Package 'rPandas'

April 28, 2026

Title Translating from R to Python's Pandas Package

Version 0.1.4

Description Provides an R interface to Python's 'pandas' library using non-standard evaluation. Users can write R code (e.g., rp_filter(), rp_select(), rp_mutate()) that is translated into pandas commands and executed via 'reticulate'. Supports chaining, grouping, and 'summarisation', and includes a 'table_name' parameter to generate 'copy-pasteable' Python code. Ideal for leveraging pandas' speed and flexibility within the R ecosystem.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Suggests ggplot2, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Imports reticulate, rlang

VignetteBuilder knitr

NeedsCompilation no

Author Akshat Maurya [aut, cre],
Rihaan Satia [aut],
David Shilane [aut]

Maintainer Akshat Maurya <codingmaster902@gmail.com>

Repository CRAN

Date/Publication 2026-04-28 20:30:09 UTC

Contents

`rPandas-package` *rPandas: A User-Friendly R Interface to Pandas*

Description

This package provides a set of wrapper functions that allow R users to interact with Python's pandas library using familiar R syntax.

Author(s)

Maintainer: Akshat Maurya <codingmaster902@gmail.com>

Authors:

- Rihaan Satia
- David Shilane <david.shilane@columbia.edu>

`rp_calculate` *Apply multiple summary functions to multiple columns*

Description

Applies a list of summary functions to a list of columns, after optionally grouping the data.

Usage

```
rp_calculate(
  .data,
  ...,
  the.functions,
  .by = NULL,
  table_name = NULL,
  return.as = "result"
)
```

Arguments

.data	An R data.frame.
...	Bare column names to summarize (e.g., price, carat).
the.functions	A character vector of R function names (e.g., c("mean", "sd")). Supports "mean", "median", "sd", "var", "min", "max", "sum".
.by	A bare column name or c(col1, col2) to group by.
table_name	An optional character string. If provided, the generated Python code will replace the internal dataframe name with this string (e.g., "diamonds.query(...)"). This is useful for seeing the exact, copy-pasteable Python code. Defaults to NULL (uses "df").
return.as	What to return: "result", "code", or "all".

Value

A data.frame with the summarized and grouped data.

Examples

```
if (reticulate::py_available(initialize = TRUE) &&
    reticulate::py_module_available("pandas")) {

  rp_calculate(
    ggplot2::diamonds,
    price, carat,
    the.functions = c("mean", "sd"),
    .by = cut
  )
}
```

---

rp_check_env *Check for rPandas dependencies and provide diagnostics*

Description

This function checks if the user's system is correctly configured with Python and the pandas library. If dependencies are missing, it stops with a detailed diagnostic report and actionable instructions (only in interactive sessions). In non-interactive contexts (e.g., CRAN checks), it issues a warning and returns FALSE.

Usage

```
rp_check_env()
```

Value

Invisibly returns TRUE if all checks pass, otherwise FALSE.

rp_count *Count rows in a data frame, optionally by groups*

Description

This function returns the number of rows in a data frame. When grouping variables are provided via `.by`, it returns the row counts for each group.

Usage

```
rp_count(.data, .by = NULL, table_name = NULL, return.as = "result")
```

Arguments

`.data`	An R data frame (or tibble) to be processed.
`.by`	Optional grouping variables. Can be one or more unquoted column names (e.g., `cut` or `c(cut, color)`). When provided, counts are computed per group.
`table_name`	An optional character string. If provided, the generated Python code will replace the internal dataframe name with this string (e.g., `"diamonds.query(...)"`). This is useful for seeing the exact, copy-pasteable Python code. Defaults to `NULL` (uses `"df"`).
`return.as`	One of `"result"`, `"code"`, or `"all"`.

Value

A data frame with one column `"n"` (total row count) if `.by = NULL`, or a data frame with the grouping columns and a column `"n"` (per-group counts).

rp_filter *Filter rows using pandas*

Description

Filters a data frame using an R expression translated to pandas.

Usage

```
rp_filter(.data, filter_expression, table_name = NULL, return.as = "result")
```

Arguments

.data	An R data.frame or tibble.
filter_expression	The filtering expression, written in R syntax.
table_name	An optional character string. If provided, the generated Python code will replace the internal dataframe name with this string (e.g., "diamonds.query(...)"). This is useful for seeing the exact, copy-pasteable Python code. Defaults to NULL (uses "df").
return.as	What to return: "result", "code", or "all".

Value

A data.frame containing the filtered rows.

Examples

```
if (reticulate::py_available(initialize = TRUE) &&
    reticulate::py_module_available("pandas")) {
      rp_filter(ggplot2::diamonds, carat > 1 & price < 4000)
}
```

---

rp_first_k_rows *Extract the first k rows of a data frame*

Description

This function returns the first k rows of the data frame. If grouping variables are provided via .by, it returns the first k rows within each group.

Usage

```
rp_first_k_rows(.data, k, .by = NULL, table_name = NULL, return.as = "result")
```

Arguments

.data	An R data frame (or tibble) to be processed.
k	An integer specifying the number of rows to return. If .by is used, returns up to k rows per group.
.by	Optional grouping variables. Can be one or more unquoted column names. When provided, the operation is performed on each group separately.
table_name	An optional character string. If provided, the generated Python code will replace the internal dataframe name with this string (e.g., "diamonds.query(...)"). This is useful for seeing the exact, copy-pasteable Python code. Defaults to NULL (uses "df").
return.as	One of "result", "code", or "all".

Value

Depending on `return.as`: a data frame, a character string, or a list.

`rp_last_k_rows` *Extract the last k rows of a data frame*

Description

This function returns the last k rows of the data frame. If grouping variables are provided via `.by`, it returns the last k rows within each group.

Usage

```
rp_last_k_rows(.data, k, .by = NULL, table_name = NULL, return.as = "result")
```

Arguments

`.data`	An R data frame (or tibble) to be processed.
`k`	An integer specifying the number of rows to return. If `.by` is used, returns up to k rows per group.
`.by`	Optional grouping variables. Can be one or more unquoted column names. When provided, the operation is performed on each group separately.
`table_name`	An optional character string. If provided, the generated Python code will replace the internal dataframe name with this string (e.g., `"diamonds.query(...)"`). This is useful for seeing the exact, copy-pasteable Python code. Defaults to `NULL` (uses `"df"`).
`return.as`	One of `"result"`, `"code"`, or `"all"`.

Value

Depending on `return.as`: a data frame, a character string, or a list.

`rp_mutate` *Mutate (add/modify/remove) columns using pandas*

Description

Mutate (add/modify/remove) columns using pandas

Usage

```
rp_mutate(
  .data,
  to_remove = NULL,
  ...,
  table_name = NULL,
  return.as = "result"
)
```

Arguments

.data	An R data frame.
to_remove	A character vector of column names to remove.
...	Named expressions for new/modified columns.
table_name	An optional character string. If provided, the generated Python code will replace the internal dataframe name with this string (e.g., "diamonds.query(...)"). This is useful for seeing the exact, copy-pasteable Python code. Defaults to NULL (uses "df").
return.as	Either "result", "code", or "all".

Value

A data frame or list depending on return.as.

---

rp_select *Filtering columns*

Description

Selects specific columns from a data frame. It captures the bare column names and translates the operation into a pandas selection command.

Usage

```
rp_select(.data, ..., table_name = NULL, return.as = "result")
```

Arguments

.data	An R data.frame or tibble.
...	The bare column names to select (e.g., carat, cut, price).
table_name	An optional character string. If provided, the generated Python code will replace the internal dataframe name with this string (e.g., "diamonds.query(...)"). This is useful for seeing the exact, copy-pasteable Python code. Defaults to NULL (uses "df").
return.as	What to return: "result", "code", or "all".

Value

A `data.frame` containing only the selected columns.

Examples

```
if (reticulate::py_available(initialize = TRUE) &&
    reticulate::py_module_available("pandas")) {
  rp_select(ggplot2::diamonds, carat, cut, price)
}
```

`rp_sort` *Sort rows of a data frame using pandas*

Description

Sorts a data frame by one or more columns. It translates the R expressions into a pandas `.sort_values()` command and executes it.

Usage

```
rp_sort(.data, ..., table_name = NULL, return.as = "result")
```

Arguments

`.data`	An R data.frame or tibble.
`...`	Bare column names to sort by. Use `desc(colname)` to sort in descending order (e.g., `cut, desc(price)`).
`table_name`	An optional character string. If provided, the generated Python code will replace the internal dataframe name with this string (e.g., `"diamonds.query(...)"`). This is useful for seeing the exact, copy-pasteable Python code. Defaults to `NULL` (uses `"df"`).
`return.as`	What to return: "result", "code", or "all".

Value

A `data.frame` sorted by the specified columns.

Examples

```
if (reticulate::py_available(initialize = TRUE) &&
    reticulate::py_module_available("pandas")) {

  # Sort by cut (ascending) and price (descending)
  rp_sort(ggplot2::diamonds, cut, desc(price))
}
```

rp_summarize *Summarize data using pandas*

Description

Aggregates a data frame by one or more groups, applying summary functions. It translates R's `dplyr::summarise` syntax into a pandas `.groupby().agg()` command.

Usage

```
rp_summarize(.data, ..., .by = NULL, table_name = NULL, return.as = "result")
```

Arguments

`.data`	An R data.frame or tibble.
`...`	Named summary expressions (e.g., `avg_price = mean(price)`). Supports `mean`, `median`, `sd`, `var`, `min`, `max`, `sum`, and `n()`.
`.by`	A bare column name or `c(col1, col2)` to group by.
`table_name`	An optional character string. If provided, the generated Python code will replace the internal dataframe name with this string (e.g., `"diamonds.query(...)"`). This is useful for seeing the exact, copy-pasteable Python code. Defaults to `NULL` (uses `"df"`).
`return.as`	What to return: "result", "code", or "all".

Value

A `data.frame` with the summarized and grouped data.

Examples

```
if (reticulate::py_available(initialize = TRUE) &&
    reticulate::py_module_available("pandas")) {

  # Summarize by one group
  rp_summarize(ggplot2::diamonds,
               avg_price = mean(price),
               .by = cut)

  # Summarize by multiple groups and multiple functions
  rp_summarize(ggplot2::diamonds,
               avg_price = mean(price),
               count = n(),
               .by = c(cut, color))
}
```

translate_filter *Translate an R filter expression into a Python query string*

Description

Capture a bare R expression and translate it to a Python-compatible string suitable for use with `pandas.DataFrame.query()`.

Usage

```
translate_filter(expr)
```

Arguments

`expr`	A bare R expression (e.g., `carat > 2 & cut == "Ideal"`).

Value

A character string of the translated Python query.

Examples

```
translate_filter(carat > 2 & cut == "Ideal")
# -> "(carat > 1) and (cut == 'Ideal')"
```

%notin% *"Not In" Operator*

Description

Provides the opposite of the standard R `%in%` operator.

Usage

```
x %notin% y
```

Arguments

`x`	Vector of values to be matched.
`y`	Vector of values to be matched against.

Value

A logical vector.

Examples

```
"a" %notin% c("b", "c")
```

Index