

Package: rPandas (via r-universe)

May 29, 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

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

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

Config/testthat/edition 3

Imports reticulate, rlang

VignetteBuilder knitr

Config/pak/sysreqs libpng-dev python3

Repository <https://akshat09867.r-universe.dev>

Date/Publication 2026-04-29 04:14:29 UTC

RemoteUrl <https://github.com/akshat09867/rpandas>

RemoteRef HEAD

RemoteSha a9db7f35da9b2367450dd0ec4c57dea973666c80

Contents

rPandas-package	2
%notin%	2
rp_calculate	3

rp_check_env	4
rp_count	4
rp_filter	5
rp_first_k_rows	6
rp_last_k_rows	6
rp_mutate	7
rp_select	8
rp_sort	8
rp_summarize	9
translate_filter	10

Index	11
--------------	-----------

rPandas-package	<i>rPandas: A User-Friendly R Interface to Pandas</i>
-----------------	---

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>

%notin%	<i>"Not In" Operator</i>
---------	--------------------------

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")
```

 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

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

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

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

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

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

Value

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

<code>rp_mutate</code>	<i>Mutate (add/modify/remove) columns using pandas</i>
------------------------	--

Description

Mutate (add/modify/remove) columns using pandas

Usage

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

Arguments

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

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

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

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))
}
```

<code>rp_summarize</code>	<i>Summarize data using pandas</i>
---------------------------	------------------------------------

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

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

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')"
```

Index

* **package**

rPandas-package, [2](#)
%notin%, [2](#)

rp_calculate, [3](#)
rp_check_env, [4](#)
rp_count, [4](#)
rp_filter, [5](#)
rp_first_k_rows, [6](#)
rp_last_k_rows, [6](#)
rp_mutate, [7](#)
rp_select, [8](#)
rp_sort, [8](#)
rp_summarize, [9](#)
rPandas (rPandas-package), [2](#)
rPandas-package, [2](#)

translate_filter, [10](#)