy covariate balance.
  - See create_matching() for more details about the parameters and default values.
- if ci.appr = 'weighting':
  - covar_bl_method: Covariate balance method.
  - covar_bl_trs: Covariate balance threshold
  - max_attempt: Maximum number of attempt to satisfy covariate balance.

Prediction models (pred_model):

- if pred_model = 'sl':
  - sl_lib: A vector of prediction algorithms.

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
  - pred_model
  - params
- pseudo_pop
- adjusted_corr_results
- original_corr_results
- optimized_compile (True or False)

Examples

```r
m_d <- generate_syn_data(sample_size = 100)
pseuoo_pop <- generate_pseudo_pop(m_d$Y,
m_d$treat,
m_d[c("cf1","cf2","cf3","cf4","cf5","cf6")],
ci_appr = "matching",
```
generate_syn_data

Generate Synthetic Data for CausalGPS Package

Description

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

Usage

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

Arguments

- **sample_size**: Number of data samples.
- **outcome_sd**: Standard deviation used to generate the outcome in the synthetic data set.
- **gps_spec**: A numerical value (1-7) that indicates the GPS model used to generate synthetic data. See the code for more details.
- **cova_spec**: A numerical value (1-2) to modify the covariates. See the code for more details.

Value

- **synthetic_data**: The function returns a data.frame saved the constructed synthetic data.
Examples

```r
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**

```r
get_logger()
```

**Value**

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

**Examples**

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

---

### plot.gpsm_erf

*Extend generic plot functions for gpsm_erf class*

**Description**

A wrapper function to extend generic plot functions for `gpsm_erf` class.

**Usage**

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

**Arguments**

- `x` A `gpsm_erf` object.
- `...` Additional arguments passed to customize the plot.
Value
Returns a ggplot2 object, invisibly. This function is called for side effects.

---

plot.gpsm_pspop  
**Extend generic plot functions for gpsm_erf class**

Description
A wrapper function to extend generic plot functions for gpsm_erf class.

Usage
```r
## S3 method for class 'gpsm_pspop'
plot(x, ...)
```

Arguments
- `x`: A gpsm_erf object.
- `...`: Additional arguments passed to customize the plot.

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

---

print.gpsm_erf  
**Extend print function for gpsm_erf object**

Description
Extend print function for gpsm_erf object

Usage
```r
## S3 method for class 'gpsm_erf'
print(x, ...)
```

Arguments
- `x`: A gpsm_erf object.
- `...`: Additional arguments passed to customize the results.

Value
No return value. This function is called for side effects.
### print.gpsm_pspop

**Extend print function for gpsm_pspop object**

**Description**

Extend print function for gpsm_pspop object

**Usage**

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

**Arguments**

- `x`: A gpsm_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**

```r
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")

Description

print summary of gpsm_erf object

Usage

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

Arguments

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

Value

Returns summary of data

Description

print summary of gpsm_pspop object

Usage

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

Arguments

object A gpsm_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/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

```r
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_fnsmoker** 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=c("NY", "MA", "PA", "RI", "NH", "ME", "VT", "CT", "NJ")
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


Index

* datasets
  synthetic_us_2010, 21
  absolute_corr_fun, 3
  absolute_weighted_corr_fun, 4

CausalGPS (CausalGPS-package), 2
CausalGPS-package, 2
check_covar_balance, 5
compile_pseudo_pop, 6
create_matching(), 15

estimate_gps, 8
estimate_npmetric_erf, 10
estimate_pmetric_erf, 11
estimate_semipmetric_erf, 12

generate_pseudo_pop, 13
generate_syn_data, 16
get_logger, 17

plot.gpsm_erf, 17
plot.gpsm_pspop, 18
print.gpsm_erf, 18
print.gpsm_pspop, 19

set_logger, 19
summary.gpsm_erf, 20
summary.gpsm_pspop, 20
synthetic_us_2010, 21