Package 'edpclient'

August 25, 2017

Type Package

Title Empirical Data Platform Client

Version 0.2.0

Date 2017-08-24

Author Empirical Engineering

Maintainer Madeleine Thompson <madeleine@empirical.com>

Description R client for Empirical Data Platform. More information is at <https://empirical.com>. For support, contact support@empirical.com.

License Apache License (== 2.0)

Imports httr, ini, jsonlite, plyr, stats

Suggests lintr, testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2017-08-25 03:06:00 UTC

R topics documented:

`edpclient-package` *Empirical Data Platform Client*

Description

R client for Empirical Data Platform. More information is at <https://empirical.com>. For support, contact support@empirical.com.

Details

edpclient is the R client for Empirical Data Platform.

Author(s)

Empirical Engineering

Maintainer: Madeleine Thompson <madeleine@empirical.com>

`build_popmod` *Create a new population model*

Description

Ask the job server to start working on a model for a population.

Usage

```
build_popmod(pop, name, models = 16, builder = "crosscat", iterations = 10)
```

Arguments

`pop`	a population from `population` or `create_population`
`name`	a name for the new model
`models`	ensemble size
`builder`	builder, rarely changed from the default
`iterations`	number of Gibbs sweeps to perform

Value

A new EDP population model id. Eventually, this will return a population model object, but those cannot currently refer to un-built models. You can see whether the model is built yet with popmods.

See Also

create_population

---

col_assoc *Compute column associations*

Description

Computes a matrix of associations (like R-squared or mutual information) of a collection of columns.

Usage

```
col_assoc(pm, target = NULL, given = NULL,
          statistic = c("mutual information", "R squared", "classic dep prob"))
```

Arguments

pm	an edp_population_model object, from popmod
target	an optional list of column names, defaults to all columns
given	a list of values to condition on; `list(a = 3)` means to compute the association conditional on a being 3.
statistic	the name of the statistic to compute

Value

a square matrix of column associations, where the edge length is `length(target)`

---

create_population *Create a new population*

Description

Create a new population by uploading data and a schema.

Usage

```
create_population(sess, data, name)
```

Arguments

sess	an edp_session from edp_session(...)
data	a data frame containing the data to build models from
name	a length-one character naming the population

Details

This function derives a population schema from data, uploads the data and schema, and creates a new remote population object. You can optionally use functions like stat_type<- and display_name<- on columns of the data frame before calling create_population if you want to give it hints about what the schema should be.

Value

A new EDP population object. If you print it or cast it to character, you can get an id that will let you get the object back later with population.

See Also

edp_session, population, delete_population, stat_type, display_name, identifying

Examples

```
d <- data.frame(x = c(1, 2, 3, 4, 5))
display_name(d$x) <- "X!"
stat_type(d$x) <- "realMultiplicative"
## Not run: pop <- create_population(sess, d, "my population")
```

---

delete_population *Delete a population*

Description

Delete a population from EDP.

Usage

```
delete_population(pop)
```

Arguments

pop	a population from create_population or population

See Also

create_population, population

`display_name` *Display Names*

Description

Get or set display names of columns on a data frame

Usage

```
display_name(col)
display_name(col) <- value
display_names(data)
display_names(data) <- value
```

Arguments

`col`	a column in a data frame
`data`	a data frame
`value`	a character vector

Details

In EDP, columns in a population can have "display names," which are human-readable names, often longer than the regular `names`. `display_name` and `display_name<-` get and set these on a single column. `display_names` and `display_names<-` get and set them on all columns of a data frame at once. The setters are used when preparing a data frame to build a population from. The getters can be applied to results from `select` and `simulate`.

Value

`display_name` returns a single-element character-vector. `display_names` returns a character vector with as many elements as the data frame has columns. If a column has no display name set, the corresponding element will be `NA`.

Examples

```
d <- data.frame(inc = c(25, NA, 17), age = c(21, 70, 30))
display_name(d$inc) <- "income (thousands of USD)"
dn <- display_names(d)  # returns c(inc = "income (thousands of USD)", age = NA)
```

edp_session *edp_session objects*

Description

Create or test for objects of type edp_session. The edp_session function is the first entry point you'll have into edpclient.

Usage

```
edp_session(profile = "default")
is.edp_session(x)
```

Arguments

profile	A profile name, which is a section name in ~/.edp_auth
x	object to be tested

Value

edp_session creates an EDP session object. is.edp_session returns a logical indicating whether the argument came from edp_session.

Examples

```
## Not run: sess <- edp_session()
## Not run: is.edp_session(sess)  # TRUE
```

identifying *Identifying Columns*

Description

Get or set identifying columns on a data frame

Usage

```
identifying(col)
identifying(col) <- value
identifying_columns(data)
identifying_columns(data) <- value
```

Arguments

col	a column in a data frame
data	a data frame
value	a single logical in the case of identifying, or a character vector in the case of identifying_columns

Details

In EDP, "identifying columns" are columns that uniquely identify a row. When preparing a data frame to build a population from, you can mark columns as identifying. select and simulate return this annotation.

Value

identifying returns a single logical. identifying_columns returns a character vector.

Examples

```
d <- data.frame(stringsAsFactors = FALSE, a = c(1, 4), id = c("A", "B"))
identifying(d$id) <- TRUE
identifying_columns(d) <- c("id")
```

---

latest_popmod *Return latest population model*

Description

Returns the latest population model for a population

Usage

```
latest_popmod(pop)
```

Arguments

pop	a population object from population

Value

a population model object, as you would get from popmod, representing the most-recently-built model in pop

See Also

popmods

popmod *EDP population model objects*

Description

Create or test for popmod objects.

Usage

```
popmod(sess, pmid)
is.popmod(x)
```

Arguments

sess	an edp_session object from edp_session()
pmid	a population model id, which begins with "pm-"
x	object to be tested

Value

popmod creates an EDP population model object. is.popmod returns a logical indicating whether its argument came from popmod.

popmods *List population models*

Description

Lists population models from a population as a data frame

Usage

```
popmods(pop)
```

Arguments

pop	a population object from population()

Value

a data frame with column names "name", "id", "creation_date", and "build_status"

See Also

edp_session, population

population *EDP population objects*

Description

Create or test for `population` objects.

Usage

```
population(sess, pid)
is.population(x)
```

Arguments

`sess`	an `edp_session` from `edp_session(...)`
`pid`	a population id, a length-one character vector starting with "p-"
`x`	object to be tested

Value

`population` creates an EDP population object. `is.population` returns a logical indicating whether the argument came from `population`.

See Also

`edp_session`, `populations`

Examples

```
## Not run: p <- population(sess, "p-abcde0123456789")
## Not run: is.population(p)  # TRUE
```

populations *List populations*

Description

Lists populations as a data frame

Usage

```
populations(sess)
```

Arguments

`sess`	an `edp_session` object from `edp_session()`

Value

a data frame with column names "name", "id", "nmodels", and "creation_date"

See Also

edp_session, population

`predict.edp_population_model`
Predict values from a population model

Description

Predict values for the columns in `target` for `rowids` given the other values in that row.

Usage

```
## S3 method for class 'edp_population_model'
predict(object, ...,
        target = NULL, rowids = NULL,
        infer_present = FALSE, seed = NULL)
```

Arguments

`object`	an EDP population model, from `popmod(...)`
`...`	ignored, accepted for compatibility with `predict`
`target`	a character vector of column names, defaults to `names(pm)` minus `names(where)`
`rowids`	a integer vector of rowids, defaults to all
`infer_present`	If TRUE, act as though each column in `targets` is missing across all rows and so infer a value for every row. Otherwise only infer values which were missing in the original data.
`seed`	if set, an integer to pass to EDP as a random seed for this call

Value

a data frame with the columns specified in `target`.

Examples

```
## Not run:
  predict(pm, target = c("INCOME"), rowids = c(3, 7))

## End(Not run)
```

schema — *Return the schema of a population model*

Description

Returns the schema of a population model, as a data.frame.

Usage

```
schema(pm)
```

Arguments

pm	an EDP population model, from `popmod(...)`

Value

A data.frame with columns `name`, `display_name`, `stat_type`, and `nvalues`. `nvalues` is NA unless `stat_type` is a categorical type.

select — *Fetch columns from a population or population model*

Description

Fetch specified columns from a population or population model.

Usage

```
select(x, target = NULL, where = NULL, rowids = NULL)
```

Arguments

x	an EDP population from `population(...)` or an EDP population model from `popmod(...)`
target	a character vector of column names, defaults to `names(pm)` minus `names(where)`
where	a list of conditions to select on; `list(a = "x", c = 3)` means only to return rows where `a` is "x" and `c` is 3.
rowids	a list of integer row ids to return. These row ids match the row names returned by `select`. `rowids` is exclusive with `where`.

Value

a data frame with the columns specified in `target`.

Examples

```
  ## Not run:
    # similar to SQL "SELECT a, b FROM pm WHERE c = 4;"
    select(pm, c("a", "b"), where = list(c = 4))

    select(pm, c("a", "b"), rowids = c(7, 22))

## End(Not run)
```

`simulate.edp_population_model`
Fetch columns from a population model

Description

Fetch specified columns from a population model. This is the data the population model is trained on.

Usage

```
## S3 method for class 'edp_population_model'
simulate(object, nsim = 1, seed = NULL, ...,
         target = NULL, columns = NULL, given = NULL)
```

Arguments

`object`	an EDP population model, from `popmod(...)`
`nsim`	number of rows to simulate
`seed`	if set, an integer to pass to EDP as a random seed for this call
`...`	ignored, accepted for compatibility with `simulate`
`target`	a character vector of column names, defaults to `names(pm)`
`columns`	alias for `target` for compatibility with `simulate`
`given`	a single-row data frame of values to condition on; `data.frame(a = 3)` means to simulate from the conditional distribution given `a` is 3.

Value

a data frame with the columns specified in `target`.

stat_type *Stat Types*

Description

Get or set stat types of columns on a data frame

Usage

```
stat_type(col)
stat_type(col) <- value
```

Arguments

col	a column in a data frame
value	a single-element character vector

Details

In EDP, columns in a population have "stat types." Numeric columns are usually "realAdditive" and "realMultiplicative". Factors are usually "categorical." These functions get and set the stat type on a data frame you will build a population from; you can also get the stat type of a column from `select` or `simulate`.

Value

a single-element character vector, `NA` if no stat type was set

Examples

```
d <- data.frame(x = c(1000, NA, 10000))
stat_type(d$x) <- "realMultiplicative"
st <- stat_type(d$x)  # returns "realMultiplicative"
```

visibility *Visibility*

Description

Get or set visibility of populations

Usage

```
visibility(x)
add_reader(x, reader = NULL, reader_domain = NULL)
```

Arguments

x	a population (from population) or a population model (from popmod)
reader	email address of a person to give read access to
reader_domain	email domain of an organization to give read access to

Details

visibility and add_reader manipulate the permissions of a population. They can be applied to population models, but that is only a convenience; population models do not have ACLs independent of populations.

When calling add_reader, exactly one of reader and reader_domain must be specified.

Value

Both visibility and add_reader return an object with keys owner (a length-one character vector), public (a length-one logical vector), readers (a character vector of email addresses), and reader_domains (a character vector of email domains).

Examples

```
## Not run:
  pm <- popmod(sess, "p-0123456789abcdef")
  add_reader(pm, "test@example.com")
  add_reader(pm, reader_domain = "example.com")
  print(visibility(pm))

## End(Not run)
```

---

wait_for *Wait until a population model is built*

Description

Wait until a population model is built, possibly showing progress.

Usage

```
wait_for(pm, quiet = FALSE, seconds = Inf)
```

Arguments

pm	a population model, probably recently created with build_popmod, but any population model will do
quiet	if TRUE, no output is printed; if FALSE, a progress bar is shown
seconds	maximum number of seconds to wait; must be the default, Inf, if quiet is FALSE

Value

`pm`, possibly with an updated `build_status` attribute

See Also

`build_popmod`

Index