

Package ‘sparkhail’

July 23, 2025

Type Package

Title A 'Sparklyr' Extension for 'Hail'

Version 0.1.1

Maintainer Samuel Macêdo <samuelmacedo@recife.ifpe.edu.br>

Description 'Hail' is an open-source, general-purpose, 'python' based data analysis tool with additional data types and methods for working with genomic data, see <<https://hail.is/>>. 'Hail' is built to scale and has first-class support for multi-dimensional structured data, like the genomic data in a genome-wide association study (GWAS). 'Hail' is exposed as a 'python' library, using primitives for distributed queries and linear algebra implemented in 'scala', 'spark', and increasingly 'C++'. The 'sparkhail' is an R extension using 'sparklyr' package. The idea is to help R users to use 'hail' functionalities with the well-know 'tidyverse' syntax, see <<https://www.tidyverse.org/>>.

License Apache License 2.0 | file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.1.2)

Imports dplyr, sparklyr (>= 1.0.1), sparklyr.nested, utils

RoxygenNote 6.1.1

Suggests testthat

NeedsCompilation no

Author Samuel Macêdo [aut, cre],
Javier Luraschi [aut],
Michael Lawrence [ctb]

Repository CRAN

Date/Publication 2019-12-23 17:50:02 UTC

Contents

hail_config	2
hail_context	2

hail_dataframe	3
hail_describe	4
hail_entries	5
hail_get_1kg	5
hail_ids	6
hail_install	7
hail_read_matrix	7
Index	9

hail_config	<i>Read Hail Configuration</i>
-------------	--------------------------------

Description

Set configuration for Hail using spark_config().

Usage

```
hail_config(config = sparklyr::spark_config())
```

Arguments

config A spark configuration.

hail_context	<i>Create Hail Context</i>
--------------	----------------------------

Description

Import and initialize Hail using a spark connection.

Usage

```
hail_context(sc)
```

Arguments

sc Spark connection.

Value

hailContext

Examples

```
library(sparklyr)

sc <- spark_connect(master = "spark://HOST:PORT", config = hail_config())
connection_is_open(sc)
hail_context(sc)
spark_disconnect(sc)
```

hail_dataframe	<i>Create a Dataframe</i>
----------------	---------------------------

Description

This function converts a hail MatrixTable in a dataframe.

Usage

```
hail_dataframe(x)
```

Arguments

x	a hail MatrixTable
---	--------------------

Value

A spark dataframe

Examples

```
## Not run:
library(sparklyr)

sc <- spark_connect(master = "local", version = "2.4", config = hail_config())

hl <- hail_context(sc)
mt <- hail_read_matrix(hl, system.file("extdata/1kg.mt", package = "sparkhail"))

df <- hail_dataframe(mt)
df

## End(Not run)
```

hail_describe	<i>Describe a MatrixTable</i>
---------------	-------------------------------

Description

hail_describe prints a hail MatrixTable structure. You can access parts of the structure using mt_globals_fields, mt_str_rows, mt_col_fields, mt_entry_fields, mt_row_key, mt_col_key.

Usage

```
hail_describe(mt)

mt_globals_fields(mt)

mt_str_rows(mt)

mt_row_fields(mt)

mt_col_fields(mt)

mt_entry_fields(mt)

mt_row_key(mt)

mt_col_key(mt)
```

Arguments

mt	A MatrixTable object.
----	-----------------------

Examples

```
## Not run:
library(sparklyr)

sc <- spark_connect(master = "local", version = "2.4", config = hail_config())

hl <- hail_context(sc)
mt <- hail_read_matrix(hl, system.file("extdata/1kg.mt", package = "sparkhail"))

hail_describe(mt)

## End(Not run)
```

hail_entries*Get Entries Field*

Description

This function retrieves the entries fields from a hail dataframe and explodes the columns call, dp and gq.

Usage

```
hail_entries(df)
```

Arguments

df A hail dataframe.

Value

A spark dataframe.

Examples

```
## Not run:
library(sparklyr)

sc <- spark_connect(master = "local", version = "2.4", config = hail_config())

hail_context(sc) %>%
  hail_read_matrix(system.file("extdata/1kg.mt", package = "sparkhail")) %>%
  hail_dataframe() %>%
  hail_entries()

## End(Not run)
```

hail_get_1kg*Download the Dataset Examples*

Description

This function creates an extdata folder and downloads the datasets necessary to run the examples: 1kg MatrixTable folder and annotations.txt.

Usage

```
hail_get_1kg(path = NULL)
```

Arguments

path	The folder that the user wants to download the data. The path is NULL the data will be downloaded in a temp folder.
------	---

hail_ids

<i>Get Sample Ids</i>

Description

Get the ids from s col key in a MatrixTable.

Usage

```
hail_ids(mt)
```

Arguments

mt	A MatrixTable object.
----	-----------------------

Value

A spark dataframe

Examples

```
## Not run:
library(sparklyr)

hl <- hail_context(sc)
mt <- hail_read_matrix(hl, system.file("extdata/1kg.mt", package = "sparkhail"))

hail_ids(mt)

## End(Not run)
```

hail_install	<i>Install Hail and Datasets</i>
--------------	----------------------------------

Description

Install hail dependencies and datasets to run the examples in documentation. To remove hail use `hail_uninstall`.

Usage

```
hail_install(datasets_examples = TRUE, hail_path = "java_folder")
```

```
hail_uninstall()
```

Arguments

`datasets_examples`

If TRUE, hail will be downloaded along with the datasets to run the examples. Use FALSE if you just want to install hail.

`hail_path`

A string with the path of the jar. Sparklyr extensions normally install the jars in the java folder, but you can select a different one.

hail_read_matrix	<i>Read a MatrixTable</i>
------------------	---------------------------

Description

Read and create a MatrixTable object, it is necessary to convert the data in dataframe using [hail_dataframe](#).

Usage

```
hail_read_matrix(hl, path)
```

Arguments

`hl`

A hail context object. Create one using `hail_context()`.

`path`

A string with the path to MatrixTable folder

Details

A hail **MatrixTable** is a standard data structure in hail framework. A MatrixTable consists of four components:

- a two-dimensional matrix of entry fields where each entry is indexed by row key(s) and column key(s)
- a corresponding rows table that stores all of the row fields that are constant for every column in the dataset
- a corresponding columns table that stores all of the column fields that are constant for every row in the dataset
- a set of global fields that are constant for every entry in the dataset

You can see the MatrixTable structure using [hail_describe](#).

Value

hail_matrix_table

Index

`hail_config`, [2](#)
`hail_context`, [2](#)
`hail_dataframe`, [3](#), [7](#)
`hail_describe`, [4](#), [8](#)
`hail_entries`, [5](#)
`hail_get_1kg`, [5](#)
`hail_ids`, [6](#)
`hail_install`, [7](#)
`hail_read_matrix`, [7](#)
`hail_uninstall` (`hail_install`), [7](#)

`mt_col_fields` (`hail_describe`), [4](#)
`mt_col_key` (`hail_describe`), [4](#)
`mt_entry_fields` (`hail_describe`), [4](#)
`mt_globals_fields` (`hail_describe`), [4](#)
`mt_row_fields` (`hail_describe`), [4](#)
`mt_row_key` (`hail_describe`), [4](#)
`mt_str_rows` (`hail_describe`), [4](#)