Package ‘representr’

September 5, 2023

Type Package

Title Create Representative Records After Entity Resolution

Version 0.1.5

Description An implementation of Kaplan, Betancourt, Steorts (2022) <doi:10.1080/00031305.2022.2041482> that creates representative records for use in downstream tasks after entity resolution is performed. Multiple methods for creating the representative records (data sets) are provided.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports doParallel, foreach, dplyr, Rcpp

Suggests knitr, rmarkdown, ggplot2

LinkingTo Rcpp

VignetteBuilder knitr

NeedsCompilation yes

Author Andee Kaplan [aut, cre],
Brenda Betancourt [aut],
Rebecca C. Steorts [aut]

Maintainer Andee Kaplan <andee.kaplan@colostate.edu>

Repository CRAN

Date/Publication 2023-09-05 19:50:02 UTC

R topics documented:

`clust_composite` *Composite record from a cluster using a weighted average of each column values.*

Description

Composite record from a cluster using a weighted average of each column values.

Usage

```
clust_composite(
  cluster,
  col_type,
  weights = rep(1/nrow(cluster), nrow(cluster))
)
```

Arguments

`cluster`	A data frame of the clustered records
`col_type`	A vector encoding the column type for each column in the dataset. Can take values in "categorical", "ordinal", "string", or "numeric"
`weights`	A vector of length equal to the number of records in the cluster indicating the weight for each. Defaults to equal weight.

Value

Returns the composite record from an individual cluster.

#' @examples data("rl_reg1")

clusters <- split(rl_reg1, identity.rl_reg1) type <- c("string", "string", "numeric", "numeric", "numeric", "categorical", "ordinal", "numeric", "numeric")

clust_composite(clusters[[1]], type)

clust_proto_random *Prototype record from a cluster.*

Description

Prototype record from a cluster.

Usage

```
clust_proto_random(
  cluster,
  prob = rep(1/nrow(cluster), nrow(cluster)),
  id = TRUE
)

clust_proto_minimax(cluster, not_cluster, distance, id = TRUE, ...)

maxmin_compare(ties, not_cluster, distance, ...)

within_category_compare(ties, not_cluster, distance, ...)

random_compare(ties, not_cluster, distance, ...)
```

Arguments

`cluster`	A data frame of the clustered records.
`prob`	A vector of length `nrow(cluster)` that sums to 1, giving the probability of selection.
`id`	Logical indicator to return id of record selected (TRUE) or actual record (FALSE). Note, if returning id, must have original row numbers as rownames in each cluster.
`not_cluster`	A data frame of the records outside the cluster
`distance`	A distance function for comparing records
`...`	Additional arguments passed to the comparison function
`ties`	A data frame of the records that are tied

Value

If `id = FALSE`, returns the prototype record from an individual cluster. Otherwise, returns the record id of the prototype record for that cluster. If there is a tie in the minimax prototype method, then random selection is used to break the tie.

Examples

```
data("rl_reg1")

clusters <- split(rl_reg1, identity.rl_reg1)
clust_proto_random(clusters[[1]])


not_clusters <- lapply(seq_along(clusters), function(x){
if(nrow(clusters[[x]]) > 1)
  do.call(rbind, clusters[-x])
})
clust_proto_minimax(clusters[[1]], not_clusters[[1]], dist_binary)
```

`dist_binary` *The distance between two records*

Description

The distance between two records

Usage

```
dist_binary(a, b)

dist_col_type_slow(
  a,
  b,
  col_type,
  string_dist = utils::adist,
  weights = rep(1/length(a), length(a)),
  orders = NULL,
  ...
)
```

Arguments

`a`	Record a
`b`	Record b
`col_type`	A vector encoding the column type for each column in the dataset. Can take values in "categorical", "ordinal", "string", or "numeric"
`string_dist`	String distance function. Default is edit distance. Function must take at least two arguments (strings)
`weights`	A vector of weights for each column for making some column distances more important. Must sum to 1. Defaults to equal weights.
`orders`	A named list containing the order of the levels in each ordinal column. Defaults to NULL, which corresponds to no ordinal variables.
`...`	Additional parameters passed to string distance function.

Value

`binary_dist` returns a numeric value indicating how many discrepancies there are between two records.

`col_type_dist` return a numeric value of the weighted column type specific distance between two records.

Examples

```
data("rl_reg1")
dist_binary(rl_reg1[1,], rl_reg1[2,])

type <- c("string", "string", "numeric", "numeric",
    "numeric", "categorical", "ordinal", "numeric", "numeric")
order <- list(education = c("Less than a high school diploma",
    "High school graduates, no college", "Some college or associate degree",
    "Bachelor's degree only", "Advanced degree"))

dist_col_type_slow(rl_reg1[1,], rl_reg1[2,], col_type = type, order = order)
```

`dist_col_type` *dist_col_type Inner column type record distance function*

Description

dist_col_type Inner column type record distance function

Usage

```
dist_col_type(a, b, col_type, weights, orders)
```

Arguments

`a`	record a
`b`	record b
`col_type`	A vector encoding the column type for each column in the dataset. Can take values in "categorical", "ordinal", "string", or "numeric"
`weights`	A vector of weights for each column for making some column distances more important. Must sum to 1. Defaults to equal weights.
`orders`	A named list containing the order of the levels in each ordinal column. Defaults to NULL, which corresponds to no ordinal variables.

`emp_kl_div` *Calculate the empirical KL divergence for a representative dataset as compared to the true dataset*

Description

Calculate the empirical KL divergence for a representative dataset as compared to the true dataset

Usage

```
emp_kl_div(
  true_dat,
  rep_dat,
  categoric_vars,
  numeric_vars,
  l_m = 10,
  weights = rep(1, nrow(rep_dat))
)
```

Arguments

`true_dat`	The true dataset
`rep_dat`	A representative dataset
`categoric_vars`	A vector of column positions or column names for the categoric variables.
`numeric_vars`	A vector of column positions or column names for the numeric variables.
`l_m`	Approximate number of true data points to be in each bin for numeric variables. Default is 10.
`weights`	If weighted frequencies are desired, pass a vector weights of the same length as representative data points.

Details

This function computes the estimated the KL divergence of two samples of data using the empirical distribution functions for the representative data set and true data set with continuous variables transformed to categorical using a histogram approach with statistically equivalent data-dependent bins, as detailed in

Wang, Qing, Sanjeev R. Kulkarni, and Sergio Verdú. "Divergence estimation of continuous distributions based on data-dependent partitions." IEEE Transactions on Information Theory 51.9 (2005): 3064-3074.

Examples

```
data("rl_reg1")

## random prototyping
```

```
rep_dat_random <- represent(rl_reg1, identity.rl_reg1, "proto_random", id = FALSE, parallel = FALSE)

## empirical KL divergence
cat_vars <- c("sex")
num_vars <- c("income", "bp")
emp_kl_div(rl_reg1[unique(identity.rl_reg1), c(cat_vars, num_vars)],
          rep_dat_random[, c(cat_vars, num_vars)],
          cat_vars, num_vars)
```

---

pp_weights — *Get posterior weights for each record post record-linkage using posterior prototyping.*

Description

Get posterior weights for each record post record-linkage using posterior prototyping.

Usage

```
pp_weights(
  data,
  posterior_linkage,
  rep_method,
  parallel = TRUE,
  cores = NULL,
  ...,
  scale = FALSE,
  save_loc = NULL,
  verbose = FALSE
)
```

Arguments

`data`	A data frame of records to be represented.
`posterior_linkage`	A matrix of size m x n, indicating the posterior cluster ids post-record linkage, each row represents the cluster assignment for each record in `data` for 1 iteration of the sampler.
`rep_method`	Which method to use for representation. Valid options include "proto_minimax" and "proto_random".
`parallel`	Logical flag if to use parallel computation or not (via `foreach`).
`cores`	If specified, the number of cores to use with `foreach`.
`...`	Additional parameters sent to cluster representation function. See minimax or random methods. If passing a probability to the random method, must be list of the same length as the number of iterations in lambda and within each must

	be a list of the same length as the number of clusters. Within each should be a vector of probabilities, the same length as the number of rows in the cluster `prob[[iteration]][[cluster]]`.
`scale`	If "proto_minimax" method is specified, logical flag to indicate if the column-type distance function should be scaled so that each distance takes value in [0, 1]. Defaults to FALSE.
`save_loc`	Location to save intermediate progress. If NULL, no intermediate progress is saved.
`verbose`	Flag for progress messages.

Examples

```
data(rl_reg1)

# make a fake posterior distribution for the linkage
m <- 10
n <- nrow(rl_reg1)
post_link <- matrix(sample(seq_len(n), n*m, replace = TRUE), nrow = m)

# get the posterior prototyping weights
col_type <- c("string", "string", "numeric", "numeric", "numeric", "categorical", "ordinal",
    "numeric", "numeric")
orders <- list(education = c("Less than a high school diploma", "High school graduates, no college",
    "Some college or associate degree", "Bachelor's degree only", "Advanced degree"))
weights <- c(.25, .25, .05, .05, .1, .15, .05, .05, .05)


pp_weight <- pp_weights(rl_reg1, post_link, "proto_minimax", distance = dist_col_type,
   col_type = col_type, weights = weights, orders = orders, scale = TRUE, parallel = FALSE)

# threshold by posterior prototyping weights
head(rl_reg1[pp_weight > 0.5, ])
```

`represent` *Create a representative dataset post record-linkage.*

Description

Create a representative dataset post record-linkage.

Usage

```
represent(
  data,
  linkage,
  rep_method,
  parallel = TRUE,
  cores = NULL,
  ...,
  scale = FALSE
)
```

Arguments

`data`	A data frame of records to be represented.
`linkage`	A numeric vector indicating the cluster ids post-record linkage for each record in `data`.
`rep_method`	Which method to use for representation. Valid options include "proto_minimax", "proto_random", and "composite".
`parallel`	Logical flag if to use parallel computation or not (via `foreach`).
`cores`	If specified, the number of cores to use with `foreach`.
`...`	Additional parameters sent to cluster representation function. See prototyping or composite methods.
`scale`	If "proto_minimax" method is specified, logical flag to indicate if the column-type distance function should be scaled so that each distance takes value in [0, 1]. Defaults to FALSE.

Examples

```
data("rl_reg1")

## random prototyping
rep_dat_random <- represent(rl_reg1, identity.rl_reg1, "proto_random", id = FALSE, parallel = FALSE)
head(rep_dat_random)

## minimax prototyping
col_type <- c("string", "string", "numeric", "numeric", "numeric", "categorical", "ordinal",
    "numeric", "numeric")
orders <- list(education = c("Less than a high school diploma", "High school graduates, no college",
    "Some college or associate degree", "Bachelor's degree only", "Advanced degree"))
weights <- c(.25, .25, .05, .05, .1, .15, .05, .05, .05)
rep_dat_minimax <- represent(rl_reg1, identity.rl_reg1, "proto_minimax", id = FALSE,
    distance = dist_col_type, col_type = col_type, weights = weights, orders = orders,
    scale = TRUE, parallel = FALSE)
head(rep_dat_minimax)

## Not run:
## with alternative tie breaker
rep_dat_minimax <- represent(rl_reg1, identity.rl_reg1, "proto_minimax", id = FALSE,
    distance = dist_col_type, col_type = col_type, weights = weights, orders = orders,
    ties_fn = "maxmin_compare", scale = TRUE, parallel = FALSE)
head(rep_dat_minimax)
```

```
rep_dat_minimax <- represent(rl_reg1, identity.rl_reg1, "proto_minimax", id = FALSE,
    distance = dist_col_type, col_type = col_type, weights = weights, orders = orders,
    ties_fn = "within_category_compare_cpp", scale = TRUE, parallel = FALSE)
head(rep_dat_minimax)

## composite prototyping
rep_dat_composite <- represent(rl_reg1, identity.rl_reg1, "composite",
                               col_type = col_type, parallel = FALSE)
head(rep_dat_composite)

## End(Not run)
```

`representr` *representr: A package for creating representative records post-record linkage.*

Description

The representr package provides two types of representative record generation: point prototyping and posterior prototyping.

Point Prototyping

To bridge the gap between record linkage and a downstream task, there are three methods to choose or create the representative records from linked data: random prototyping, minimax prototyping, and composite. These are all based on a point estimate of the linkage structure post-record linkage (rather than a posterior distribution).

Random prototyping chooses a record from each cluster at random, either uniformly or according to a supplied distribution. **Minimax prototyping** selects the record whose farthest neighbors within the cluster is closest, based on some notion of closeness that is measured by a record distance function. There are two distance functions included in this package (binary and column-based), or the user can specify their own. **Composite** record creation constructs the representative record by aggregating the records (in each cluster) to form a composite record that includes information from each linked record.

Each of these three types of prototyping can be used from the function represent.

Posterior prototyping

The posterior distribution of the linkage can be used in two ways in this package. The first, is as weights or in a distance function for the above point prototyping methods. The second, is through the posterior prototyping (PP) weights presented in Kaplan, Betancourt, and Steorts (2018+). The PP weights are accessible through the pp_weights function.

References

Kaplan, Andee, Brenda Betancourt, and Rebecca C. Steorts. "Posterior Prototyping: Bridging the Gap between Bayesian Record Linkage and Regression." arXiv preprint arXiv:1810.01538 (2018).

`rl_reg1` *500 records suitable for record linkage with additional regression variables*

Description

Simulated datasets containing the name, birthdate, and additional attributes of 500 records, of which there are 350 unique individuals.

Usage

```
rl_reg1

rl_reg2

rl_reg5

identity.rl_reg1

identity.rl_reg2

identity.rl_reg5

linkage.rl
```

Format

`rl_reg1` and `rl_reg5` are data frames with 500 rows and 9 columns. Each row represents 1 records with the following columns:

fname First name

lname Last name

bm Birth month (numeric)

bd Birth day

by Birth year

sex Sex ("M" or "F")

education Education level ("Less than a high school diploma", ""High school graduates, no college", "Some college or associate degree", "Bachelor's degree only", or "Advanced degree")

income Yearly income (in 1000s)

bp Systolic blood pressure

`identity.rl_reg1` and `identity.rl_reg5` are integer vectors indicating the true record ids of the two datasets. Two records represent the same individual if and only if their corresponding identity values are equal.

`linkage.rl` contains the result of running 100,000 iterations of a record linkage model using the package `dblinkR`.

An object of class `data.frame` with 500 rows and 9 columns.

An object of class `data.frame` with 500 rows and 9 columns.

An object of class `integer` of length 500.

An object of class `integer` of length 500.

An object of class `integer` of length 500.

An object of class `matrix` (inherits from `array`) with 100000 rows and 500 columns.

Details

There is a known relationship between three of the variables in the dataset, blood pressure (bp), income, and sex.

$$bp = 160 + 10I(sex = "M") - income + 0.5income * I(sex = "M") + \epsilon$$

where ϵ $Normal(0, \sigma^2)$ and $\sigma = 1, 2, 5$.

The 150 duplicated records have randomly generated errors.

Source

Names and birthdates generated with the ANU Online Personal Data Generator and Corruptor (GeCO) version 0.1 https://dmm.anu.edu.au/geco/.

`within_category_compare_cpp`

within_category_compare_cpp Inner column type record distance function

---

Description

within_category_compare_cpp Inner column type record distance function

Usage

```
within_category_compare_cpp(
  ties,
  not_cluster,
  col_type,
  weights,
  orders,
  distance
)
```

Arguments

`ties`	A data frame of the records that are tied
`not_cluster`	A data frame of the records outside the cluster
`col_type`	A vector encoding the column type for each column in the dataset. Can take values in "categorical", "ordinal", "string", or "numeric"
`weights`	A vector of weights for each column for making some column distances more important. Must sum to 1. Defaults to equal weights.
`orders`	A named list containing the order of the levels in each ordinal column. Defaults to NULL, which corresponds to no ordinal variables.
`distance`	function that does nothing right now, but must be supplied to not break other code.

Index