Package 'fxtract'

February 12, 2019

Type Package

Title Feature Extraction from Grouped Data

Date 2019-02-07

Description An R6 implementation for calculating features from grouped data.
The output will be one row for each group.
This functionality is often needed in the feature extraction process of machine learning problems.
Very large datasets are supported, since data is only read into RAM when needed.
Calculation can be done in parallel and the process can be monitored.
Global error handling is supported.
Results are available in one final dataframe.

Version 0.9.1

URL https://github.com/QuayAu/fxtract

BugReports https://github.com/QuayAu/fxtract/issues

NeedsCompilation yes

ByteCompile yes

Depends R (>= 3.4.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports dplyr, magrittr, checkmate, utils, data.table, future,
future.apply, R6

Suggests testthat, knitr, covr, pkgdown

VignetteBuilder knitr

Author Quay Au [aut, cre],
Clemens Stachl [ctb],
Ramona Schoedel [ctb],
Theresa Ullmann [ctb],
Andreas Hofheinz [ctb]

Maintainer Quay Au <quayau@gmail.com>

Repository CRAN

Date/Publication 2019-02-12 17:30:03 UTC

R topics documented:

`dplyr_wrapper` *Wrapper for dplyr's summarize*

Description

This function wraps dplyr's `summarize()` function in a convenient way. The user only needs to define functions on the dataset with a named vector or list (with atomic entries of length 1) as return.

Usage

```
dplyr_wrapper(data, group_by, fun, check_fun = TRUE)
```

Arguments

`data`	('dataframe'). A dataframe with a grouping variable.
`group_by`	('character()'). Name of column, which contains identifiers on which the dataset should be grouped by. E.g. different user IDs.
`fun`	('function'). Must be a function, which has a dataframe as input and a (named) vector of desired length as output.
`check_fun`	('logical(1)'). If `TRUE`, fun(data) will be evaluated and checked if the outcome is of correct form. Set to `FALSE` if evaluation on the whole dataset takes too long.

Value

('dataframe')

Examples

```
# Number of used chrome apps
fun1 = function(data) {
  c(uses_chrome = nrow(
    dplyr::filter(data, RUNNING_TASKS_baseActivity_mPackage == "com.android.chrome"))
  )
}
dplyr_wrapper(data = studentlife_small, group_by = "userId", fun = fun1)
```

```
# mean, max, sd of a column
fun2 = function(data) {
  c(mean_sepal_length = mean(data$Sepal.Length),
    max_sepal_length = max(data$Sepal.Length),
    sd_sepal_length = sd(data$Sepal.Length)
  )
}
dplyr_wrapper(data = iris, group_by = "Species", fun = fun2)

# return list
fun3 = function(data) {
  list(mean_sepal_length = mean(data$Sepal.Length),
    max_sepal_length = max(data$Sepal.Length),
    sd_sepal_length = sd(data$Sepal.Length)
  )
}
dplyr_wrapper(data = iris, group_by = "Species", fun = fun3)

# group by two columns
df = data.frame(id = c(rep(1, 10), rep(2, 10)))
df$task = rep(c(rep("task1", 5), rep("task2", 5)), 2)
df$hour = rep(c(rep("hour1", 3), rep("hour2", 2), rep("hour1", 2), rep("hour2", 3)), 2)
df$x = 1:20
fun4 = function(data) c(mean_x = mean(data$x))
dplyr_wrapper(data = df, group_by = c("id", "task"), fun = fun4)
```

`studentlife_small` *Exemplary logging data from smartphones.*

Description

Contains exemplary gps and app data from the studentlife dataset (users: u00, u01, u02)

Usage

```
studentlife_small
```

Format

An object of class `data.frame` with 162154 rows and 19 columns.

Details

http://studentlife.cs.dartmouth.edu/dataset.html

Wang, Rui, Fanglin Chen, Zhenyu Chen, Tianxing Li, Gabriella Harari, Stefanie Tignor, Xia Zhou, Dror Ben-Zeev, and Andrew T. Campbell. "StudentLife: Assessing Mental Health, Academic Performance and Behavioral Trends of College Students using Smartphones." In Proceedings of the ACM Conference on Ubiquitous Computing. 2014.

`Xtractor` *R6 Object for Feature Extraction.*

Description

`Xtractor` calculates features from raw data for each ID of a grouping variable individually. This process can be parallelized with the package future.

Format

`R6Class` object.

Usage

```
xtractor = Xtractor$new("xtractor")
```

Arguments

For Xtractor$new():

`name`: (‘character(1)‘): A user defined name of the Xtractor. All necessary data will be saved on the path: ./fxtract_files/name/

`load`: (‘logical(1)‘): If TRUE, an existing Xtractor will be loaded.

`file.dir`: (‘character(1)‘): Path where all files of the Xtractor are saved. Default is the current working directory.

Details

All datasets and feature functions are saved in this R6 object. Datasets will be saved as single RDS files (for each ID) and feature functions are calculated on each single dataset. A big advantage of this method is that it scales nicely for larger datasets. Data is only read into RAM, when needed.

Fields

`error_messages`: (‘data.frame()‘): Active binding. A dataframe with information about error messages.

`ids`: (‘character()‘): Active binding. A character vector with the IDs of the grouping variable.

`features`: (‘character()‘): Active binding. A character vector with the feature functions which were added.

`status`: (‘data.frame()‘): Active binding. A dataframe with an overview over which features are calculated on which datasets.

`results`: (‘data.frame()‘): Active binding. A dataframe with all calculated features of all IDs.

Methods

`add_data(data, group_by)` [data: (‘data.frame‘ | ‘data.table‘)] A dataframe or data.table which shall be added to the R6 object.
[group_by: (‘character(1)‘)] The grouping variable’s name of the dataframe.

This method writes single RDS files (this can be parallelized with future)

`preprocess_data(fun)` [fun: (‘function‘)] A function, which has a dataframe as input and a dataframe as output.

This method loads the RDS files and applies this function on them. The old RDS files are overwritten.

`remove_data(ids)` [ids: (‘character()‘)] One or many IDs of the grouping variable.

This method deletes the RDS files of the given IDs.

`get_data(ids)` [ids: (‘character()‘)] One or many IDs of the grouping variable.

This method returns one dataframe with the chosen IDs.

`add_feature(fun, check_fun)` [fun: (‘function‘)] A function, which has a dataframe as input and a named vector or list as output.
[check_fun: (‘logical(1)‘)] The function will be checked if it returns a vector or a list. Defaults to `TRUE`. Disable, if calculation takes too long.

This method adds the feature function to the R6 object. It writes an RDS file of the function which can be retrieved later.

`remove_feature(fun)` [fun: (‘function | character(1)‘)] A function (or the name of the function as character) which shall be removed.

This method removes the function from the object and deletes all corresponding files and results.

`get_feature(fun)` [fun: (‘character(1)‘)] The name of a function as character.

This method reads the RDS file of the function. Useful for debugging after loading an Xtractor.

`calc_features(features, force)` [features: (‘character()‘)] A character vector of the names of the features which shall be calculated. Defaults to all features.
[force: (‘logical(1)‘)] The default action is to skip feature functions, which already have been calculated once. Set to `TRUE` for forcing re-calculation.

This method calculates all features on all datasets.

`retry_failed_features(features)` [features: (‘character()‘)] A character vector of the names of the features which shall be calculated. Defaults to all features.

This method retries calculation of failed features. Useful if calculation failed because of memory problems.

`plot()` [internal] method to print the R6 object.

`clone()` [internal] method to clone the R6 object.

`initialize()` [internal] method to initialize the R6 object.

Examples

```
# one feature function
dir = tempdir()
xtractor = Xtractor$new("xtractor", file.dir = dir)
xtractor$add_data(iris, group_by = "Species")
xtractor$ids
fun = function(data) {
  c(mean_sepal_length = mean(data$Sepal.Length))
}
xtractor$add_feature(fun)
xtractor$features
xtractor$calc_features()
xtractor$results
xtractor$status
xtractor

# failing function on only one ID
fun2 = function(data) {
  if ("setosa" %in% data$Species) stop("my error")
  c(sd_sepal_length = sd(data$Sepal.Length))
}
xtractor$add_feature(fun2)
xtractor$calc_features()
xtractor$results
xtractor$error_messages
xtractor

# remove feature function
xtractor$remove_feature("fun2")
xtractor$results
xtractor

# remove ID
xtractor$remove_data("setosa")
xtractor$results
xtractor$ids
xtractor

# get datasets and functions
fun3 = xtractor$get_feature("fun")
df = xtractor$get_data()
dplyr_wrapper(data = df, group_by = "Species", fun = fun3)
```

Index