

Package ‘crplyr’

March 18, 2025

Type Package

Title A 'dplyr' Interface for Crunch

Description In order to facilitate analysis of datasets hosted on the Crunch data platform <<https://crunch.io/>>, the 'crplyr' package implements 'dplyr' methods on top of the Crunch backend. The usual methods 'select', 'filter', 'group_by', 'summarize', and 'collect' are implemented in such a way as to perform as much computation on the server and pull as little data locally as possible.

Version 0.4.2

URL <https://crunch.io/r/crplyr/>, <https://github.com/Crunch-io/crplyr>

BugReports <https://github.com/Crunch-io/crplyr/issues>

License LGPL (>= 3)

Depends R (>= 3.0.0), crunch (>= 1.15.3), dplyr

Imports methods, ggplot2, httptest (>= 3.0.0), lazyeval, lifecycle, purrr, tibble, rlang, scales, stringr, tidyselect, viridisLite

Suggests covr, magrittr, spelling, vdiff, knitr, testthat, rmarkdown

RoxygenNote 7.3.2

Language en-US

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation no

Author Greg Freedman Ellis [aut, cre],
Jonathan Keane [aut],
Neal Richardson [aut],
Mike Malecki [aut],
Gordon Shotwell [aut],
Aljaž Sluga [aut]

Maintainer Greg Freedman Ellis <greg@crunch.io>

Repository CRAN

Date/Publication 2025-03-18 12:40:14 UTC

Contents

as_cr_tibble	2
autoplot	3
collect	4
filter	5
filter_CrunchDataset	5
GroupedCrunchDataset-class	6
group_by	6
mutate	7
select	8
summarize	8
theme_crunch	9
unweighted_n	10

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

as_cr_tibble	<i>Flatten a Crunch Cube</i>
--------------	------------------------------

Description

Crunch Cubes can be expressed as a long data frame instead of a multidimensional array. In this form each dimension of the cube is a variable and the cube values are expressed as columns for each measure. This is useful both to better understand what each entry of a cube represents, and to work with the cube result using tidyverse tools.

Usage

```
as_cr_tibble(x, ...)
```

Arguments

x	a CrunchCube
...	further arguments passed on to <code>tibble::as_tibble()</code>

Details

The `cr_tibble` class is a subclass of `tibble` that has extra metadata to allow `ggplot::autoplot()` to work. If you find that this extra metadata is getting in the way, you can use `as_tibble()` to get a true `tibble`.

Description

The Crunch autoplot methods generate ggplots that are tailored to various Crunch objects. This allows you to visualize the object without bringing it into memory. You can select between three families of plots, which will attempt to accommodate the dimensionality of the plotted object. These plots can be further extended and customized with other ggplot methods.

Usage

```
## S3 method for class 'DatetimeVariable'
autoplot(object, ...)

## S3 method for class 'NumericVariable'
autoplot(object, ...)

## S3 method for class 'CategoricalVariable'
autoplot(object, ...)

## S3 method for class 'CategoricalArrayVariable'
autoplot(object, ...)

## S3 method for class 'MultipleResponseVariable'
autoplot(object, ...)

## S3 method for class 'CrunchCube'
autoplot(object, ...)

## S3 method for class 'CrunchCubeCalculation'
autoplot(object, plot_type = "dot", ...)

## S3 method for class 'tbl_crunch_cube'
autoplot(object, plot_type = c("dot", "tile", "bar"), measure, ...)
```

Arguments

object	A Crunch variable or cube aggregation
...	additional plotting arguments
plot_type	One of "dot", "tile", or "bar" which indicates the plot family you would like to use. Higher dimensional plots add color coding or facets depending on the dimensionality of the data.
measure	The measure you wish to plot. This will usually be "count", the default but can also be ".unweighted_counts" or any other measure stored in the cube. If omitted, autoplot will select the first measure appearing in the data.

Value

A ggplot object.

collect	<i>Collect a Crunch dataset from the server</i>
---------	---

Description

This function brings a Crunch dataset into memory so that you can work with the data using R functions. Since this can create a long running query it is recommended that you try to filter the dataset down as much as possible before running `collect()`.

Usage

```
## S3 method for class 'CrunchDataset'  
collect(x, ...)  
  
## S3 method for class 'GroupedCrunchDataset'  
collect(x, ...)
```

Arguments

x	A Crunch Dataset
...	Other arguments passed to <code>crunch::as.data.frame()</code>

Details

When collecting a grouped `CrunchDataset`, the grouping will be preserved.

Value

A `tbl_df` or `grouped_df`

Examples

```
## Not run:  
ds %>%  
  group_by(cyl) %>%  
  select(cyl, gear) %>%  
  collect()  
  
## End(Not run)
```

filter	<i>Filter a Crunch dataset</i>
--------	--------------------------------

Description

This function applies a `CrunchLogicalExpression` filter to a `CrunchDataset`. It's a "tidy" way of doing `ds[ds$var == val,]`.

Usage

```
## S3 method for class 'CrunchDataset'
filter(.data, ..., .preserve = FALSE)
```

Arguments

<code>.data</code>	A <code>CrunchDataset</code>
<code>...</code>	filter expressions
<code>.preserve</code>	Relevant when the <code>.data</code> input is grouped. If <code>.preserve = FALSE</code> (the default), the grouping structure is recalculated based on the resulting data, otherwise the grouping is kept as is.

Value

`.data` with the filter expressions applied.

Examples

```
## Not run:
ds %>%
  select(cyl, gear) %>%
  filter(cyl > 4) %>%
  collect()

## End(Not run)
```

<code>filter_.CrunchDataset</code>	<i>Filter a Crunch dataset (deprecated)</i>
------------------------------------	---

Description

This function is deprecated, use `filter()` instead. Applies a `CrunchLogicalExpression` filter to a `CrunchDataset`. It's a "tidy" way of doing `ds[ds$var == val,]`.

Usage

```
## S3 method for class 'CrunchDataset'
filter_(.data, ..., .dots)
```

Arguments

.data	A CrunchDataset
...	filter expressions
.dots	More dots!

Value

.data with the filter expressions applied.

GroupedCrunchDataset-class

A Crunch Dataset "Grouped By" Something

Description

This is a subclass of `crunch::CrunchDataset` that has a field for recording "group_by" expressions.

Examples

```
## Not run:
ds <- loadDataset("Your dataset name")
class(ds) ## "CrunchDataset"
grouped_ds <- group_by(ds, var1)
class(grouped_ds) ## "GroupedCrunchDataset"

## End(Not run)
```

group_by

Group-by for Crunch datasets

Description

`group_by()` sets grouping variables that affect what `summarize()` computes. `ungroup()` removes any grouping variables.

Usage

```
## S3 method for class 'CrunchDataset'
group_by(.data, ..., .add = FALSE)

## S3 method for class 'CrunchDataset'
ungroup(x, ...)
```

Arguments

<code>.data</code>	For <code>group_by()</code> , a Crunch Dataset
<code>...</code>	references to variables to group by, passed to <code>dplyr::group_by_prepare()</code>
<code>.add</code>	Logical: add the variables in <code>...</code> to any existing grouping variables, or replace them (the default).
<code>x</code>	For <code>ungroup()</code> , a Crunch Dataset

Details

Note that `group_by()` only supports grouping on variables that exist in the dataset, not ones that are derived on the fly. `dplyr::group_by()` supports that by calling `mutate()` internally, but `mutate` is not yet supported in `crplyr`.

Value

`group_by()` returns a `GroupedCrunchDataset` object (a `CrunchDataset` with grouping annotations). `ungroup()` returns a `CrunchDataset`.

Examples

```
## Not run:
ds %>%
  group_by(cyl) %>%
  select(cyl, gear) %>%
  collect()

## End(Not run)
```

mutate

Mutate Crunch datasets (not implemented)

Description

Just a method that returns a nicer error message. `mutate()` hasn't been implemented yet. You can, however, derive expressions on the fly in `summarize()`.

Usage

```
## S3 method for class 'CrunchDataset'
mutate(.data, ...)
```

Arguments

<code>.data</code>	A crunch Dataset
<code>...</code>	Other arguments, currently ignored

select	<i>Select columns from a Crunch dataset</i>
--------	---

Description

This function uses "tidy select" methods of subsetting the columns of a dataset. It's another way of doing `ds[, vars]`.

Usage

```
## S3 method for class 'CrunchDataset'
select(.data, ...)
```

Arguments

<code>.data</code>	A <code>CrunchDataset</code>
<code>...</code>	names of variables in <code>.data</code> or other valid selection functions, passed to <code>tidyselect::vars_select()</code>

Value

`.data` with only the selected variables.

Examples

```
## Not run:
ds %>%
  select(contains("ear")) %>%
  filter(gear > 4) %>%
  collect()

## End(Not run)
```

summarize	<i>Aggregate a Crunch dataset</i>
-----------	-----------------------------------

Description

This is an alternate interface to `crunch::crtabs()` that, in addition to being "tidy", makes it easier to query multiple measures at the same time.

Usage

```
## S3 method for class 'CrunchDataset'
summarise(.data, ...)
```


Arguments

`.data` A CrunchDataset
`...` named aggregations to include in the resulting table.

Details

Note that while `mutate()` is not generally supported in `crplyr`, you can derive expressions on the fly in `summarize()`.

Value

A `tbl_crunch_cube` or `cr_tibble` of results. This subclass of `tibble` allows `ggplot2::autoplot` to work, but can get in the way in some tidyverse operations. You may wish to convert to a `tibble` using `as_tibble()`.

Examples

```
## Not run:  
ds %>%  
  filter(cyl == 6) %>%  
  group_by(vs) %>%  
  summarize(hp=mean(hp), sd_hp=sd(hp), count=n())  
  
## End(Not run)
```

theme_crunch	<i>Crunch ggplot theme</i>
--------------	----------------------------

Description

Style ggplots according to Crunch style.

Usage

```
theme_crunch(base_size = 12, base_family = "sans")
```

Arguments

`base_size` Base text size
`base_family` Base text family

`unweighted_n`*Return the unweighted counts from summarize*

Description

This function allows you to return the unweighted counts from a Crunch dataset or grouped crunch dataset. It can only be used from within a `summarise()` call. If your dataset is unweighted, then `unweighted_n()` is equivalent to `n()`.

Usage`unweighted_n()`**Examples**

```
## Not run:
ds %>%
  group_by(cyl) %>%
  summarize(
    raw_counts = unweighted_n(),
    mean = mean(wt)
  )
## End(Not run)
```

Index

`as_cr_tibble`, 2
`autoplot`, 3

`collect`, 4

`dplyr::group_by_prepare()`, 7

`filter`, 5
`filter_.CrunchDataset`, 5

`group_by`, 6
`GroupedCrunchDataset`
 (`GroupedCrunchDataset-class`), 6
`GroupedCrunchDataset-class`, 6

`mutate`, 7

`select`, 8
`summarise.CrunchDataset (summarize)`, 8
`summarize`, 8
`summarize()`, 6, 7

`theme_crunch`, 9

`ungroup.CrunchDataset (group_by)`, 6
`unweighted_n`, 10