Package ‘h2oplyr’

November 12, 2026

Type Package

Title Tidy Data Manipulation on Remote 'H2O' Frames

Version 0.3.0

Description Provides selection, filtering, mutation, aggregation, joins, reshaping and window operations on remote 'H2O' frames. Data manipulation is delegated to 'H2O'; no local table backend is used. Includes an English tutorial and reproducible examples using Palmer penguin measurements.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Imports h2o (>= 3.44.0.3)

Suggests knitr, rmarkdown

VignetteBuilder knitr

SystemRequirements Java (>= 8), a running H2O cluster matching the h2o R package version

NeedsCompilation no

Author Carlos Ortega [aut, cre]

Maintainer Carlos Ortega <cof@qualityexcellence.es>

Repository CRAN

Date/Publication 2026-11-12 16:40:02 UTC

Contents

h2oplyr-package 3
all_of 3
any_of 4
as_h2o 5
contains 6
desc 7
ends_with 8
everything 9

h2o_available . . . 9
h2o_tbl . . . 10
h_across . . . 11
h_anti_join . . . 12
h_arrange . . . 13
h_between . . . 14
h_bind_cols . . . 15
h_bind_rows . . . 16
h_case_when . . . 17
h_coalesce . . . 18
h_collect . . . 19
h_count . . . 20
h_cummax . . . 21
h_cummin . . . 22
h_cumsum . . . 23
h_distinct . . . 24
h_drop_na . . . 25
h_explain . . . 26
h_fill . . . 27
h_filter . . . 28
h_filter_out . . . 29
h_full_join . . . 30
h_glimpse . . . 31
h_group_by . . . 32
h_group_keys . . . 33
h_group_vars . . . 34
h_if_else . . . 35
h_inner_join . . . 36
h_lag . . . 37
h_lead . . . 38
h_left_join . . . 39
h_min_rank . . . 40
h_mutate . . . 41
h_mutate_across . . . 42
h_na_if . . . 43
h_nest_by . . . 44
h_penguins . . . 45
h_pick . . . 46
h_pivot_longer . . . 47
h_pivot_wider . . . 48
h_relocate . . . 49
h_rename . . . 50
h_replace_na . . . 51
h_replace_na_cols . . . 52
h_right_join . . . 53
h_roll_mean . . . 54
h_row_number . . . 55
h_select . . . 56

h_semi_join . . . 57
h_separate . . . 58
h_slice_head . . . 59
h_slice_max . . . 60
h_slice_min . . . 61
h_slice_sample . . . 62
h_slice_tail . . . 63
h_summarise . . . 64
h_summarize . . . 65
h_summarize_across . . . 66
h_transmute . . . 67
h_ungroup . . . 68
h_unite . . . 69
last_col . . . 70
matches . . . 71
num_range . . . 72
starts_with . . . 73
when_all . . . 74
when_any . . . 75
where . . . 76
%.% . . . 77

Index **78**

`h2oplyr-package` *Remote H2O Data Manipulation*

Description

Tidy-style manipulation of H2O frames using H2O as the only data backend. Start a matching H2O cluster explicitly, then use h_penguins() for the included tutorial data.

See Also

`h_penguins`, `h_summarize`. Run `vignette("h2oplyr-penguins")` for the English tutorial.

`all_of` *H2O all of*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
all_of(x)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.

Value

A column-selection specification containing metadata only.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      cols(h_select(x, all_of(c("id", "grp"))), c("id", "grp"))
  }
}
```

---

any_of *H2O any of*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
any_of(x)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.

Value

A column-selection specification containing metadata only.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      cols(h_select(x, any_of(c("id", "absent"))), "id")
  }
}
```

---

as_h2o *Access the remote H2O frame*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
as_h2o(x)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      stopifnot(inherits(as_h2o(x), "H2OFrame"))
  }
}
```

`contains` *H2O contains*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
contains(x)
```

Arguments

`x`	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.

Value

A column-selection specification containing metadata only.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
```

```
                "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      cols(h_select(p, contains("bill")), c("bill_length_mm", "bill_depth_mm"))
  }
}
```

---

desc *H2O desc*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
desc(x)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.

Value

Its input, unchanged; interpreted specially by h_arrange and h_min_rank.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
```

```
    z <- h_arrange(x, desc(y))
    equal_sum(as_h2o(h_slice_head(z, 1))[, "y"], 60)
  }
}
```

`ends_with` *H2O ends with*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
ends_with(x)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.

Value

A column-selection specification containing metadata only.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      cols(h_select(x, ends_with("1")), "x1")
  }
}
```

everything *H2O everything*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
everything()
```

Value

A column-selection specification containing metadata only.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      cols(h_select(x, everything()), names(as_h2o(x)))
  }
}
```

h2o_available *Check an existing H2O connection*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h2o_available()
```

Value

A logical scalar.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      stopifnot(h2o_available())
  }
}
```

---

h2o_tbl *Wrap an existing remote H2O frame*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h2o_tbl(x)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h2o_tbl(as_h2o(x)), 6)
      stopifnot(all(dim(x) == c(6, 7)), nrow(x) == 6, ncol(x) == 7, identical(names(x),
          names(as_h2o(x))))
  }
}
```

`h_across` *H2O across*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. General mutation is ungrouped. h_mutate additionally supports grouped window calls through h_group_by or .by (see below). Named expressions are evaluated sequentially. The closed expression interpreter accepts arithmetic, comparisons, is.na, membership in literal vectors, supported H2O math/text conversions and h2oplyr vector helpers. Unknown functions and other package namespaces are rejected. One-argument across closures are interpreted, never executed over downloaded rows.

Usage

```
h_across(.cols, .fns, .names = "{col}_{fn}")
```

Arguments

`.cols`	A column selection or character vector of column names.
`.fns`	A function or named list. Summary functions must be supported aggregates; mutation functions must have one argument and a supported expression body.
`.names`	Output naming template using col and fn tokens enclosed in braces.

Value

An across specification interpreted by mutation or summary verbs.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      z <- h_mutate(x, h_across(c("x1", "x2"), list(double = function(z) z * 2)))
      equal_sum(as_h2o(z)[, "x1_double"], 114)
  }
}
```

`h_anti_join` *H2O anti join*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Equijoins use H2O merge. Missing-key behavior follows H2O, not SQL NULL semantics. Output row order is unspecified; sort explicitly. Full joins combine a left join and unmatched right rows. Character keys are converted to categorical keys remotely. Suffix collisions and violated relationship constraints raise errors.

Usage

```
h_anti_join(x, y, by = NULL)
```

Arguments

`x`	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.
`y`	A second remote table for joins, or a scalar comparison value for h_na_if.
`by`	Join column names; a named character vector maps left names to right names. For h_count, an optional grouping selection.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_anti_join(h_slice_head(x, 3), h_slice_tail(x, 4), by = "id"), 2)
  }
}
```

`h_arrange` *H2O arrange*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_arrange(.data, ..., na_last = TRUE)
```

Arguments

`.data`	An h2o_tbl. Its frame remains on the H2O cluster.
`...`	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.
`na_last`	Whether missing sort keys follow nonmissing values.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      z <- h_arrange(x, grp, desc(y))
      equal_sum(as_h2o(h_slice_head(z, 1))[, "y"], 20)
  }
}
```

---

h_between *H2O between*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_between(x, left, right)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.
left	Inclusive lower bound.
right	Inclusive upper bound.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(h_between(v, 20, 40), 3)
  }
}
```

`h_bind_cols` *H2O bind cols*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_bind_cols(...)
```

Arguments

`...`	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
```

```
                "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      cols(h_bind_cols(h_select(x, id), h_select(x, y)), c("id", "y"))
  }
}
```

---

h_bind_rows *H2O bind rows*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Inputs must have the same column-name set and compatible H2O types; column order is aligned. Missing-column union is not implemented.

Usage

```
h_bind_rows(...)
```

Arguments

...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
```

```
  {
      rows(h_bind_rows(x, x), 12)
  }
}
```

---

h_case_when *H2O case when*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_case_when(..., .default = NA)
```

Arguments

`...`	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.
`.default`	Default scalar or compatible H2O column when no condition matches.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      z <- h_mutate(x, label = h_case_when(y >= 40 ~ 2, TRUE ~ 1))
      equal_sum(as_h2o(z)[, "label"], 9)
  }
}
```

h_coalesce *H2O coalesce*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_coalesce(...)
```

Arguments

...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(h_coalesce(as_h2o(x)[, "x"], 0), 16)
  }
}
```

h_collect *Return the remote frame without downloading*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. This intentionally differs from the standalone prototype: it returns an H2OFrame, never a local table.

Usage

```
h_collect(.data)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      stopifnot(inherits(h_collect(x), "H2OFrame"))
  }
}
```

h_count *H2O count*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_count(.data, ..., wt = NULL, sort = FALSE, name = "n", by = NULL)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.
wt	Optional weight column, captured as a name. Missing weights are removed.
sort	Order counts from largest to smallest.
name	Output count-column name.
by	Join column names; a named character vector maps left names to right names. For h_count, an optional grouping selection.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      z <- h_count(x, grp)
      rows(z, 3)
      equal_sum(as_h2o(z)[, "n"], 6)
  }
}
```

h_cummax *H2O cummax*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Operations use current frame order. Arrange explicitly before windows. Grouped slicing and filling are rejected. Sampling is performed with H2O random columns; replacement uses a remote draw-index join.

Usage

```
h_cummax(x)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(h_cummax(v), 210)
  }
}
```

h_cummin *H2O cummin*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Operations use current frame order. Arrange explicitly before windows. Grouped slicing and filling are rejected. Sampling is performed with H2O random columns; replacement uses a remote draw-index join.

Usage

```
h_cummin(x)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(h_cummin(v), 60)
  }
}
```

h_cumsum *H2O cumsum*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Operations use current frame order. Arrange explicitly before windows. Grouped slicing and filling are rejected. Sampling is performed with H2O random columns; replacement uses a remote draw-index join.

Usage

```
h_cumsum(x)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(h_cumsum(v), 560)
  }
}
```

`h_distinct` *Deduplicate remote rows*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_distinct(.data, ..., .keep_all = FALSE)
```

Arguments

`.data`	An h2o_tbl. Its frame remains on the H2O cluster.
`...`	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.
`.keep_all`	Keep every column, retaining the first row for each distinct key.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_distinct(x, grp), 3)
  }
}
```

h_drop_na *H2O drop na*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_drop_na(.data, ...)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_drop_na(p), 333)
  }
}
```

h_explain *H2O explain*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Displays dimensions, column metadata and grouping only. No row preview or cost estimate is downloaded.

Usage

```
h_explain(.data)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      stopifnot(identical(h_explain(x), x))
  }
}
```

h_fill *Fill missing values in row order*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Operations use current frame order. Arrange explicitly before windows. Grouped slicing and filling are rejected. Sampling is performed with H2O random columns; replacement uses a remote draw-index join.

Usage

```
h_fill(.data, ..., .direction = c("down", "up"))
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.
.direction	Fill down from the preceding value, or up from the following value.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(as_h2o(h_fill(x, all_of("x")))[, "x"], 20)
  }
}
```

h_filter *Keep rows satisfying remote predicates*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_filter(.data, ...)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_filter(x, x >= 3), 3)
  }
}
```

h_filter_out *Exclude rows satisfying remote predicates*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_filter_out(.data, ...)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_filter_out(x, x >= 3), 2)
  }
}
```

h_full_join *H2O full join*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Equijoins use H2O merge. Missing-key behavior follows H2O, not SQL NULL semantics. Output row order is unspecified; sort explicitly. Full joins combine a left join and unmatched right rows. Character keys are converted to categorical keys remotely. Suffix collisions and violated relationship constraints raise errors.

Usage

```
h_full_join(x, y, by = NULL, suffix = c(".x", ".y"), relationship = NULL)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.
y	A second remote table for joins, or a scalar comparison value for h_na_if.
by	Join column names; a named character vector maps left names to right names. For h_count, an optional grouping selection.
suffix	Two suffixes for overlapping non-key column names.
relationship	Optional one-to-one, one-to-many, many-to-one or many-to-many key constraint.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_full_join(h_slice_head(x, 3), h_slice_tail(x, 4), by = "id"), 6)
  }
}
```

h_glimpse *H2O glimpse*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Displays dimensions, column metadata and grouping only. No row preview or cost estimate is downloaded.

Usage

```
h_glimpse(.data, width = 60)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
width	Optional zero-padding width for num_range; retained metadata-display width for h_glimpse.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      stopifnot(identical(h_glimpse(x), x))
  }
}
```

h_group_by *H2O group by*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_group_by(.data, ..., .add = FALSE)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.
.add	Add columns to existing grouping metadata rather than replace it.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      stopifnot(identical(h_group_by(x, grp)$groups, "grp"))
  }
}
```

h_group_keys *H2O group keys*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. For an ungrouped input, returns one remote cell containing its row count.

Usage

```
h_group_keys(.data)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_group_keys(h_group_by(x, grp)), 3)
  }
}
```

h_group_vars *H2O group vars*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_group_vars(.data)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.

Value

A character vector of grouping column names.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      stopifnot(identical(h_group_vars(h_group_by(x, grp)), "grp"))
  }
}
```

h_if_else *H2O if else*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_if_else(condition, true_val, false_val, missing = NA)
```

Arguments

condition	A one-column H2OFrame of conditions.
true_val	Scalar or compatible H2O column for true conditions.
false_val	Scalar or compatible H2O column for false conditions.
missing	Value for missing conditions.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(h_if_else(v >= 40, 1, 0), 3)
  }
}
```

h_inner_join *H2O inner join*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Equijoins use H2O merge. Missing-key behavior follows H2O, not SQL NULL semantics. Output row order is unspecified; sort explicitly. Full joins combine a left join and unmatched right rows. Character keys are converted to categorical keys remotely. Suffix collisions and violated relationship constraints raise errors.

Usage

```
h_inner_join(x, y, by = NULL, suffix = c(".x", ".y"), relationship = NULL)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.
y	A second remote table for joins, or a scalar comparison value for h_na_if.
by	Join column names; a named character vector maps left names to right names. For h_count, an optional grouping selection.
suffix	Two suffixes for overlapping non-key column names.
relationship	Optional one-to-one, one-to-many, many-to-one or many-to-many key constraint.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_inner_join(h_slice_head(x, 3), h_slice_tail(x, 4), by = "id"), 1)
  }
}
```

h_lag *H2O lag*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Within grouped h_mutate, each expression must be a direct h_lag, h_lead, h_roll_mean, h_min_rank or h_row_number call. Window inputs must be column names (or desc(column) for h_min_rank). Group keys cannot be overwritten. Missing grouping keys form groups. Windows follow the input order within each group; output rows retain the original input order. Arrange by a complete ordering key first for reproducible windows. Other grouped expressions, including cumulative helpers, are rejected. h_transmute and h_mutate_across remain ungrouped. Operations use current frame order. Arrange explicitly before windows. Grouped slicing and filling are rejected. Sampling is performed with H2O random columns; replacement uses a remote draw-index join.

Usage

```
h_lag(x, n = 1, default = NA)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.
n	A nonnegative integer row count or shift distance.
default	Padding value for lag or lead.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
```

```
      equal_sum(h_lag(v, default = 0), 150)
  }
}
```

---

h_lead *H2O lead*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Within grouped h_mutate, each expression must be a direct h_lag, h_lead, h_roll_mean, h_min_rank or h_row_number call. Window inputs must be column names (or desc(column) for h_min_rank). Group keys cannot be overwritten. Missing grouping keys form groups. Windows follow the input order within each group; output rows retain the original input order. Arrange by a complete ordering key first for reproducible windows. Other grouped expressions, including cumulative helpers, are rejected. h_transmute and h_mutate_across remain ungrouped. Operations use current frame order. Arrange explicitly before windows. Grouped slicing and filling are rejected. Sampling is performed with H2O random columns; replacement uses a remote draw-index join.

Usage

```
h_lead(x, n = 1, default = NA)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.
n	A nonnegative integer row count or shift distance.
default	Padding value for lag or lead.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
```

```
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(h_lead(v, default = 0), 200)
  }
}
```

`h_left_join` *H2O left join*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Equijoins use H2O merge. Missing-key behavior follows H2O, not SQL NULL semantics. Output row order is unspecified; sort explicitly. Full joins combine a left join and unmatched right rows. Character keys are converted to categorical keys remotely. Suffix collisions and violated relationship constraints raise errors.

Usage

```
h_left_join(x, y, by = NULL, suffix = c(".x", ".y"), relationship = NULL)
```

Arguments

`x`	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.
`y`	A second remote table for joins, or a scalar comparison value for h_na_if.
`by`	Join column names; a named character vector maps left names to right names. For h_count, an optional grouping selection.
`suffix`	Two suffixes for overlapping non-key column names.
`relationship`	Optional one-to-one, one-to-many, many-to-one or many-to-many key constraint.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
```

```
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_left_join(h_slice_head(x, 3), h_slice_tail(x, 4), by = "id"), 3)
  }
}
```

`h_min_rank` *Rank numeric values with minimum ranks for ties*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Within grouped h_mutate, each expression must be a direct h_lag, h_lead, h_roll_mean, h_min_rank or h_row_number call. Window inputs must be column names (or desc(column) for h_min_rank). Group keys cannot be overwritten. Missing grouping keys form groups. Windows follow the input order within each group; output rows retain the original input order. Arrange by a complete ordering key first for reproducible windows. Other grouped expressions, including cumulative helpers, are rejected. h_transmute and h_mutate_across remain ungrouped. Only numeric columns are supported. Ties receive the smallest position: 1, 2, 2, 4. Missing values receive NA and do not consume ranks. desc(x) reverses the ranking direction. Ranks are computed using remote sorting, aggregation and joins.

Usage

```
h_min_rank(x)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
```

```
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(h_min_rank(desc(v)), 21)
      z <- h_mutate(x, r = h_min_rank(desc(y)), .by = grp)
      equal_sum(as_h2o(z)[, "r"], 9)
  }
}
```

`h_mutate` *Create or replace columns remotely*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. General mutation is ungrouped. h_mutate additionally supports grouped window calls through h_group_by or .by (see below). Named expressions are evaluated sequentially. The closed expression interpreter accepts arithmetic, comparisons, is.na, membership in literal vectors, supported H2O math/text conversions and h2oplyr vector helpers. Unknown functions and other package namespaces are rejected. One-argument across closures are interpreted, never executed over downloaded rows. Within grouped h_mutate, each expression must be a direct h_lag, h_lead, h_roll_mean, h_min_rank or h_row_number call. Window inputs must be column names (or desc(column) for h_min_rank). Group keys cannot be overwritten. Missing grouping keys form groups. Windows follow the input order within each group; output rows retain the original input order. Arrange by a complete ordering key first for reproducible windows. Other grouped expressions, including cumulative helpers, are rejected. h_transmute and h_mutate_across remain ungrouped.

Usage

```
h_mutate(.data, ..., .by = NULL)
```

Arguments

`.data`	An h2o_tbl. Its frame remains on the H2O cluster.
`...`	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.
`.by`	Per-call grouping selection on ungrouped input. Grouping does not persist in the result.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      z <- h_mutate(x, double = y * 2, next_value = double + 1)
      equal_sum(as_h2o(z)[, "next_value"], 426)
  }
}
```

`h_mutate_across` *H2O mutate across*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. General mutation is ungrouped. h_mutate additionally supports grouped window calls through h_group_by or .by (see below). Named expressions are evaluated sequentially. The closed expression interpreter accepts arithmetic, comparisons, is.na, membership in literal vectors, supported H2O math/text conversions and h2oplyr vector helpers. Unknown functions and other package namespaces are rejected. One-argument across closures are interpreted, never executed over downloaded rows.

Usage

```
h_mutate_across(.data, .cols, .fns, .names = "{col}_{fn}")
```

Arguments

`.data`	An h2o_tbl. Its frame remains on the H2O cluster.
`.cols`	A column selection or character vector of column names.
`.fns`	A function or named list. Summary functions must be supported aggregates; mutation functions must have one argument and a supported expression body.
`.names`	Output naming template using col and fn tokens enclosed in braces.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      z <- h_mutate_across(x, c("x1", "x2"), list(double = function(z) z * 2))
      equal_sum(as_h2o(z)[, "x2_double"], 1140)
  }
}
```

---

h_na_if *H2O na if*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_na_if(x, y)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.
y	A second remote table for joins, or a scalar comparison value for h_na_if.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      stopifnot(h2o::h2o.sum(is.na(h_na_if(v, 10))) == 1)
  }
}
```

`h_nest_by` *Create a list of remote group handles*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Returns a list with keys (h2o_tbl) and data (list of h2o_tbl handles), not a data frame with a list-column. This makes one remote join per group and is intended for modest group counts.

Usage

```
h_nest_by(.data, ...)
```

Arguments

`.data`	An h2o_tbl. Its frame remains on the H2O cluster.
`...`	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.

Value

A list containing remote keys and a list of remote table handles.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      z <- h_nest_by(x, grp)
      stopifnot(length(z$data) == 3)
      invisible(lapply(z$data, rows, n = 2))
  }
}
```

---

h_penguins *Upload the bundled Palmer penguins CSV*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_penguins()
```

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
```

```
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(p, 344)
  }
}
```

---

h_pick *H2O pick*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_pick(.data, ...)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      cols(h_pick(x, starts_with("x")), c("x", "x1", "x2"))
  }
}
```

`h_pivot_longer` *H2O pivot longer*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_pivot_longer(.data, cols, names_to = "name", values_to = "value")
```

Arguments

`.data`	An h2o_tbl. Its frame remains on the H2O cluster.
`cols`	Columns to reshape from wide to long.
`names_to`	Name of the output column containing former column names.
`values_to`	Name of the output value column.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_pivot_longer(h_select(x, id, x1, x2), c("x1", "x2")), 12)
  }
}
```

h_pivot_wider *Widen unique ID and name combinations*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Duplicate ID/name combinations and missing names are rejected; summarize first. Category levels are schema metadata. All output cells are formed by remote joins. values_fill replaces both absent combinations and existing missing cells. No list-columns or implicit aggregation are supported.

Usage

```
h_pivot_wider(.data, names_from, values_from, id_cols = NULL, values_fill = NULL)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
names_from	One column whose category domain becomes output column names.
values_from	One column containing cell values.
id_cols	ID column selection; defaults to all columns other than names_from and values_from.
values_fill	Optional replacement for missing output cells.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      long <- h_pivot_longer(h_select(x, id, x1, x2), c("x1", "x2"))
      wide <- h_pivot_wider(long, name, value)
```

```
      rows(wide, 6)
      equal_sum(as_h2o(wide)[, "x2"], 570)
  }
}
```

`h_relocate` *H2O relocate*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_relocate(.data, ..., .before = NULL, .after = NULL)
```

Arguments

`.data`	An h2o_tbl. Its frame remains on the H2O cluster.
`...`	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.
`.before`	Column selection before which the selected columns are moved.
`.after`	Column selection after which the selected columns are moved.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      stopifnot(names(as_h2o(h_relocate(x, y, .before = id)))[1] == "y")
  }
}
```

h_rename *H2O rename*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_rename(.data, ...)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      cols(h_rename(h_select(x, id), identifier = id), "identifier")
  }
}
```

h_replace_na *H2O replace na*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_replace_na(x, replace)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.
replace	For h_replace_na, a scalar or named list of replacements. For h_slice_sample, whether to sample with replacement.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(as_h2o(h_replace_na(x, list(x = 5)))[, "x"], 21)
  }
}
```

h_replace_na_cols *H2O replace na cols*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_replace_na_cols(.data, replace, ...)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
replace	For h_replace_na, a scalar or named list of replacements. For h_slice_sample, whether to sample with replacement.
...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(as_h2o(h_replace_na_cols(x, 5, all_of("x")))[, "x"], 21)
  }
}
```

h_right_join *H2O right join*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Equijoins use H2O merge. Missing-key behavior follows H2O, not SQL NULL semantics. Output row order is unspecified; sort explicitly. Full joins combine a left join and unmatched right rows. Character keys are converted to categorical keys remotely. Suffix collisions and violated relationship constraints raise errors.

Usage

```
h_right_join(x, y, by = NULL, suffix = c(".x", ".y"), relationship = NULL)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.
y	A second remote table for joins, or a scalar comparison value for h_na_if.
by	Join column names; a named character vector maps left names to right names. For h_count, an optional grouping selection.
suffix	Two suffixes for overlapping non-key column names.
relationship	Optional one-to-one, one-to-many, many-to-one or many-to-many key constraint.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_right_join(h_slice_head(x, 3), h_slice_tail(x, 4), by = "id"), 4)
  }
}
```

h_roll_mean *Compute right-aligned remote rolling means*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Within grouped h_mutate, each expression must be a direct h_lag, h_lead, h_roll_mean, h_min_rank or h_row_number call. Window inputs must be column names (or desc(column) for h_min_rank). Group keys cannot be overwritten. Missing grouping keys form groups. Windows follow the input order within each group; output rows retain the original input order. Arrange by a complete ordering key first for reproducible windows. Other grouped expressions, including cumulative helpers, are rejected. h_transmute and h_mutate_across remain ungrouped. The window is right-aligned and includes the current row. Missing values propagate by default. With na.rm = TRUE, only observed values contribute; all-missing windows return NA. partial = FALSE requires k rows, even when na.rm = TRUE. The implementation combines up to k remote shifts, so large widths can be expensive; it is intended for short windows, not constant-cost rolling computation.

Usage

```
h_roll_mean(x, k = 3, partial = TRUE, na.rm = FALSE)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.
k	A positive integer window width, measured in rows.
partial	Use the available prefix for windows shorter than k. With FALSE, incomplete windows are missing.
na.rm	Remove missing values within each rolling window. An all-missing window returns NA.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
```

```
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(h_roll_mean(v, k = 3), 165)
      z <- h_mutate(x, avg = h_roll_mean(y, k = 3), .by = grp)
      equal_sum(as_h2o(z)[, "avg"], 195)
  }
}
```

`h_row_number` *Number rows within the current mutation group*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Within grouped h_mutate, each expression must be a direct h_lag, h_lead, h_roll_mean, h_min_rank or h_row_number call. Window inputs must be column names (or desc(column) for h_min_rank). Group keys cannot be overwritten. Missing grouping keys form groups. Windows follow the input order within each group; output rows retain the original input order. Arrange by a complete ordering key first for reproducible windows. Other grouped expressions, including cumulative helpers, are rejected. h_transmute and h_mutate_across remain ungrouped. This zero-argument context helper is only valid inside h_mutate. It returns positions 1 through the current group size, or 1 through the frame size on ungrouped input. Calling it directly raises an error.

Usage

```
h_row_number()
```

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
```

```
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      z <- h_mutate(x, i = h_row_number(), .by = grp)
      equal_sum(as_h2o(z)[, "i"], 9)
  }
}
```

`h_select` *Select columns using schema metadata*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_select(.data, ...)
```

Arguments

`.data`	An h2o_tbl. Its frame remains on the H2O cluster.
`...`	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      cols(h_select(x, id, grp), c("id", "grp"))
  }
}
```

h_semi_join *H2O semi join*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Equijoins use H2O merge. Missing-key behavior follows H2O, not SQL NULL semantics. Output row order is unspecified; sort explicitly. Full joins combine a left join and unmatched right rows. Character keys are converted to categorical keys remotely. Suffix collisions and violated relationship constraints raise errors.

Usage

```
h_semi_join(x, y, by = NULL)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.
y	A second remote table for joins, or a scalar comparison value for h_na_if.
by	Join column names; a named character vector maps left names to right names. For h_count, an optional grouping selection.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_semi_join(h_slice_head(x, 3), h_slice_tail(x, 4), by = "id"), 1)
  }
}
```

h_separate *H2O separate*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. H2O splits using a regular expression. The maximum number of split fields must equal length(into); shorter rows are padded by H2O. The source column is removed.

Usage

```
h_separate(.data, col, into, sep = "-")
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
col	The source column for h_separate, or the new output name for h_unite.
into	Names of split output columns.
sep	Regular expression separator for h_separate; literal separator for h_unite (see its restrictions).

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      z <- h_separate(x, txt, c("letter", "number"))
      rows(z, 6)
      stopifnot(all(c("letter", "number") %in% names(as_h2o(z))))
  }
}
```

h_slice_head *H2O slice head*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Operations use current frame order. Arrange explicitly before windows. Grouped slicing and filling are rejected. Sampling is performed with H2O random columns; replacement uses a remote draw-index join.

Usage

```
h_slice_head(.data, n = 6)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
n	A nonnegative integer row count or shift distance.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_slice_head(x, 2), 2)
  }
}
```

h_slice_max *H2O slice max*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Operations use current frame order. Arrange explicitly before windows. Grouped slicing and filling are rejected. Sampling is performed with H2O random columns; replacement uses a remote draw-index join.

Usage

```
h_slice_max(.data, order_by, n = 1, with_ties = TRUE)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
order_by	One unquoted ordering column.
n	A nonnegative integer row count or shift distance.
with_ties	Include all rows tied at the selected cutoff. Missing ordering values are excluded.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(as_h2o(h_slice_max(x, y, n = 2))[, "y"], 110)
  }
}
```

`h_slice_min` *H2O slice min*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Operations use current frame order. Arrange explicitly before windows. Grouped slicing and filling are rejected. Sampling is performed with H2O random columns; replacement uses a remote draw-index join.

Usage

```
h_slice_min(.data, order_by, n = 1, with_ties = TRUE)
```

Arguments

`.data`	An h2o_tbl. Its frame remains on the H2O cluster.
`order_by`	One unquoted ordering column.
`n`	A nonnegative integer row count or shift distance.
`with_ties`	Include all rows tied at the selected cutoff. Missing ordering values are excluded.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(as_h2o(h_slice_min(x, y, n = 2))[, "y"], 30)
  }
}
```

h_slice_sample *H2O slice sample*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Operations use current frame order. Arrange explicitly before windows. Grouped slicing and filling are rejected. Sampling is performed with H2O random columns; replacement uses a remote draw-index join.

Usage

```
h_slice_sample(.data, n = NULL, prop = NULL, replace = FALSE, seed = NULL)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
n	A nonnegative integer row count or shift distance.
prop	Sample fraction. Supply either n or prop.
replace	For h_replace_na, a scalar or named list of replacements. For h_slice_sample, whether to sample with replacement.
seed	Optional H2O random seed. With NULL (the default), h2o.runif() is called without a seed argument so H2O chooses the seed. A supplied seed is forwarded unchanged; reproducibility assumes the same cluster layout and input frame.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_slice_sample(x, n = 3, seed = 42), 3)
```

```
    rows(h_slice_sample(x, n = 9, replace = TRUE, seed = 42), 9)
  }
}
```

`h_slice_tail` *H2O slice tail*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Operations use current frame order. Arrange explicitly before windows. Grouped slicing and filling are rejected. Sampling is performed with H2O random columns; replacement uses a remote draw-index join.

Usage

```
h_slice_tail(.data, n = 6)
```

Arguments

`.data`	An h2o_tbl. Its frame remains on the H2O cluster.
`n`	A nonnegative integer row count or shift distance.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      equal_sum(as_h2o(h_slice_tail(x, 2))[, "y"], 110)
  }
}
```

h_summarise *Compute grouped or global H2O summaries*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Supported reductions: mean, sum, min, max, sd, var, median, n() and n_distinct(). A reduction accepts one column and optional na.rm. Summary results are ungrouped. Empty-input summaries raise an error. Across summaries use the default na.rm = FALSE; drop missing rows explicitly when needed.

Usage

```
h_summarise(.data, ..., .by = NULL)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.
.by	Per-call grouping selection on ungrouped input. Grouping does not persist in the result.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      z <- h_summarise(x, n = n())
      equal_sum(as_h2o(z), 6)
  }
}
```

h_summarize *Compute grouped or global H2O summaries*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Supported reductions: mean, sum, min, max, sd, var, median, n() and n_distinct(). A reduction accepts one column and optional na.rm. Summary results are ungrouped. Empty-input summaries raise an error. Across summaries use the default na.rm = FALSE; drop missing rows explicitly when needed.

Usage

```
h_summarize(.data, ..., .by = NULL)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.
.by	Per-call grouping selection on ungrouped input. Grouping does not persist in the result.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      z <- h_summarize(x, n = n(), avg = mean(y), .by = grp)
      rows(z, 3)
      equal_sum(as_h2o(z)[, "avg"], 105)
  }
}
```

`h_summarize_across` *H2O summarize across*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Supported reductions: mean, sum, min, max, sd, var, median, n() and n_distinct(). A reduction accepts one column and optional na.rm. Summary results are ungrouped. Empty-input summaries raise an error. Across summaries use the default na.rm = FALSE; drop missing rows explicitly when needed.

Usage

```
h_summarize_across(.data, .cols, .fns, .names = "{col}_{fn}")
```

Arguments

`.data`	An h2o_tbl. Its frame remains on the H2O cluster.
`.cols`	A column selection or character vector of column names.
`.fns`	A function or named list. Summary functions must be supported aggregates; mutation functions must have one argument and a supported expression body.
`.names`	Output naming template using col and fn tokens enclosed in braces.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      z <- h_summarize_across(h_group_by(x, grp), "y", list(mean = mean))
      equal_sum(as_h2o(z)[, "y_mean"], 105)
  }
}
```

h_transmute *H2O transmute*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. General mutation is ungrouped. h_mutate additionally supports grouped window calls through h_group_by or .by (see below). Named expressions are evaluated sequentially. The closed expression interpreter accepts arithmetic, comparisons, is.na, membership in literal vectors, supported H2O math/text conversions and h2oplyr vector helpers. Unknown functions and other package namespaces are rejected. One-argument across closures are interpreted, never executed over downloaded rows.

Usage

```
h_transmute(.data, ...)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      z <- h_transmute(x, kg = y/1000)
      cols(z, "kg")
      equal_sum(as_h2o(z), 0.21)
  }
}
```

h_ungroup *H2O ungroup*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
h_ungroup(.data)
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      stopifnot(length(h_group_vars(h_ungroup(h_group_by(x, grp)))) == 0)
  }
}
```

h_unite *Concatenate categorical values on H2O*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed. Uses H2O interaction with underscore encoding. Missing inputs produce missing output. A reserved encoding token is rejected. Separators containing dollar signs or backslashes are rejected because H2O uses Java replacement syntax. Interaction domains can be expensive for high-cardinality inputs.

Usage

```
h_unite(.data, col, ..., sep = "_")
```

Arguments

.data	An h2o_tbl. Its frame remains on the H2O cluster.
col	The source column for h_separate, or the new output name for h_unite.
...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.
sep	Regular expression separator for h_separate; literal separator for h_unite (see its restrictions).

Value

An h2o_tbl holding a remote H2OFrame; inspection helpers return it invisibly.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      z <- h_unite(h_separate(x, txt, c("letter", "number")), "text", letter, number,
          sep = "-")
      rows(h_filter(z, text == "b-2"), 1)
```

```
  }
}
```

`last_col` *H2O last col*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
last_col(offset = 0)
```

Arguments

`offset`	A nonnegative offset from the last column.

Value

A column-selection specification containing metadata only.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      cols(h_select(x, last_col()), "x2")
  }
}
```

`matches` *H2O matches*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
matches(x)
```

Arguments

`x`	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.

Value

A column-selection specification containing metadata only.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      cols(h_select(x, matches("^x[12]$")), c("x1", "x2"))
  }
}
```

num_range *H2O num range*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
num_range(prefix, range, width = NULL)
```

Arguments

prefix	A column-name prefix.
range	Numeric suffixes to match.
width	Optional zero-padding width for num_range; retained metadata-display width for h_glimpse.

Value

A column-selection specification containing metadata only.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      cols(h_select(x, num_range("x", 1:2)), c("x1", "x2"))
  }
}
```

starts_with *H2O starts with*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
starts_with(x)
```

Arguments

x	An H2OFrame, an h2o_tbl, or (for selection helpers) a character pattern or vector. Local tables are not accepted.

Value

A column-selection specification containing metadata only.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      cols(h_select(x, starts_with("x")), c("x", "x1", "x2"))
  }
}
```

when_all *H2O when all*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
when_all(...)
```

Arguments

...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_filter(x, when_all(y >= 20, y <= 40)), 3)
  }
}
```

when_any *H2O when any*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
when_any(...)
```

Arguments

...	Column selections, named expressions, remote tables or conditions, as appropriate to the verb. See the example and details.

Value

An H2OFrame expression (vector helpers return one column).

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(h_filter(x, when_any(y == 10, y == 60)), 2)
  }
}
```

`where` *H2O where*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
where(predicate)
```

Arguments

`predicate`	One of base is.numeric, is.character or is.factor; inspected through H2O schema metadata.

Value

A column-selection specification containing metadata only.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      cols(h_select(x, where(is.numeric)), c("id", "x", "y", "x1", "x2"))
  }
}
```

`%.%` *H2O %.%*

Description

Row data remain in H2O. Base R handles expressions, column names, configuration and remote object handles. H2O may return schema metadata or scalar reductions to the client; no local table transformation is performed.

Usage

```
`%.%`(lhs, rhs)
```

Arguments

`lhs`	The left-hand pipeline value.
`rhs`	A function call receiving lhs as its first argument.

Value

The result of the right-hand function call.

Examples

```
# Connect explicitly before running these examples:
# h2o::h2o.init(nthreads = 2, max_mem_size = "2G")
if (h2o_available()) {
  x <- h2o_tbl(h2o::h2o.uploadFile(
    system.file("extdata", "toy.csv", package = "h2oplyr"),
    col.types = c("numeric", "enum", "numeric", "numeric",
                  "string", "numeric", "numeric")))
  p <- h_penguins()
  v <- as_h2o(x)[, "y"]
  rows <- function(z, n) stopifnot(nrow(as_h2o(z)) == n)
  cols <- function(z, nm) stopifnot(identical(names(as_h2o(z)), nm))
  equal_sum <- function(z, value) stopifnot(isTRUE(all.equal(
    h2o::h2o.sum(z, na.rm = TRUE), value, check.attributes = FALSE)))
  {
      rows(x %.% h_filter(y > 30), 3)
  }
}
```

Index

%.%, 77

all_of, 3
any_of, 4
as_h2o, 5

contains, 6

desc, 7
dim.h2o_tbl *(*h2o_tbl*)*, 10

ends_with, 8
everything, 9

h2o_available, 9
h2o_tbl, 10
h2oplyr *(*h2oplyr-package*)*, 3
h2oplyr-package, 3
h_across, 11
h_anti_join, 12
h_arrange, 13
h_between, 14
h_bind_cols, 15
h_bind_rows, 16
h_case_when, 17
h_coalesce, 18
h_collect, 19
h_count, 20
h_cummax, 21
h_cummin, 22
h_cumsum, 23
h_distinct, 24
h_drop_na, 25
h_explain, 26
h_fill, 27
h_filter, 28
h_filter_out, 29
h_full_join, 30
h_glimpse, 31
h_group_by, 32
h_group_keys, 33
h_group_vars, 34
h_if_else, 35
h_inner_join, 36
h_lag, 37
h_lead, 38
h_left_join, 39
h_min_rank, 40
h_mutate, 41
h_mutate_across, 42
h_na_if, 43
h_nest_by, 44
h_penguins, *3*, 45
h_pick, 46
h_pivot_longer, 47
h_pivot_wider, 48
h_relocate, 49
h_rename, 50
h_replace_na, 51
h_replace_na_cols, 52
h_right_join, 53
h_roll_mean, 54
h_row_number, 55
h_select, 56
h_semi_join, 57
h_separate, 58
h_slice_head, 59
h_slice_max, 60
h_slice_min, 61
h_slice_sample, 62
h_slice_tail, 63
h_summarise, 64
h_summarize, *3*, 65
h_summarize_across, 66
h_transmute, 67
h_ungroup, 68
h_unite, 69

last_col, 70

matches, 71

names.h2o_tbl *(*h2o_tbl*)*, 10
num_range, 72

print.h2o_tbl *(*h2o_tbl*)*, 10

starts_with, 73

when_all, 74
when_any, 75
where, 76