Package 'crunch'

August 10, 2016

Type Package

Title Crunch.io Data Tools

Description The Crunch.io service (http://crunch.io/) provides a cloud-based data store and analytic engine, as well as an intuitive web interface. Using this package, analysts can interact with and manipulate Crunch datasets from within R. Importantly, this allows technical researchers to collaborate naturally with team members, managers, and clients who prefer a point-and-click interface.

Version 1.12.2

Date 2016-08-10

Author Neal Richardson [aut, cre]

Maintainer Neal Richardson <neal@crunch.io>

URL https://github.com/Crunch-io/rcrunch

BugReports https://github.com/Crunch-io/rcrunch/issues

License LGPL (>= 3)

Depends R (>= 3.0.0)

Imports httr (>= 1.0.0), httpcache (>= 0.1.4), jsonlite (>= 0.9.15), methods, curl

Suggests knitr, testthat

VignetteBuilder knitr

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-08-10 21:37:58

R topics documented:

`addSubvariable` *Add subvariable to an array*

Description

This function conceals the dirty work in making this happen. The array gets unbound and rebound into a new array with the new variable added.

Usage

```
addSubvariable(variable, subvariable)
```

Arguments

`variable`	the array variable to modify
`subvariable`	the subvariable to add

Value

a new version of `variable` with the indicated subvariables

addVariables *Add multiple variables to a dataset*

Description

This function lets you add more than one variable at a time to a dataset. If you have multiple variables to add, this function will be faster than doing `ds$var <- value` assignment because it doesn't refresh the dataset's state in between variable POST requests.

Usage

```
addVariables(dataset, ...)
```

Arguments

`dataset`	a CrunchDataset
`...`	`VariableDefinition`s or a list of VariableDefinitions.

Value

`dataset` with the new variables added (invisibly)

appendDataset *Append one Crunch dataset to another*

Description

Append one Crunch dataset to another

Usage

```
appendDataset(dataset1, dataset2, cleanup = TRUE)
```

Arguments

`dataset1`	a CrunchDataset
`dataset2`	another CrunchDataset, or possibly a data.frame. If `dataset2` is not a Crunch dataset, it will be uploaded as a new dataset before appending.
`cleanup`	Deprecated. See `cleanseBatches`.

Value

A CrunchDataset with `dataset2` appended to `dataset1`

`as.environment,CrunchDataset-method`

as.environment method for CrunchDataset

Description

This method allows you to `eval` within a Dataset.

Usage

```
## S4 method for signature 'CrunchDataset'
as.environment(x)
```

Arguments

`x`	CrunchDataset

Value

an environment in which named objects are (promises that return) CrunchVariables.

`as.vector,CrunchExpr-method`

Convert Variables to local R objects

Description

Convert Variables to local R objects

Usage

```
## S4 method for signature 'CrunchExpr'
as.vector(x, mode = "any")

## S4 method for signature 'CrunchVariable'
as.vector(x, mode = "any")
```

Arguments

`x`	a CrunchVariable subclass
`mode`	for Categorical variables, one of either "factor" (default, which returns the values as factor); "numeric" (which returns the numeric values); or "id" (which returns the category ids). If "id", values corresponding to missing categories will return as the underlying integer codes; i.e., the R representation will not have any `NA`s. Otherwise, missing categories will all be returned `NA`. For non-Categorical variables, the `mode` argument is ignored.

Value

an R vector of the type corresponding to the Variable. E.g. CategoricalVariable yields type factor by default, NumericVariable yields numeric, etc.

`batches` *See the appended batches of this dataset*

Description

See the appended batches of this dataset

Usage

```
batches(x)
```

Arguments

`x`	a `CrunchDataset`

Value

a `BatchCatalog`

`c-categories` *S3 method to concatenate Categories and Category objects*

Description

S3 method to concatenate Categories and Category objects

Usage

```
## S3 method for class 'Categories'
c(...)

## S3 method for class 'Category'
c(...)
```

Arguments

`...`	see `c`

Value

An object of class `Categories`

Examples

```
cat.a <- Category(name="First", id=1, numeric_value=1, missing=FALSE)
cat.b <- Category(name="Second", id=2)
cat.c <- Category(name="Third", id=3, missing=TRUE)
cats.1 <- Categories(cat.a, cat.b)
identical(cats.1, c(cat.a, cat.b))
identical(c(cats.1, cat.c), Categories(cat.a, cat.b, cat.c))
```

`catalog-extract` *Extract and modify subsets of Catalog-type objects*

Description

Extract and modify subsets of Catalog-type objects

Usage

```
## S4 method for signature 'DatasetCatalog,character'
x[[i, j, ...]]

## S4 method for signature 'DatasetCatalog,ANY'
x[[i, j, ...]]

## S4 replacement method for signature 'DatasetCatalog,character,missing,DatasetTuple'
x[[i, j]] <- value

## S4 method for signature 'FilterCatalog,character'
x[[i, j, ...]]

## S4 method for signature 'FilterCatalog,numeric'
x[[i, j, ...]]

## S4 replacement method for signature 'FilterCatalog,character,missing,CrunchLogicalExpr'
x[[i, j]] <-
  value

## S4 replacement method for signature 'FilterCatalog,numeric,missing,CrunchLogicalExpr'
x[[i, j]] <- value

## S4 replacement method for signature 'FilterCatalog,character,missing,CrunchFilter'
x[[i, j]] <- value

## S4 replacement method for signature 'FilterCatalog,numeric,missing,CrunchFilter'
x[[i, j]] <- value

## S4 method for signature 'MemberCatalog,character'
x[[i, j, ...]]
```

```
## S4 method for signature 'MemberCatalog,character,ANY'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'MemberCatalog,ANY,missing,ANY'
x[[i, j]] <- value

## S4 replacement method for signature 'MemberCatalog,character,missing,`NULL`'
x[[i, j]] <- value

## S4 method for signature 'PermissionCatalog,character'
x[[i, j, ...]]

## S4 method for signature 'PermissionCatalog,character,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'ProjectCatalog,character'
x[[i, j, ...]]

## S4 method for signature 'ProjectCatalog,ANY'
x[[i, j, ...]]

## S4 replacement method for signature 'ProjectCatalog,character,missing,list'
x[[i, j]] <- value

## S4 replacement method for signature 'ProjectCatalog,character,missing,CrunchProject'
x[[i, j]] <- value

## S4 method for signature 'ShojiCatalog,character,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'ShojiCatalog,numeric,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'ShojiCatalog,logical,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'ShojiCatalog,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'ShojiCatalog,ANY'
x[[i, j, ...]]

## S4 method for signature 'ShojiCatalog,character'
x[[i, j, ...]]

## S4 method for signature 'ShojiCatalog'
x$name
```

```
## S4 replacement method for signature 'ShojiCatalog'
x$name <- value

## S4 replacement method for signature 'ShojiCatalog,ANY,missing,ShojiCatalog'
x[i, j] <- value

## S4 method for signature 'TeamCatalog,character'
x[[i, j, ...]]

## S4 method for signature 'TeamCatalog,numeric'
x[[i, j, ...]]

## S4 replacement method for signature 'TeamCatalog,character,missing,list'
x[[i, j]] <- value

## S4 replacement method for signature 'TeamCatalog,character,missing,CrunchTeam'
x[[i, j]] <- value

## S4 method for signature 'VariableCatalog,character'
x[[i, j, ...]]

## S4 method for signature 'VariableCatalog,ANY'
x[[i, j, ...]]

## S4 replacement method for signature 'VariableCatalog,character,missing,VariableTuple'
x[[i, j]] <- value

## S4 replacement method for signature 'VariableCatalog,character,missing,CrunchVariable'
x[[i, j]] <-
  value

## S4 method for signature 'VariableCatalog,VariableOrder,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'VariableCatalog,VariableGroup,ANY'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature
## 'VariableCatalog,VariableOrder,missing,VariableCatalog'
x[i, j] <-
  value

## S4 replacement method for signature
## 'VariableCatalog,VariableGroup,missing,VariableCatalog'
x[i, j] <-
  value
```

Arguments

x	a Catalog object
i	which catalog elements to extract
j	Invalid
...	additional arguments
value	For updating, an object of the appropriate class and size to insert
drop	Invalid
name	for $, the same as i for [[

Value

A subset of x if extracting, otherwise x duly modified

---

catalog-length *Length of Catalog*

Description

Length of Catalog

Usage

```
## S4 method for signature 'ShojiCatalog'
length(x)
```

Arguments

x	a Catalog

Value

Integer: the number of elements in the index list

catalogToDataFrame *Utility to get a more human-readable view of a Shoji Catalog*

Description

Utility to get a more human-readable view of a Shoji Catalog

Usage

```
catalogToDataFrame(x, keys = TRUE, rownames, ...)
```

Arguments

x	ShojiCatalog or subclass
keys	character vector of attribute names from each catalog tuple to include in the result. Default is TRUE, which means all.
rownames	See data.frame, the row.names argument, to which this is passed in data.frame. The difference here is that if rownames is explicitly set as NULL, the resulting object will not have row names set. By default, row names will be the URLs of the catalog tuples.
...	additional arguments passed to data.frame

Value

a data.frame view of the catalog

Categories-class *Categories in CategoricalVariables*

Description

CategoricalVariables, as well as the array types composed from Categoricals, contain Categories. Categories are a subclass of list that contains only Category objects. Category objects themselves subclass list and contain the following fields: "name", "id", "numeric_value", "missing", and optionally "selected".

Usage

```
Categories(..., data = NULL)

Category(..., data = NULL)

## S4 method for signature 'Categories,ANY,ANY'
x[i, j, ..., drop = TRUE]
```

```
## S4 method for signature 'Categories,numeric,ANY'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'Categories,ANY,ANY,ANY'
x[i, j, ...] <- value

## S4 method for signature 'Categories'
names(x)

## S4 method for signature 'Categories'
values(x)

## S4 method for signature 'Categories'
ids(x)

## S4 replacement method for signature 'Categories'
names(x) <- value

## S4 replacement method for signature 'Categories'
values(x) <- value

## S4 replacement method for signature 'Categories'
ids(x) <- value
```

Arguments

...	additional arguments to [, ignored
data	For the constructor functions Category and Categories, you can either pass in attributes via ... or you can create the objects with a fully defined list representation of the objects via the data argument. See the examples.
x	For the attribute getters and setters, an object of class Category or Categories
i	For the [methods, just as with list extract methods
j	Invalid argument to [, but in the generic's signature
drop	Invalid argument to [, but in the generic's signature
value	For [<-, the replacement Category to insert

Examples

```
cat.a <- Category(name="First", id=1, numeric_value=1, missing=FALSE)
cat.b <- Category(data=list(name="First", id=1, numeric_value=1, missing=FALSE))
identical(cat.a, cat.b)
cat.c <- Category(name="Second", id=2)
cats.1 <- Categories(cat.a, cat.c)
cats.2 <- Categories(data=list(cat.a, cat.c))
identical(cats.1, cats.2)
```

category-extract *Access Category fields directly*

Description

Don't do this. Instead, use the category setters.

Usage

```
## S4 method for signature 'Category'
x$name

## S4 replacement method for signature 'Category'
x$name <- value
```

Arguments

x	a Category
name	a field within x
value	a value for that field to update

Value

$ returns the value of the desired field. Setter returns x duly modified.

See Also

describe-category

cleanseBatches *Remove batches from a dataset*

Description

Sometimes append operations do not succeed, whether due to conflicts between the two datasets or other server-side issues. Failed appends can leave behind "error" status batch records, which can cause confusion. This function lets you delete batches that don't match the status or statuses you want to keep.

Usage

```
cleanseBatches(dataset, keep = c("imported", "appended"))
```

Arguments

dataset	CrunchDataset
keep	character batch status(es) you want to keep. By default, batches that don't have either "imported" or "appended" status will be deleted.

Value

dataset with the undesired batches removed.

---

combine *Combine categories or responses*

Description

Combine categories or responses

Usage

```
combine(variable, combinations = list(), ...)
```

Arguments

variable	Categorical, Categorical Array, or Multiple Response variable
combinations	list of named lists containing (1) "categories": category ids or names for categorical types, or for multiple response, "responses": subvariable names, aliases, or positional indices; (2) a "name" for the new category or response; and (3) optionally, other category ("missing", "numeric_value") or subvariable ("alias", "description") attributes. If combinations is omitted, the resulting variable will essentially be a copy (but see link{copy} for a more natural way to do that, if desired).
...	Additional variable metadata for the new derived variable

Value

A VariableDefinition that will create the new comined-category or -response derived variable. Categories/responses not referenced in combinations will be appended to the end of the combinations.

Examples

```
## Not run:
ds$fav_pet2 <- combine(ds$fav_pet, name="Pets (combined)",
    combinations=list(list(name="Mammals", categories=c("Cat", "Dog")),
                      list(name="Reptiles", categories=c("Snake", "Lizard"))))
ds$pets_owned2 <- combine(ds$allpets, name="Pets owned (collapsed)",
    combinations=list(list(name="Mammals", responses=c("Cat", "Dog"))))

## End(Not run)
```

`compareDatasets` *Compare two datasets to see how they will append*

Description

When one dataset is appended to another, variables and subvariables are matched on their aliases, and then categories for variables that have them are matched on category name. This function lines up the metadata between two datasets as the append operation will so that you can inspect how well the datasets will align before you do the append.

Usage

```
compareDatasets(A, B)
```

Arguments

`A`	CrunchDataset
`B`	CrunchDataset

Details

Calling `summary` on the return of this function will print an overview of places where the matching on variable alias and category name may lead to undesired outcomes, enabling you to alter one or both datasets to result in better alignment.

Value

An object of class 'compareDatasets', a list of three elements: (1) 'variables', a data.frame of variable metadata joined on alias; (2) 'categories', a list of data.frames of category metadata joined on category name, one for each variable with categories; and (3) 'subvariables', a list of data.frames of subvariable metadata joined on alias, one for each array variable.

Summary output reports on (1) variables that, when matched across datasets by alias, have different types; (2) variables that have the same name but don't match on alias; (3) for variables that match and have categories, any categories that have the same id but don't match on name; (4) for array variables that match, any subvariables that have the same name but don't match on alias; and (5) array variables that, after assembling the union of their subvariables, point to subvariables that belong to other arrays.

Examples

```
## Not run:
    comp <- compareDataset(ds1, ds2)
    summary(comp)

## End(Not run)
```

consent *Give consent to do things that require permission*

Description

Give consent to do things that require permission

Usage

```
consent()
```

Value

an S3 class "contextManager" object

See Also

with-context-manager ContextManager

ContextManager *Context managers*

Description

Context managers

Usage

```
ContextManager(enter = function() { }, exit = function() { },
  error = NULL, as = NULL)
```

Arguments

enter	function to run before doing things
exit	function to run after doing things
error	optional function to run if an error is thrown
as	character optional way to specify a default name for assinging the return of the enter function.

Value

an S3 class "contextManager" object

See Also

with-context-manager

`copyVariable` *Copy a variable*

Description

Makes a copy of a Crunch variable on the server.

Usage

```
copyVariable(x, deep = FALSE, ...)

copy(x, deep = FALSE, ...)
```

Arguments

`x`	a CrunchVariable to copy
`deep`	logical: should this be a deep copy, in which there is no dependence on the original variable, or a shallow one, in which the copy is more of a symbolic link? Default is `FALSE`, meaning symlink.
`...`	Additional metadata to give to the new variable. If not given, the new variable will have a name that is the same as the original but with " (copy)" appended, and its alias will be the old alias with "_copy" appended.

Details

Copies can be shallow (linked) or deep. Shallow copying is faster and should be preferred unless a true hard copy is required, though keep in mind the implications of shallow copying. When you append data to the original variable or otherwise alter its values, the values in the copy automatically update. This linking may be desirable, but it comes with some limitations. First, you cannot edit the values of the copy independently of the original. Second, some attributes of the copy are immutable: of note, properties of categories cannot be altered independely in the copy. Subvariable names and ordering within arrays, however, can.

Value

a VariableDefinition for the copy expression. Assign into a Dataset to make the copy happen.

crtabs — *Crunch xtabs: Crosstab and otherwise aggregate variables in a Crunch Dataset*

Description

Create a contingency table or other aggregation from cross-classifying variables in a CrunchDataset.

Usage

```
crtabs(formula, data, weight = crunch::weight(data), useNA = c("no",
  "ifany", "always"))
```

Arguments

formula	an object of class 'formula' object with the cross-classifying variables, separated by '+', on the right hand side. Compare to xtabs.
data	an object of class CrunchDataset
weight	a CrunchVariable that has been designated as a potential weight variable for data, or NULL for unweighted results. Default is the currently applied weight, weight(data).
useNA	whether to include missing values in tabular results. See table.

Value

an object of class CrunchCube

crunch — *Crunch.io: instant, visual, collaborative data analysis*

Description

Crunch.io provides a cloud-based data store and analytic engine. It has a web client for interactive data exploration and visualization. The crunch package for R allows analysts to interact with and manipulate Crunch datasets from within R. Importantly, this allows technical researchers to collaborate naturally with team members, managers, and clients who prefer a point-and-click interface: because all connect to the same dataset in the cloud, there is no need to email files back and forth continually to share results.

See Also

To learn more about using the package, see `vignette("getting-started", package="crunch")`. To sign up for a Crunch.io account, visit https://beta.crunch.io/.

crunch-uni *Univariate statistics on Crunch objects*

Description

Univariate statistics on Crunch objects

Usage

```
## S4 method for signature 'CrunchVariable'
mean(x, ...)

## S4 method for signature 'NumericVariable'
mean(x, ...)

## S4 method for signature 'CrunchVariable'
sd(x, na.rm = FALSE)

## S4 method for signature 'NumericVariable'
sd(x, na.rm = FALSE)

## S4 method for signature 'CrunchVariable'
median(x, na.rm = FALSE)

## S4 method for signature 'NumericVariable'
median(x, na.rm = FALSE)

## S4 method for signature 'CrunchVariable'
min(x, na.rm)

## S4 method for signature 'NumericVariable'
min(x, na.rm = FALSE)

## S4 method for signature 'DatetimeVariable'
min(x, na.rm = FALSE)

## S4 method for signature 'CrunchVariable'
max(x, na.rm)

## S4 method for signature 'NumericVariable'
max(x, na.rm = FALSE)

## S4 method for signature 'DatetimeVariable'
max(x, na.rm = FALSE)
```

Arguments

x	a NumericVariable, or for min and max, possibly a DatetimeVariable

...	additional arguments to mean
na.rm	logical: exclude missings?

See Also

mean sd median min max

CrunchDataset-class *Crunch Datasets*

Description

Crunch Datasets

CrunchVariable-class *Variables in Crunch*

Description

Variables are S4 objects. All inherit from the base class CrunchVariable. be persisted on the server? Default is FALSE

Slots

filter either NULL or CrunchLogicalExpr

---

cube-computing *Work with CrunchCubes*

Description

Crunch.io supports more complex data types than base R does, such as multiple response and array types. If you want to compute margin or proportion tables on an aggregation of these variable types, special methods are required. These functions provide an interface like margin.table and prop.table for the CrunchCube object, handling those special data types.

Usage

```
## S4 method for signature 'CrunchCube'
margin.table(x, margin = NULL)

## S4 method for signature 'CrunchCube'
prop.table(x, margin = NULL)

## S4 method for signature 'CrunchCube'
round(x, digits = 0)
```

Arguments

x	a CrunchCube
margin	index, or vector of indices to generate margin for. See prop.table
digits	see round

Value

The appropriate margin.table or prop.table.

See Also

margin.table prop.table

---

cube-methods *Methods on Cube objects*

Description

These methods provide an array-like interface to the CrunchCube object.

Usage

```
## S4 method for signature 'CubeDims'
dimnames(x)

## S4 method for signature 'CubeDims'
dim(x)

## S4 method for signature 'CubeDims'
is.na(x)
```

Arguments

x	a CrunchCube or its CubeDims component.

Value

Generally, the same shape of result that each of these functions return when applied to an array object.

See Also

cube-computing

dataset-extract *Subset datasets and extract variables*

Description

Subset datasets and extract variables

Usage

```
## S4 method for signature 'CrunchDataset,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'CrunchDataset,character,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'CrunchDataset,missing,ANY'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'CrunchDataset,CrunchLogicalExpr,missing'
x[i, j, ...,
  drop = FALSE]

## S4 method for signature 'CrunchDataset,CrunchLogicalExpr,ANY'
x[i, j, ..., drop = FALSE]

## S4 method for signature 'CrunchDataset'
subset(x, ...)

## S4 method for signature 'CrunchDataset,ANY'
x[[i, ..., drop = FALSE]]

## S4 method for signature 'CrunchDataset,character'
x[[i, ..., drop = FALSE]]

## S4 method for signature 'CrunchDataset'
x$name
```

Arguments

x	a CrunchDataset
i	As with a `data.frame`, there are two cases: (1) if no other arguments are supplied (i.e `x[i]`), `i` provides for `as.list` extraction: columns of the dataset rather than rows. If character, identifies variables to extract based on their aliases (by default: set `options(crunch.namekey.dataset="name")` to use variable names); if numeric or logical, extracts variables accordingly. Alternatively, (2) if `j` is specified (as either `x[i, j]` or `x[i,]`), `i` is an object of class `CrunchLogicalExpr` that will define a subset of rows.

j	columnar extraction, as described above
...	additional arguments
drop	logical: autmatically simplify a 1-column Dataset to a Variable? Default is FALSE, and the TRUE option is in fact not implemented.
name	columnar extraction for $

Value

[yields a Dataset; [[and $ return a Variable

---

dataset-owner *Change the owner of a dataset*

Description

Change the owner of a dataset

Usage

```
## S4 method for signature 'CrunchDataset'
owner(x)

## S4 replacement method for signature 'CrunchDataset'
owner(x) <- value
```

Arguments

x	CrunchDataset
value	For the setter, either a URL (character) or a Crunch object with a self method. Users and Projects are valid objects to assign as dataset owners.

Value

The dataset.

dataset-to-R *as.data.frame method for CrunchDataset*

Description

This method is defined principally so that you can use a CrunchDataset as a `data` argument to other R functions (such as `lm`). Unless you give it the `force==TRUE` argument, this function does not in fact return a `data.frame`: it returns an object with an interface like a data.frame, such that you get R vectors when you access its columns (unlike a CrunchDataset, which returns CrunchVariable objects). This allows modeling functions that require select columns of a dataset to retrieve only those variables from the remote server, rather than pulling the entire dataset into local memory.

Usage

```
## S3 method for class 'CrunchDataset'
as.data.frame(x, row.names = NULL, optional = FALSE,
  force = FALSE, ...)

## S3 method for class 'CrunchDataFrame'
as.data.frame(x, row.names = NULL,
  optional = FALSE, ...)
```

Arguments

`x`	a CrunchDataset
`row.names`	part of as.data.frame signature. Ignored.
`optional`	part of as.data.frame signature. Ignored.
`force`	logical: actually coerce the dataset to `data.frame`, or leave the columns as unevaluated promises. Default is `FALSE`.
`...`	additional arguments passed to as.data.frame.default

Value

an object of class `CrunchDataFrame` unless `force`, in which case the return is a `data.frame`.

dataset-update *Update a variable or variables in a dataset*

Description

Update a variable or variables in a dataset

Usage

```
## S4 replacement method for signature 'CrunchDataset,character,missing,CrunchVariable'
x[[i]] <- value

## S4 replacement method for signature 'CrunchDataset,ANY,missing,CrunchVariable'
x[[i]] <- value

## S4 replacement method for signature 'CrunchDataset,character,missing,ANY'
x[[i]] <- value

## S4 replacement method for signature 'CrunchDataset,character,missing,CrunchLogicalExpr'
x[[i]] <- value

## S4 replacement method for signature 'CrunchDataset,ANY,ANY,ANY'
x[[i]] <- value

## S4 replacement method for signature 'CrunchDataset,character,missing,`NULL`'
x[[i]] <- value

## S4 replacement method for signature 'CrunchDataset,ANY,missing,`NULL`'
x[[i]] <- value

## S4 replacement method for signature 'CrunchDataset'
x$name <- value

## S4 replacement method for signature 'CrunchDataset,ANY,missing,list'
x[i, j] <- value

## S4 replacement method for signature 'CrunchDataset,CrunchExpr,ANY,ANY'
x[i, j] <- value
```

Arguments

x	a CrunchDataset
i	For [, a CrunchLogicalExpr, numeric, or logical vector defining a subset of the rows of x. For [[, see j for the as.list column subsetting.
value	replacement values to insert. These can be crunchExprs or R vectors of the corresponding type
name	like j but for $
j	if character, identifies variables to extract based on their aliases (by default: set options(crunch.namekey.dataset="name") to use variable names); if numeric or logical, extracts variables accordingly. Note that this is the as.list extraction, columns of the dataset rather than rows.

Value

x, modified.

dataset-variables *Access a Dataset's Variables Catalog*

Description

Datasets contain collections of variables. For a few purposes, such as editing variables' metadata, it is helpful to access these variable catalogs more directly.

Usage

```
## S4 method for signature 'CrunchDataset'
variables(x)

## S4 replacement method for signature 'CrunchDataset,VariableCatalog'
variables(x) <- value

## S4 method for signature 'CrunchDataset'
allVariables(x)

## S4 replacement method for signature 'CrunchDataset,VariableCatalog'
allVariables(x) <- value
```

Arguments

x	a Dataset
value	For the setters, a VariableCatalog to assign.

Details

`variables` gives just the active variables in the dataset, while `allVariables`, as the name suggests, yields all variables, including hidden variables.

Value

Getters return VariableCatalog; setters return `x` duly modified.

DatasetOrder-class *Organize Datsets*

Description

A DatasetOrder object is a subclass of `list` that contains DatasetGroups. DatasetGroup objects contain a group name and an set of "entities", which can be dataset references or other nested DatasetGroups.

Slots

`group` character, the name of the DatasetGroup. In the constructor and more generally, this field can be referenced as "name" as well.

`entities` a character vector of dataset URLs, or a list containing a combination of dataset URLs and DatasetGroup objects.

`datasets` *Get the dataset catalog*

Description

Get the dataset catalog

Usage

```
datasets(x = getAPIRoot())

datasets(x) <- value
```

Arguments

`x`	a `ShojiObject`, such as a `CrunchProject`. If omitted, the default value for `x` means that you will load the user's primary dataset catalog.
`value`	`CrunchDataset` for the setter

Value

An object of class `DatasetCatalog`. The setter returns the project (or other object that contains a dataset catalog with the given dataset added to it (via changing its owner to be the specified object, `x`).

Examples

```
## Not run:
# Get the primary dataset catalog
mydatasets <- datasets()
# Can load a dataset from that
ds <- loadDataset(mydatasets[["Dataset name"]])
# Can use the same function to get the dataset catalog for a project
proj <- projects()[["Project name"]]
projdatasets <- datasets(proj)
# The assignment method lets you move a dataset to a project
datasets(proj) <- ds

## End(Not run)
```

delete *Delete a Crunch object from the server*

Description

These methods delete entities, notably Datasets and Variables within them, from the server. This action is permanent and cannot be undone, so it should not be done lightly. Consider instead using `archive` for datasets and `hide` for variables

Usage

```
## S4 method for signature 'CrunchDataset'
delete(x, confirm = requireConsent(), ...)

## S4 method for signature 'CrunchProject'
delete(x, confirm = requireConsent(), ...)

## S4 method for signature 'ShojiObject'
delete(x, ...)

## S4 method for signature 'ANY'
delete(x, ...)

## S4 method for signature 'CrunchTeam'
delete(x, confirm = requireConsent(), ...)

## S4 method for signature 'CrunchVariable'
delete(x, ...)

## S4 method for signature 'CategoricalArrayVariable'
delete(x, ...)
```

Arguments

`x`	a Crunch object
`confirm`	logical: should the user be asked to confirm deletion. Option available for datasets and teams only. Default is `TRUE` if in an interactive session. You can avoid the confirmation prompt if you delete `with(consent)`.
`...`	additional arguments, in the generic

See Also

`hide` `deleteDataset`

deleteDataset *Delete a dataset from the dataset list*

Description

This function lets you delete a dataset without first loading it. If you have a dataset that somehow is corrupted and won't load, you can delete it this way.

Usage

```
deleteDataset(x, ...)
```

Arguments

x	The name (character) of a dataset, its (numeric) position in the return of listDatasets, or an object of class CrunchDataset. x can only be of length 1–this function is not vectorized (for your protection).
...	additional parameters (such as confirm) passed to delete

Details

The function also works on CrunchDataset objects, just like delete, which may be useful if you have loaded another package that masks the delete method.

Value

(Invisibly) the API response from deleting the dataset

See Also

delete

deleteSubvariables *Delete subvariables from an array*

Description

This function conceals the dirty work in making this happen. The array gets unbound, the subvariables deleted, and then the remaining subvariable are rebound into a new array.

Usage

```
deleteSubvariables(variable, to.delete)

deleteSubvariable(variable, to.delete)
```

Arguments

variable	the array variable
to.delete	character alias(es) or name(s), depending on the value of getOption("crunch.namekey.array"), of the subvariables to delete.

Value

a new version of variable without the indicated subvariables

---

deleteVariables *Delete Variables Within a Dataset*

Description

Delete Variables Within a Dataset

Usage

```
deleteVariables(dataset, variables = NULL, pattern = NULL,
  key = namekey(dataset), confirm = requireConsent(), ...)

deleteVariable(dataset, variables = NULL, pattern = NULL,
  key = namekey(dataset), confirm = requireConsent(), ...)
```

Arguments

dataset	the Dataset to modify
variables	names or indices of variables to delete
pattern	optional regular expression to identify Variables to delete. Note that this argument is deprecated. If you wish to grep, you can grep(pattern, aliases(variables(dataset))) or similar outside this function.
key	the Variable attribute to grep with the pattern. Default is "alias"
confirm	logical: should the user be asked to confirm deletion. Default is TRUE if in an interactive session. You can avoid the confirmation prompt if you delete with(consent).
...	optional additional arguments to grep. Likewise deprecated.

Value

(invisibly) dataset with the specified variables deleted

See Also

hide

describe *Name, alias, and description for Crunch objects*

Description

Name, alias, and description for Crunch objects

Usage

```
## S4 method for signature 'CrunchDataset'
name(x)

## S4 replacement method for signature 'CrunchDataset'
name(x) <- value

## S4 method for signature 'CrunchDataset'
description(x)

## S4 replacement method for signature 'CrunchDataset'
description(x) <- value

## S4 method for signature 'CrunchDataset'
startDate(x)

## S4 replacement method for signature 'CrunchDataset'
startDate(x) <- value

## S4 method for signature 'CrunchDataset'
endDate(x)

## S4 replacement method for signature 'CrunchDataset'
endDate(x) <- value

## S4 method for signature 'CrunchDataset'
id(x)

## S4 method for signature 'CrunchDataset'
notes(x)

## S4 replacement method for signature 'CrunchDataset'
notes(x) <- value

## S4 method for signature 'ShojiObject'
name(x)

## S4 method for signature 'CrunchVariable'
name(x)
```

```
## S4 replacement method for signature 'CrunchVariable'
name(x) <- value

## S4 method for signature 'CrunchVariable'
description(x)

## S4 replacement method for signature 'CrunchVariable'
description(x) <- value

## S4 method for signature 'CrunchVariable'
alias(object)

## S4 replacement method for signature 'CrunchVariable'
alias(x) <- value

## S4 method for signature 'CrunchVariable'
notes(x)

## S4 replacement method for signature 'CrunchVariable'
notes(x) <- value
```

Arguments

x	a Dataset or Variable.
value	For the setters, a length-1 character vector to assign
object	Same as x but for the alias method, in order to match the generic from another package. Note that alias is only defined for Variables.

Value

Getters return the character object in the specified slot; setters return x duly modified.

See Also

Categories describe-catalog

`describe-category` *Category attributes*

Description

Category attributes

Usage

```
## S4 method for signature 'Category'
name(x)

## S4 replacement method for signature 'Category'
name(x) <- value

## S4 replacement method for signature '`NULL`'
name(x) <- value

## S4 method for signature 'Category'
value(x)

## S4 replacement method for signature 'Category'
value(x) <- value

## S4 method for signature 'Category'
id(x)

## S4 method for signature 'Category'
is.selected(x)
```

Arguments

x	a Category
value	For the setters, an appropriate value to set

Value

`name` returns character; `value` and `id` return numeric; value but not id may be `NA`; `is.selected` returns logical indicating whether this Category is a "selected" dichotomy. Setters return `x` duly modified.

See Also

`Categories` `dichotomize`

`dichotomize` *Indicate how categories represent a dichotomized value*

Description

Multiple Response variables are essentially Categorical Arrays that have had a category or categories indicated as the "selected" value. These methods let you set that state.

Usage

```
## S4 method for signature 'Categories'
is.dichotomized(x)

## S4 method for signature 'Categories,numeric'
dichotomize(x, i)

## S4 method for signature 'Categories,logical'
dichotomize(x, i)

## S4 method for signature 'Categories,character'
dichotomize(x, i)

## S4 method for signature 'Categories'
undichotomize(x)

## S4 method for signature 'CategoricalVariable,ANY'
dichotomize(x, i)

## S4 method for signature 'CategoricalArrayVariable,ANY'
dichotomize(x, i)

## S4 method for signature 'CategoricalVariable'
undichotomize(x)

## S4 method for signature 'CategoricalArrayVariable'
undichotomize(x)
```

Arguments

x	Categories or a Variable subclass that has Categories
i	For the dichotomize methods, the numeric or logical indices of the categories to mark as "selected", or if character, the Category "names". Note that unlike some other categorical variable methods, numeric indices are positional, not with reference to category ids.

Details

dichotomize lets you specify which categories are "selected", while undichotomize strips that selection information. Dichotomize converts a Categorical Array to a Multiple Response, and undichotomize converts back.

Value

Categories or the Variable, (un)dichotomized accoringly

See Also

describe-category

dim-dataset *Dataset dimensions*

Description

Dataset dimensions

Usage

```
## S4 method for signature 'CrunchDataset'
dim(x)

## S4 method for signature 'CrunchDataset'
ncol(x)
```

Arguments

x	a Dataset

Value

integer vector of length 2, indicating the number of rows and non-hidden variables in the dataset. Array subvariables are excluded from the column count.

See Also

dim

dropRows *Permanently delete rows from a dataset*

Description

Permanently delete rows from a dataset

Usage

```
dropRows(dataset, expr)
```

Arguments

dataset	a CrunchDataset
expr	a CrunchLogicalExpr

Value

dataset without the rows indicated by expr

See Also

exclusion for a non-destructive way to suppress rows

Examples

```
## Not run:
ds <- dropRows(ds, ds$gender == "Male")

## End(Not run)
```

---

entity,CrunchProject-method

Methods for ShojiTuples

Description

ShojiTuples are objects extracted from ShojiCatalogs. They are internally used.

Usage

```
## S4 method for signature 'CrunchProject'
entity(x)

## S4 method for signature 'ShojiTuple'
refresh(x)

## S4 method for signature 'ShojiTuple'
x$name

## S4 replacement method for signature 'ShojiTuple'
x$name <- value

## S4 method for signature 'ShojiTuple,ANY'
x[[i]]

## S4 replacement method for signature 'ShojiTuple,ANY,ANY,ANY'
x[[i]] <- value

## S4 method for signature 'ShojiTuple'
self(x)

## S4 method for signature 'VariableTuple'
entity(x)

## S4 method for signature 'CrunchVariable'
entity(x)
```

```
## S4 method for signature 'DatasetTuple'
entity(x)

## S4 method for signature 'ShojiTuple'
delete(x, ...)

## S4 method for signature 'DatasetTuple'
delete(x, confirm = requireConsent(), ...)

## S4 method for signature 'ShojiTuple'
name(x)

## S4 replacement method for signature 'ShojiTuple'
name(x) <- value

## S4 method for signature 'ShojiTuple'
type(x)
```

Arguments

x	a Tuple
name	a Tuple slot to get or set
value	What to set in a given slot
i	In [[, a Tuple slot to get
...	additional arguments to [[, ignored
confirm	For delete, whether confirmation is required. See delete.

---

exclusion *View and set exclusion filters*

Description

Exclusion filters express logic that defines a set of rows that should be dropped from the dataset. The rows aren't permanently deleted—you can recover them at any time by removing the exclusion filter—but they are omitted from all views and calculations, as if they had been deleted.

Usage

```
exclusion(x)

exclusion(x) <- value
```

Arguments

x	a Dataset
value	an object of class CrunchLogicalExpr, or NULL

Details

Note that exclusion filters work opposite from how "normal" filters work. That is, a regular filter expression defines the subset of rows to operate on: it says "keep these rows." An exclusion filter defines which rows to omit. Applying a filter expression as a query filter will have the opposite effect if applied as an exclusion. Indeed, applying it as both query filter and exclusion at the same time will result in 0 rows.

Value

`exclusion` returns a `CrunchFilter` if there is one, else `NULL`. The setter returns `x` with the filter set.

`exportDataset` *Export a dataset to a file*

Description

Export a dataset to a file

Usage

```
exportDataset(dataset, file, format = c("csv", "spss"),
  categorical = c("name", "id"), ...)

## S4 method for signature 'CrunchDataset'
write.csv(...)
```

Arguments

`dataset`	CrunchDataset
`file`	character local filename to write to
`format`	character export format: currently supported values are "csv" and "spss".
`categorical`	character: export categorical values to CSV as category "name" (default) or "id". Ignored by the SPSS exporter.
`...`	additional arguments, currently ignored

Value

Invisibly, `file`.

expressions *Construct Crunch Expressions*

Description

Crunch Expressions, i.e. `CrunchExpr` and `CrunchLogicalExpr`, encapuslate derivations of Crunch variables, which are only evaluated when passed to a function like `as.vector`. They allow you to compose functional expressions of variables and evaluate them against the server only when appropriate.

Usage

```
## S4 method for signature 'CrunchExpr'
!x

## S4 method for signature 'CategoricalVariable,character'
x %in% table

## S4 method for signature 'CategoricalVariable,factor'
x %in% table

## S4 method for signature 'TextVariable,character'
x %in% table

## S4 method for signature 'NumericVariable,numeric'
x %in% table

## S4 method for signature 'DatetimeVariable,Date'
x %in% table

## S4 method for signature 'DatetimeVariable,POSIXt'
x %in% table

## S4 method for signature 'DatetimeVariable,character'
x %in% table

## S4 method for signature 'CategoricalVariable,numeric'
x %in% table

## S4 method for signature 'CategoricalVariable,character'
e1 == e2

## S4 method for signature 'CategoricalVariable,factor'
e1 == e2

## S4 method for signature 'CategoricalVariable,character'
e1 != e2
```

```
## S4 method for signature 'CategoricalVariable,factor'
e1 != e2

## S4 method for signature 'CrunchVariable'
is.na(x)

bin(x)

rollup(x, resolution = rollupResolution(x))
```

Arguments

x	an input
table	For %in%. See match
e1	an input
e2	an input
resolution	For rollup. Either NULL or a character in c("Y", "Q", "M", "W", "D", "h", "m", "s", "ms") indicating the unit of time at which a Datetime variable should be aggregated. If NULL, the server will determine an appropriate resolution based on the range of the data.

Value

Most functions return a CrunchExpr or CrunchLogicalExpr. as.vector returns an R vector.

---

extendDataset *Add columns from one dataset to another, joining on a key*

Description

As merge does for data.frames, this function takes two datasets, matches rows based on a specified key variable, and adds columns from one to the other.

Usage

```
extendDataset(x, y, by.x, by.y, all = FALSE, all.x = TRUE, all.y = FALSE,
  ...)

## S3 method for class 'CrunchDataset'
merge(x, y, by.x, by.y, all = FALSE, all.x = TRUE,
  all.y = FALSE, ...)
```

Arguments

x	CrunchDataset to add data to
y	CrunchDataset to copy data from. May be filtered by rows and/or columns.
by.x	CrunchVariable in x on which to join. Must have all unique, non-missing values.
by.y	CrunchVariable in y on which to join. Must have all unique, non-missing values.
all	logical: should all rows in x and y be kept, i.e. a "full outer" join? Only FALSE is currently supported.
all.x	logical: should all rows in x be kept, i.e. a "left outer" join? Only TRUE is currently supported.
all.y	logical: should all rows in y be kept, i.e. a "right outer" join? Only FALSE is currently supported.
...	additional arguments, ignored

Value

x extended by the columns of y, matched on the "by" variables.

---

filter-catalog *Filter entities for a dataset*

Description

Filter entities for a dataset

Usage

```
## S4 method for signature 'CrunchDataset'
filters(x)

## S4 replacement method for signature 'CrunchDataset'
filters(x) <- value
```

Arguments

x	a CrunchDataset
value	for the assignment method, a FilterCatalog

Value

an object of class FilterCatalog containing references to Filter entities usable in the web application. (Setter returns the Dataset.)

filter-methods *View and modify Filter entity attributes*

Description

View and modify Filter entity attributes

Usage

```
## S4 method for signature 'CrunchFilter'
is.public(x)

## S4 replacement method for signature 'CrunchFilter'
is.public(x) <- value
```

Arguments

x	a CrunchFilter
value	an attribute to set

Value

For `is.public`, a logical value for whether the filter is flagged as shared with all dataset viewers. (Its setter thus takes a logical value as well.)

forkDataset *Create a fork of a dataset*

Description

As with many other version control systems, in Crunch you can fork a dataset's revision history, effectively making a copy on which you can work independently of the original dataset. You can then merge those change back to the original dataset or keep working independently.

Usage

```
forkDataset(dataset, name = defaultForkName(dataset), draft = FALSE, ...)
```

Arguments

dataset	The `CrunchDataset` to fork
name	character name to give the fork. If omitted, one will be provided for you
draft	logical: Should the dataset be a draft, available only to editors? Default is `FALSE`.
...	Additional dataset metadata

Value

The new fork, a `CrunchDataset`.

`getAccountUserCatalog` *Find all users on your account*

Description

Find all users on your account

Usage

```
getAccountUserCatalog(x = shojiURL(getAccount(), "catalogs", "users"))
```

Arguments

`x`	URL of the user catalog. Default is the right thing; you shouldn't specify one

Value

a `UserCatalog`

`getTeams` *Retrive all teams you're a member of*

Description

Retrive all teams you're a member of

Usage

```
getTeams()
```

Value

A `TeamCatalog`. Extract an individual team by name. Create a team by assigning in with a new name.

See Also

teams

grouped *Get un(grouped) OrderGroups*

Description

"ungrouped" is a magic OrderGroup that contains all entities not found in groups at a given level of nesting.

Usage

```
grouped(order.obj)

ungrouped(order.obj)
```

Arguments

order.obj	an subclass of ShojiOrder or OrderGroup

Value

For grouped(), an Order/Group, respectively, with "ungrouped" omitted. For ungrouped(), an OrderGroup subclass.

See Also

VariableOrder

hiddenVariables *Show the names of hidden variables within the dataset*

Description

Show the names of hidden variables within the dataset

Usage

```
hiddenVariables(dataset, key = "name")
```

Arguments

dataset	the Dataset
key	the Variable attribute to return. Default is "alias"

Value

a vector of the names of Variables marked as hidden.

hide *Hide and Unhide Variables*

Description

Hide and Unhide Variables

Usage

```
## S4 method for signature 'CrunchVariable'
hide(x)

## S4 method for signature 'VariableCatalog'
hide(x)

## S4 method for signature 'CrunchVariable'
unhide(x)

## S4 method for signature 'VariableCatalog'
unhide(x)
```

Arguments

x	a Variable or subset of a VariableCatalog to hide or unhide

Value

(invisibly) the Variable or VariableCatalog, hidden or unhidden

hideVariables *Hide and Unhide Variables Within a Dataset*

Description

Hide and Unhide Variables Within a Dataset

Usage

```
hideVariables(dataset, variables = NULL, pattern = NULL,
  key = namekey(dataset), ...)

hiddenVariables(x) <- value

unhideVariables(dataset, variables = NULL, pattern = NULL,
  key = namekey(dataset), ...)
```

Arguments

dataset	the Dataset to modify
variables	names or indices of variables to (un)hide
pattern	optional regular expression to identify Variables to (un)hide. Note that this argument is deprecated. If you wish to grep, you can `grep(pattern, aliases(variables(dataset)))` or similar outside this function.
key	the Variable attribute to grep with the pattern. Default is "alias"
...	optional additional arguments to grep, likewise deprecated.
x	same as dataset, for 'hiddenVariables<-'
value	same as variables, for 'hiddenVariables<-'

Value

(invisibly) dataset with the specified variables (un)hidden

See Also

hide

---

http-methods *HTTP methods for communicating with the Crunch API*

Description

HTTP methods for communicating with the Crunch API

Usage

```
crGET(...)

crPUT(...)

crPATCH(...)

crPOST(...)

crDELETE(...)
```

Arguments

...	see crunchAPI for details. url is the first named argument and is required; body is also required for PUT, PATCH, and POST.

Value

Depends on the response status of the HTTP request and any custom handlers.

index *Get the body of a Catalog*

Description

The core of Catalog data is in its "index". These methods get and set that slot.

Usage

```
## S4 method for signature 'ShojiCatalog'
index(x)

## S4 replacement method for signature 'ShojiCatalog'
index(x) <- value
```

Arguments

x	a Catalog (VariableCatalog, Subvariables, or similar object)
value	For the setters, an appropriate-length list to assign

Value

Getters return the list object in the "index" slot; setters return x duly modified.

is-na-categories *is.na for Categories*

Description

is.na for Categories

Usage

```
## S4 method for signature 'Categories'
is.na(x)

## S4 replacement method for signature 'Categories,character'
is.na(x) <- value

## S4 replacement method for signature 'Categories,logical'
is.na(x) <- value

## S4 method for signature 'Category'
is.na(x)

## S4 replacement method for signature 'Category,logical'
is.na(x) <- value
```

Arguments

x	Categories or a single Category
value	To change the missingness of categories, supply either (1) a logical vector of equal length of the categories (or length 1 for the Category method), or (2) the names of the categories to mark as missing. If supplying the latter, any categories already indicated as missing will remain missing.

Value

Getters return logical, a named vector in the case of the Categories method; setters return x duly modified.

---

is.archived,DatasetCatalog-method

Get and set "archived" and "published" status of a dataset

Description

"Archived" datasets are excluded from some views. "Draft" datasets are visible only to editors. "Published" is the inverse of "Draft", i.e. `is.draft(x)` entails `!is.published(x)`. These properties are accessed and set with the "is" methods. The verb functions `archive` and `publish` are alternate versions of the setters (at least in the `TRUE` direction).

Usage

```
## S4 method for signature 'DatasetCatalog'
is.archived(x)

## S4 method for signature 'DatasetCatalog'
is.draft(x)

## S4 method for signature 'DatasetCatalog'
is.published(x)

## S4 replacement method for signature 'DatasetCatalog,logical'
is.archived(x) <- value

## S4 replacement method for signature 'DatasetCatalog,logical'
is.draft(x) <- value

## S4 replacement method for signature 'DatasetCatalog,logical'
is.published(x) <- value

## S4 method for signature 'CrunchDataset'
is.archived(x)
```

```
## S4 method for signature 'CrunchDataset'
is.draft(x)

## S4 method for signature 'CrunchDataset'
is.published(x)

## S4 replacement method for signature 'CrunchDataset,logical'
is.archived(x) <- value

archive(x)

## S4 replacement method for signature 'CrunchDataset,logical'
is.draft(x) <- value

## S4 replacement method for signature 'CrunchDataset,logical'
is.published(x) <- value

publish(x)
```

Arguments

x	CrunchDataset
value	logical

Value

For the getters, the logical value of whether the dataset is archived, in draft mode, or published, where draft and published are inverses. The setters return the dataset.

---

is.dataset *Is it?*

Description

Is it?

Usage

```
is.dataset(x)

is.shoji(x)

is.variable(x)

is.Numeric(x)

is.Categorical(x)
```

```
is.Text(x)

is.Datetime(x)

is.Multiple(x)

is.MR(x)

is.MultipleResponse(x)

is.CA(x)

is.Array(x)

is.CategoricalArray(x)
```

Arguments

x	an object

Value

logical

`is.editor` *Read and set edit privileges*

Description

Read and set edit privileges

Usage

```
## S4 method for signature 'MemberCatalog'
is.editor(x)

## S4 replacement method for signature 'MemberCatalog,logical'
is.editor(x) <- value

## S4 method for signature 'PermissionCatalog'
is.editor(x)

## S4 method for signature 'PermissionTuple'
is.editor(x)
```

Arguments

x	PermissionCatalog or MemberCatalog
value	For the setter, logical: should the indicated users be allowed to edit the associated object?

Value

`is.editor` returns a logical vector corresponding to whether the users in the catalog can edit or not. `is.editor<-` returns the catalog, modified.

---

listDatasets *Show the names of all Crunch datasets*

Description

Show the names of all Crunch datasets

Usage

```
listDatasets(kind = c("active", "all", "archived"), project = NULL,
  refresh = FALSE)
```

Arguments

kind	character specifying whether to look in active, archived, or all datasets. Default is "active", i.e. non-archived.
project	`CrunchProject` entity, character name of a project, or NULL, the default. If a Project entity or reference is supplied, the function will display datasets from that Project's datasets. If NULL, the primary dataset catalog for the user will be used.
refresh	logical: should the function check the Crunch API for new datasets? Default is FALSE.

Value

Character vector of dataset names, each of which would be a valid input for `loadDataset`

loadDataset *Load a Crunch Dataset*

Description

Load a Crunch Dataset

Usage

```
loadDataset(dataset, kind = c("active", "all", "archived"), project = NULL,
  refresh = FALSE)
```

Arguments

dataset	character, the name of a Crunch dataset you have access to. Or, a DatasetTuple.
kind	character specifying whether to look in active, archived, or all datasets. Default is "active", i.e. non-archived.
project	CrunchProject entity, character name of a project, or NULL, the default. If a Project entity or reference is supplied, the function will display datasets from that Project's datasets. If NULL, the primary dataset catalog for the user will be used.
refresh	logical: should the function check the Crunch API for new datasets? Default is FALSE.

Value

An object of class CrunchDataset

lock *Lock and unlock a dataset for editing*

Description

Crunch allows a single active editor. If you have edit privileges but are not currently editing the dataset, you must unlock the dataset before making changes. You may then lock the dataset when you're done editing.

Usage

```
lock(dataset)

unlock(dataset)
```

Arguments

dataset	a CrunchDataset

Value

`dataset`, invisibly, after having set the current editor.

---

login *Authenticate with the Crunch API*

Description

Note that you can store your Crunch account info in your .Rprofile under "crunch.email" and "crunch.pw" for convenience. If you do so, you can simply `login()` to authenticate. For running batch jobs, this could be particularly useful. However, be warned that storing your password in a plain text file such as .Rprofile is a security risk (though perhaps less so than in every .R script you write), and we cannot officially recommend that you do so.

Usage

```
login(email = getOption("crunch.email"), password = getOption("crunch.pw"),
  ...)
```

Arguments

`email`	the email address associated with the user's Crunch account
`password`	the password associated with the user's Crunch account
`...`	additional parameters passed in the authentication. Not currently supported by the Crunch API.

Details

If a password is not supplied (or, if no arguments are supplied and only the `crunch.email` is specified in .Rprofile), and you are in an interactive session, you will be prompted to enter your password. At present, this is the most secure practice as your password is not stored locally.

---

logout *Kill the active Crunch session*

Description

Kill the active Crunch session

Usage

```
logout()
```

makeArray *Make a Categorical Array or Multiple Response variable*

Description

Make a Categorical Array or Multiple Response variable

Usage

```
makeArray(subvariables, dataset = NULL, pattern = NULL,
  key = namekey(dataset), name, ...)

makeMR(subvariables, dataset = NULL, pattern = NULL,
  key = namekey(dataset), name, selections, ...)
```

Arguments

subvariables	a list of Variable objects to bind together, or a Dataset object containing only the Variables to bind (as in from subsetting a Dataset), or values (e.g. names) of variables corresponding to key. If omitted, must supply dataset and pattern. If specifying values, must include dataset.
dataset	the Crunch Dataset to which the variables in subvariables belong, or in which to search for variables based on pattern. If omitted, subvariables must exist and all Variables in the list must belong to the same Dataset
pattern	An optional regular expression to search for variables to bind within dataset. Note that this argument is deprecated. If you wish to grep, you can grep(pattern, aliases(variables(or similar outside this function.
key	character, the name of the Variable field in which to search with pattern. Default is 'alias'.
name	character, the name that the new Categorical Array variable should have. Required.
...	Optional additional attributes to set on the new variable.
selections	character, for makeMR, the names of the categories to mark as the dichotomous selections. Required for makeMR; ignored in makeArray.

Value

A VariableDefinition that when added to a Dataset will create the categorical-array or multiple-response variable.

me *My user entity*

Description

My user entity

Usage

```
me()
```

Value

A UserEntity that corresponds to you, the authenticated user

members<-,CrunchTeam,MemberCatalog-method *Teams*

Description

Teams contain users and datasets. You can share a dataset with a group of users by sharing the dataset with a team. You can also share a bunch of datasets with a user all at once by adding them to a team that has those datasets.

Usage

```
## S4 replacement method for signature 'CrunchTeam,MemberCatalog'
members(x) <- value

## S4 replacement method for signature 'CrunchTeam,character'
members(x) <- value

## S4 method for signature 'CrunchProject'
members(x)

## S4 replacement method for signature 'CrunchProject,MemberCatalog'
members(x) <- value

## S4 replacement method for signature 'CrunchProject,character'
members(x) <- value

## S4 method for signature 'CrunchTeam'
members(x)
```

Arguments

x	a CrunchTeam
value	for members<-, a character vector of emails or URLs of users to add to the team.

Details

These methods allow you to work with teams. Find your teams with the getTeams function, which returns your TeamCatalog. Extract an individual team by name. Create a team by assigning in with a new name, with the assignment value a list, either empty (to just create a team with that name), or with a "members" element, containing emails or URLs of users to add to the team. Users can be added later with the members<- method.

Value

members returns a MemberCatalog, which has references to the users that are members of the team. members<- returns x with the given users added to the members catalog.

See Also

getTeams

---

mergeFork *Merge changes to a dataset from a fork*

Description

Merge changes to a dataset from a fork

Usage

```
mergeFork(dataset, fork, autorollback = TRUE)
```

Arguments

dataset	The CrunchDataset to merge to
fork	The CrunchDataset, perhaps forked from dataset, that is to be merged in.
autorollback	logical If the merge fails, should dataset be restored to its state prior to the merge, or should it be left in its partially merged state for debugging and manual fixing? Default is TRUE, i.e. the former.

Value

dataset with changes from fork merged to it.

na-omit-categories *Omit missing categories*

Description

Omit missing categories

Usage

```
## S4 method for signature 'Categories'
na.omit(object, ...)
```

Arguments

object	Categories
...	additional arguments, ignored

Value

object with any categories that have missing: TRUE excluded

names,BatchCatalog-method *Get and set names, aliases on Catalog-type objects*

Description

These methods let you get and set names and aliases for variables in a Dataset's catalog, or within Subvariables in an array variable. They work like the base R names methods.

Usage

```
## S4 method for signature 'BatchCatalog'
names(x)

## S4 method for signature 'CrunchDataset'
names(x)

## S4 method for signature 'ShojiCatalog'
names(x)

## S4 replacement method for signature 'ShojiCatalog'
names(x) <- value

## S4 method for signature 'ShojiCatalog'
```

```
emails(x)

## S4 method for signature 'Subvariables'
aliases(x)

## S4 replacement method for signature 'Subvariables'
aliases(x) <- value

## S4 method for signature 'CategoricalArrayVariable'
names(x)

## S4 method for signature 'VariableCatalog'
aliases(x)

## S4 replacement method for signature 'VariableCatalog'
aliases(x) <- value

## S4 method for signature 'VariableCatalog'
notes(x)

## S4 replacement method for signature 'VariableCatalog'
notes(x) <- value

## S4 method for signature 'VariableCatalog'
descriptions(x)

## S4 replacement method for signature 'VariableCatalog'
descriptions(x) <- value

## S4 method for signature 'VariableCatalog'
types(x)

## S4 method for signature 'VersionCatalog'
names(x)

## S4 method for signature 'VersionCatalog'
descriptions(x)

## S4 method for signature 'VersionCatalog'
timestamps(x)
```

Arguments

x	a VariableCatalog, Subvariables, or similar object
value	For the setters, an appropriate-length character vector to assign

Details

Note that the names method on a Dataset returns the aliases of its variables by default. This is controlled by getOption("crunch.namekey.dataset"), which is "alias" by default. Set options(crunch.namekey.dataset= if you wish to use variable names. See the vignette on variables for more information.

Value

Getters return the character object in the specified slot; setters return x duly modified.

See Also

Subvariables Categories names vignette("variables", package="crunch")

---

newDataset *Upload a data.frame to Crunch to make a new dataset*

Description

Upload a data.frame to Crunch to make a new dataset

Usage

```
newDataset(x, name = deparse(substitute(x))[1], ...)
```

Arguments

x	a data.frame or other rectangular R object
name	character, the name to give the new Crunch dataset. Default is the name of the R object passed in x
...	additional arguments passed to createDataset

Value

If successful, an object of class CrunchDataset.

newDatasetByColumn *Upload a data.frame column-by-column to make a new dataset*

Description

Use this version if you have lots of variables, under 1M rows, perhaps backed by ff or other memory-mapped files, and time to kill.

Usage

```
newDatasetByColumn(x, name = deparse(substitute(x))[1], ...)
```

Arguments

x	a data.frame or other rectangular R object
name	character, the name to give the new Crunch dataset. Default is the name of the R object passed in x
...	additional arguments passed to createDataset

Value

If successful, an object of class CrunchDataset.

See Also

newDataset newDatasetByCSV

newDatasetByCSV *Upload a data.frame to Crunch to make a new dataset*

Description

This function uses the CSV+JSON import format, which is faster and more effective for certain dataset sizes and shapes than newDatasetByColumn.

Usage

```
newDatasetByCSV(x, name = deparse(substitute(x))[1], ...)
```

Arguments

x	a data.frame or other rectangular R object
name	character, the name to give the new Crunch dataset. Default is the name of the R object passed in x
...	additional arguments passed to createDataset

Value

If successful, an object of class CrunchDataset.

See Also

`newDataset` `newDatasetByColumn`

`newDatasetFromFile` *Upload a file to Crunch to make a new dataset*

Description

Use this import method if you have an SPSS data file. Reading such a file into R as a data.frame will result in lost metadata. You can just send it directly to Crunch and let the server process it.

Usage

```
newDatasetFromFile(file, name = basename(file), ...)
```

Arguments

`file`	character, the path to a file to upload. This should either be a .csv or .sav (SPSS) file.
`name`	character, the name to give the new Crunch dataset. Default is the file name
`...`	additional arguments passed to `createDataset`

Value

On success, an object of class `CrunchDataset`.

`ordering` *Get and set VariableOrder*

Description

The `ordering` methods allow you to get and set a `VariableOrder` on a `CrunchDataset` or on the `VariableCatalog` that the dataset contains.

Usage

```
## S4 method for signature 'CrunchDataset'
ordering(x)

## S4 replacement method for signature 'CrunchDataset'
ordering(x) <- value

## S4 method for signature 'VariableCatalog'
ordering(x)

## S4 replacement method for signature 'VariableCatalog'
ordering(x) <- value

## S4 method for signature 'DatasetCatalog'
ordering(x)

## S4 replacement method for signature 'DatasetCatalog'
ordering(x) <- value
```

Arguments

x	a VariableCatalog or CrunchDataset
value	a valid VariableOrder object

Value

`ordering` returns a VariableOrder object, while `ordering<-` sets the VariableOrder in `value` on `x`

---

owners *See who owns these datasets*

Description

See who owns these datasets

Usage

```
owners(x)

ownerNames(x)
```

Arguments

x	DatasetCatalog

Value

For owners, the URLs of the users or projects that own these datasets. For ownerNames, their names.

---

permissions *See who has access to this dataset*

Description

See who has access to this dataset

Usage

```
## S4 method for signature 'CrunchDataset'
permissions(x)
```

Arguments

x	CrunchDataset

Value

A PermissionCatalog containing information on the users and teams that have access to this dataset.

---

project-icon *A project's icon*

Description

A project's icon

Usage

```
icon(x)

icon(x) <- value
```

Arguments

x	a CrunchProject
value	charcter file path of the icon image file to set

Value

The URL of the project's icon. The setter returns the project after having uploaded the specified file as the new icon.

projects *Get the project catalog*

Description

Get the project catalog

Usage

```
projects(x = getAPIRoot())
```

Arguments

x	a ShojiObject that has a project catalog associated. If omitted, the default value for x means that you will load the user's primary project catalog. (Currently, there are no other project catalogs to load.)

Value

An object of class ProjectCatalog.

Examples

```
## Not run:
myprojects <- projects()
proj <- myprojects[["Project name"]]

## End(Not run)
```

refresh *Get a fresh copy from the server*

Description

Crunch objects usually keep themselves in sync with the server when you manipulate them, but sometimes they can drift. Maybe someone else has modified the dataset you're working on, or maybe you have modified a variable outside of the context of its dataset. refresh() allows you to get back in sync.

Usage

```
## S4 method for signature 'CrunchDataset'
refresh(x)

## S4 method for signature 'ShojiObject'
refresh(x)

## S4 method for signature 'CrunchVariable'
refresh(x)
```

Arguments

x	pretty much any Crunch object

Value

a new version of x

---

restoreVersion *Restore a dataset to a previously saved version*

Description

Restore a dataset to a previously saved version

Usage

```
restoreVersion(dataset, version)
```

Arguments

dataset	a CrunchDataset
version	either the name ("description") of the version to restore to or the integer index of the version, as given by versions(dataset)

Value

dataset, rolled back to version.

See Also

versions saveVersion

saveVersion *Create a new saved version*

Description

Create a new saved version

Usage

```
saveVersion(dataset, description = paste("Version", length(versions(dataset))
  + 1))
```

Arguments

dataset	a CrunchDataset
description	character name to give the saved version, as in a commit message. You are encouraged, though not strictly required, to give versions unique descriptions.

Value

invisibly, the URL of the newly created version

See Also

versions restoreVersion

self *Get the URL of this object*

Description

Get the URL of this object

Usage

```
## S4 method for signature 'ShojiObject'
self(x)

## S4 method for signature 'CrunchVariable'
self(x)
```

Arguments

x	a Crunch object

Value

the URL for x

session *Get various catalogs for your Crunch session*

Description

Get various catalogs for your Crunch session

Usage

```
session()
```

Value

a list.

share *Share a dataset*

Description

Share a dataset

Usage

```
share(dataset, users, edit = FALSE, notify = TRUE)
```

Arguments

dataset	a CrunchDataset
users	character: email address(es) or URLs of the users or teams with whom to share the dataset. If there is no Crunch user associated with an email, an invitation will be sent.
edit	logical: should the specified user(s) be given edit privileges on the dataset? Default is FALSE. edit can be a single value or, if inviting multiple users, a vector of logical values of equal length of the number of emails given.
notify	logical: should users who are getting new privileges on this dataset be sent an email informing them of this fact? Default is TRUE.

Value

Invisibly, the dataset.

See Also

unshare

ShojiObject-class *Mix-in class for multiple inheritance of variables and datasets.*

Description

Exists for common methods in interacting with Crunch API only. Has no Extract methods declared so as not to conflict with the vector/list/data.frame methods jointly inherited in CrunchVariable and CrunchDataset.

ShojiOrder-extract *Extract and update in VariableOrder and VariableGroup*

Description

Extract and update in VariableOrder and VariableGroup

Usage

```
## S4 method for signature 'ShojiOrder,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'ShojiOrder,character,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'ShojiOrder,ANY'
x[[i, j, ...]]

## S4 method for signature 'ShojiOrder,character'
x[[i, j, ...]]

## S4 method for signature 'ShojiOrder'
x$name

## S4 replacement method for signature 'ShojiOrder,character,missing,ShojiOrder'
x[i, j] <- value

## S4 replacement method for signature 'ShojiOrder,ANY,missing,ShojiOrder'
x[i, j] <- value

## S4 replacement method for signature 'ShojiOrder,character,missing,list'
x[[i, j]] <- value

## S4 replacement method for signature 'ShojiOrder,character,missing,character'
x[[i, j]] <- value
```

```
## S4 replacement method for signature 'ShojiOrder,character,missing,OrderGroup'
x[[i, j]] <- value

## S4 replacement method for signature 'ShojiOrder,ANY,missing,OrderGroup'
x[[i, j]] <- value

## S4 replacement method for signature 'ShojiOrder,ANY,missing,ANY'
x[[i, j]] <- value

## S4 replacement method for signature 'ShojiOrder,ANY,missing,`NULL`'
x[[i, j]] <- value

## S4 replacement method for signature 'ShojiOrder,character,missing,`NULL`'
x[[i, j]] <- value

## S4 replacement method for signature 'ShojiOrder,character,missing,ShojiOrder'
x[[i, j]] <- value

## S4 replacement method for signature 'ShojiOrder'
x$name <- value

## S4 method for signature 'OrderGroup,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'OrderGroup,character,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'OrderGroup,character'
x[[i, j, ...]]

## S4 method for signature 'OrderGroup,ANY'
x[[i, j, ...]]

## S4 method for signature 'OrderGroup'
x$name

## S4 replacement method for signature 'OrderGroup,character,missing,list'
x[[i, j]] <- value

## S4 replacement method for signature 'OrderGroup,character,missing,character'
x[[i, j]] <- value

## S4 replacement method for signature 'OrderGroup,character,missing,ShojiOrder'
x[[i, j]] <- value

## S4 replacement method for signature 'OrderGroup,character,missing,OrderGroup'
x[[i, j]] <- value
```

```
## S4 replacement method for signature 'OrderGroup,ANY,missing,OrderGroup'
x[[i, j]] <- value

## S4 replacement method for signature 'OrderGroup,numeric,missing,`NULL`'
x[[i, j]] <- value

## S4 replacement method for signature 'OrderGroup,character,missing,`NULL`'
x[[i, j]] <- value

## S4 replacement method for signature 'OrderGroup'
x$name <- value

## S4 replacement method for signature 'VariableOrder,character,missing,CrunchDataset'
x[[i, j]] <- value

## S4 replacement method for signature 'VariableGroup,character,missing,CrunchDataset'
x[[i, j]] <- value
```

Arguments

x	a VariableOrder or VariableGroup
i	an index. Numeric and logical indexing supported for both classes; character indexing supported for VariableOrder, matching on VariableGroup names
j	Invalid
...	additional arguments
drop	Ignored
name	Same as i but for $
value	For update methods, an object equivalent in class to what is being updated

Value

[[and $ on a VariableOrder return the VariableGroup. [[on VariableGroup returns the entity within, either a character (URL) or nested VariableGroup. [and assignment methods return objects of the same class as x

---

ShojiOrder-length *Length of an Order*

Description

Length of an Order

Usage

```
## S4 method for signature 'ShojiOrder'
length(x)
```

Arguments

x	a ShojiOrder

Value

Integer: the number of elements in the Order

ShojiOrder-slots *Manipulate VariableGroup and VariableOrder*

Description

Manipulate VariableGroup and VariableOrder

Usage

```
## S4 method for signature 'OrderGroup'
entities(x, simplify = FALSE)

## S4 method for signature 'ShojiOrder'
entities(x, simplify = FALSE)

## S4 method for signature 'list'
entities(x, simplify = FALSE)

## S4 replacement method for signature 'OrderGroup'
entities(x) <- value

## S4 replacement method for signature 'ShojiOrder'
entities(x) <- value

## S4 method for signature 'OrderGroup'
name(x)

## S4 replacement method for signature 'OrderGroup'
name(x) <- value

## S4 method for signature 'ShojiOrder'
names(x)

## S4 method for signature 'OrderGroup'
names(x)

## S4 replacement method for signature 'ShojiOrder'
names(x) <- value

## S4 method for signature 'ShojiOrder'
```

```
duplicates(x)

## S4 method for signature 'OrderGroup'
duplicates(x)

## S4 method for signature 'VariableCatalog'
duplicates(x)

## S4 replacement method for signature 'ShojiOrder,logical'
duplicates(x) <- value

## S4 replacement method for signature 'OrderGroup,logical'
duplicates(x) <- value

## S4 replacement method for signature 'VariableCatalog,logical'
duplicates(x) <- value
```

Arguments

x	a VariableGroup or VariableOrder
simplify	logical: should variable URLs inside of groups be flattened or preserved in their nested lists? Default is FALSE.
value	(1) For name, a character (length-1 vector); for names, a character vector of equal length to the number of VariableGroups beind modified; for entities, either a character vector of variable URLs or a list containing a combination of variable URLs and VariableGroups. Note that group names must be unique, should be greater than 0 characters long, and "ungrouped" is a reserved group name. (2) For duplicates, logical for whether duplicate variable entries should be allowed in the VariableOrder.

Value

`entities` returns Variable references and VariableGroups; `names` returns group names; `duplicates` returns logical for whether duplicate variable entries should be allowed

See Also

VariableOrder

grouped

---

show-crunch *Show methods for Crunch objects*

Description

Show methods for Crunch objects

Usage

```
## S4 method for signature 'ShojiObject'
show(object)

## S4 method for signature 'ShojiCatalog'
show(object)

## S4 method for signature 'CrunchVariable'
show(object)

## S4 method for signature 'Category'
show(object)

## S4 method for signature 'Categories'
show(object)

## S4 method for signature 'CrunchExpr'
show(object)

## S4 method for signature 'CrunchLogicalExpr'
show(object)

## S4 method for signature 'CrunchCube'
show(object)
```

Arguments

object	the object

Value

invisibly

See Also

show

Subvariables-class *Subvariables in Array Variables*

Description

Multiple-response and categorical-array variables contain a set of subvariables within them. The Subvariables class encapsulates them.

Usage

```
## S4 method for signature 'CategoricalArrayVariable'
subvariables(x)

## S4 method for signature 'VariableTuple'
subvariables(x)

## S4 replacement method for signature 'CategoricalArrayVariable,ANY'
subvariables(x) <- value

## S4 replacement method for signature 'CategoricalArrayVariable,Subvariables'
subvariables(x) <- value
```

Arguments

x	A Variable or Subvariables object
value	For the setters, the appropriate values to set

Details

Subvariables can be accessed from array variables (including multiple response) with the `subvariables` method. They can be assigned back with the `subvariables<-` setter, but there are limitations to what is supported. Specifically, you can reorder subvariables, but you cannot add or remove subvariables by `subvariables<-` assignment. See `deleteSubvariable` to remove subvariables from an array.

Subvariables have a `names` attribute that can be accessed, showing the display names of the subvariables. These can be set with the `names<-` method.

Finally, subvariables can be accessed as regular (categorical) variables with the `$` and `[[` extract methods.

See the vignette on array variables for further details and examples.

See Also

`subvars-extract` `describe-catalog` `deleteSubvariable` `vignette("array-variables", package="crunch")`

---

subvars-extract *Extract and modify subsets of subvariables*

Description

Extract and modify subsets of subvariables

Usage

```
## S4 method for signature 'Subvariables,character'
x[[i, j, ...]]

## S4 method for signature 'Subvariables,ANY'
x[[i, j, ...]]

## S4 method for signature 'Subvariables,character,ANY'
x[i, j, ..., drop = TRUE]

## S4 replacement method for signature 'Subvariables,character,missing,CrunchVariable'
x[[i]] <- value

## S4 replacement method for signature 'Subvariables,ANY,missing,CrunchVariable'
x[[i]] <- value

## S4 replacement method for signature 'Subvariables,ANY,missing,`NULL`'
x[[i]] <- value

## S4 replacement method for signature 'Subvariables,ANY,missing,ANY'
x[[i]] <- value

## S4 replacement method for signature 'Subvariables,character,missing,Subvariables'
x[i] <- value

## S4 replacement method for signature 'Subvariables,ANY,missing,Subvariables'
x[i] <- value

## S4 replacement method for signature 'Subvariables,ANY,missing,ANY'
x[i] <- value

## S4 method for signature 'CategoricalArrayVariable,character,ANY'
x[i, j, ...,
  drop = TRUE]

## S4 method for signature 'CategoricalArrayVariable,ANY'
x[[i, j, ...]]

## S4 method for signature 'CategoricalArrayVariable,character'
x[[i, j, ...]]

## S4 method for signature 'CategoricalArrayVariable'
x$name

## S4 replacement method for signature 'CategoricalArrayVariable,ANY,missing,ANY'
x[[i]] <- value

## S4 replacement method for signature 'CategoricalArrayVariable,character,missing,ANY'
```

```
x[[i]] <- value

## S4 replacement method for signature 'CategoricalArrayVariable'
x$name <- value
```

Arguments

x	Subvariables or an array Variable (which contains subvariables)
i	which subvariables to extract
j	Invalid
...	additional arguments
drop	Invalid
value	For updating, a CrunchExpr
name	For $, the name (not alias) of the subvariable to extract

Value

A subset of x if extracting, otherwise x duly modified

---

table *Table function for Crunch objects*

Description

Table function for Crunch objects

Usage

```
table(..., exclude, useNA = c("no", "ifany", "always"), dnn, deparse.level)
```

Arguments

...	things to tabulate
exclude	see table
useNA	see table
dnn	see table
deparse.level	see table

Value

a table object

See Also

table

temp.options *Set some global options temporarily*

Description

Set some global options temporarily

Usage

```
temp.options(...)

temp.option(...)
```

Arguments

...	named options to set

Value

an S3 class "contextManager" object

See Also

with-context-manager ContextManager

tojson-crunch *toJSON methods for Crunch objects*

Description

crunch uses the jsonlite package for (de)serialization of JSON. Unlike RJSONIO's toJSON, toJSON does not allow for defining S4 methods for other object types. So, crunch::toJSON wraps jsonprep, which exists to translate objects to base R objects, which jsonlite::toJSON can handle. jsonprep is defined as an S4 generic, and it is exported (unlike codejsonlite::asJSON), so you can define methods for it if you have other objects that you want to successfully serialize to JSON.

Usage

```
jsonprep(x, ...)

## S4 method for signature 'Categories'
jsonprep(x, ...)

## S4 method for signature 'list'
jsonprep(x, ...)
```

```
## S4 method for signature 'ANY'
jsonprep(x, ...)

## S4 method for signature 'ShojiOrder'
jsonprep(x, ...)

## S4 method for signature 'OrderGroup'
jsonprep(x, ...)

toJSON(x, ...)
```

Arguments

x	the object
...	additional arguments

Value

jsonprep returns a base R object that jsonlite::toJSON can handle. toJSON returns the JSON-serialized character object.

See Also

toJSON

---

toVariable *Generic method for converting objects to Crunch representations*

Description

If you have other object types you wish to convert to Crunch variables, you can declare methods for toVariable

Usage

```
toVariable(x, ...)

## S4 method for signature 'character'
toVariable(x, ...)

## S4 method for signature 'numeric'
toVariable(x, ...)

## S4 method for signature 'factor'
toVariable(x, ...)

## S4 method for signature 'Date'
```

```
toVariable(x, ...)

## S4 method for signature 'POSIXt'
toVariable(x, ...)

## S4 method for signature 'VariableDefinition'
toVariable(x, ...)

## S4 method for signature 'logical'
toVariable(x, ...)

## S4 method for signature 'CrunchExpr'
toVariable(x, ...)
```

Arguments

x	the object
...	additional arguments

Value

a list object suitable for POSTing to the Crunch API. See the API documentation for specifications.

---

type *Change the type of Crunch variables*

Description

Numeric, text, and categorical variables can be cast to one another by assigning them a new "type". This modifes the storage of the data on the server and should only be done in narrow circumstances, as in when importing data from a different file format has resulted in incorrect types being specified.

Usage

```
## S4 method for signature 'CrunchVariable'
type(x)

## S4 replacement method for signature 'CrunchVariable'
type(x) <- value
```

Arguments

x	a Variable
value	For the setter, a character value in c("numeric", "text", "categorical")

Value

Getter returns character; setter returns x duly modified.

unbind — *Split an array or multiple-response variable into its CategoricalVariables*

Description

Split an array or multiple-response variable into its CategoricalVariables

Usage

```
unbind(x)
```

Arguments

x	a CategoricalArrayVariable or MultipleResponseVariable

Value

invisibly, the API response from DELETEing the array variable definition. If you `refresh` the corresponding dataset after unbinding, you should see the array variable removed and its subvariables promoted to regular variables.

unshare — *Revoke a user's access to a dataset*

Description

Revoke a user's access to a dataset

Usage

```
unshare(dataset, users)
```

Arguments

dataset	a CrunchDataset
users	character: email address(es) or URLs of the users or teams to unshare with.

Value

Invisibly, the dataset.

See Also

`share`

updateDatasetList *Refresh the local list of Crunch datasets*

Description

Refresh the local list of Crunch datasets

Usage

```
updateDatasetList()
```

Value

Nothing. Called for its side effects of setting local environment variables.

var-categories *Get and set Categories on Variables*

Description

Get and set Categories on Variables

Usage

```
## S4 method for signature 'CrunchVariable'
categories(x)

## S4 method for signature 'CategoricalVariable'
categories(x)

## S4 method for signature 'CategoricalArrayVariable'
categories(x)

## S4 method for signature 'VariableEntity'
categories(x)

## S4 replacement method for signature 'CategoricalVariable,Categories'
categories(x) <- value

## S4 replacement method for signature 'CategoricalArrayVariable,Categories'
categories(x) <- value

## S4 replacement method for signature 'CategoricalVariable,numeric'
categories(x) <- value
```

```
## S4 replacement method for signature 'CategoricalVariable,character'
categories(x) <- value

## S4 replacement method for signature 'CategoricalVariable,ANY'
categories(x) <- value

## S4 replacement method for signature 'CategoricalArrayVariable,numeric'
categories(x) <- value

## S4 replacement method for signature 'CategoricalArrayVariable,character'
categories(x) <- value

## S4 replacement method for signature 'CategoricalArrayVariable,ANY'
categories(x) <- value

## S4 replacement method for signature 'CrunchVariable,ANY'
categories(x) <- value
```

Arguments

x	a Variable
value	for the setters, an object of class Categories to set.

Value

Getters return Categories; setters return x duly modified.

`variable-update` *Updating variables with expressions or values*

Description

Updating variables with expressions or values

Usage

```
## S4 replacement method for signature 'CrunchVariable,ANY,missing,ANY'
x[i, j] <- value

## S4 replacement method for signature 'CrunchVariable,ANY,missing,`NULL`'
x[i, j] <- value

## S4 replacement method for signature 'TextVariable,ANY,missing,character'
x[i, j] <- value

## S4 replacement method for signature 'NumericVariable,ANY,missing,numeric'
x[i, j] <- value
```

```
## S4 replacement method for signature 'DatetimeVariable,ANY,missing,Date'
x[i, j] <- value

## S4 replacement method for signature 'DatetimeVariable,ANY,missing,POSIXt'
x[i, j] <- value

## S4 replacement method for signature 'CrunchVariable,ANY,missing,CrunchExpr'
x[i, j] <- value

## S4 replacement method for signature 'CrunchVariable,CrunchExpr,missing,CrunchExpr'
x[i, j] <- value

## S4 replacement method for signature 'CategoricalVariable,ANY,missing,numeric'
x[i, j] <- value

## S4 replacement method for signature 'CategoricalVariable,ANY,missing,character'
x[i, j] <- value

## S4 replacement method for signature 'CategoricalVariable,ANY,missing,factor'
x[i, j] <- value

## S4 replacement method for signature 'CategoricalArrayVariable,ANY,missing,numeric'
x[i, j] <- value

## S4 replacement method for signature 'CategoricalArrayVariable,ANY,missing,character'
x[i, j] <- value

## S4 replacement method for signature 'CategoricalArrayVariable,ANY,missing,factor'
x[i, j] <- value

## S4 replacement method for signature 'CrunchVariable,ANY,missing,logical'
x[i, j] <- value

## S4 replacement method for signature 'CrunchVariable,ANY'
is.na(x) <- value
```

Arguments

x	a Variable
i	a CrunchLogicalExpr or R index, optionally
j	Invalid
value	an R vector or a CrunchExpr with which to update

Value

x duly modified

VariableCatalog-class *Collection of Variables within a Dataset*

Description

A VariableCatalog contains references to all variables in a dataset, plus some descriptive metadata about each. VariableCatalogs also contain a VariableOrder that governs how variables within it are organized.

VariableDefinition *Construct a variable definition with (optional) additional metadata*

Description

Construct a variable definition with (optional) additional metadata

Usage

```
VariableDefinition(data, ...)

VarDef(data, ...)
```

Arguments

data	an R vector of data to convert to the Crunch payload format. See codetoVariable for how R types are converted. If data is not supplied, you may instead supply values, which will not be converted in any way, nor will extra type information be supplied. Only send values if you know what you're doing. You may also omit both data and values to create an empty variable on the server (all values will be system missing "No Data").
...	additional metadata attributes to send.

Value

a VariableDefinition object, ready to POST to Crunch.

See Also

toVariable

Examples

```
VariableDefinition(rnorm(5), name="Some numbers",
    description="Generated pseudorandomly from the normal distribution")
VarDef(name="Integers", values=1:5, type="numeric",
    description="When creating variable definitions with 'values', you must
    specify 'type', and categorical variables will require 'categories'.")
```

variableMetadata *Get all variable metadata for a dataset*

Description

Get all variable metadata for a dataset

Usage

```
variableMetadata(dataset, parent = FALSE)
```

Arguments

dataset	CrunchDataset
parent	logical: Embed in subvariables a reference to their array parent? Default is FALSE.

Value

A VariableCatalog that has things like categories embedded in each categorical variable, and all subvariables are represented

VariableOrder-class *Organize Variables within a Dataset*

Description

Variables in the Crunch web application can be viewed in an ordered, hierarchical list. These objects and methods allow you to modify that order from R.

Details

A VariableOrder object is a subclass of `list` that contains VariableGroups. VariableGroup objects contain a group name and an set of "entities", which can be variable references or other nested VariableGroups.

Slots

`group` character, the name of the VariableGroup. In the constructor and more generally, this field can be referenced as "name" as well.

`entities` a character vector of variable URLs, or a list containing a combination of variable URLs and VariableGroup objects.

`duplicates` logical: should duplicate variable references be allowed in this object? Default is FALSE.

`vars` either NULL or a VariableCatalog. If not NULL, it will be used to look up variable names from the URLs.

versions *Access the saved versions of a dataset*

Description

Access the saved versions of a dataset

Usage

```
versions(x)
```

Arguments

x	a CrunchDataset

Value

an object of class VersionCatalog. Supported methods on the catalog include "names" and "timestamps".

See Also

saveVersion restoreVersion

weight *Dataset weights*

Description

Dataset weights

Usage

```
weight(x)

weight(x) <- value
```

Arguments

x	a Dataset
value	a Variable to set as weight, or NULL to remove the existing weight

Value

For the getter, a Variable if there is a weight, else NULL. For the setter, x, modified accordingly

with-context-manager *Context manager's "with" method*

Description

Context manager's "with" method

Usage

```
## S3 method for class 'contextManager'
with(data, expr, ...)
```

Arguments

data	contextManager
expr	code to evaluate within that context
...	additional arguments. One additional supported argument is "as", which lets you assign the return of your "enter" function to an object you can access.

Value

Nothing.

See Also

ContextManager

[,CrunchExpr,CrunchLogicalExpr,ANY-method *"Subset" a Variable*

Description

These methods subset variables by creating Expressions, which can be composed and evaluated as needed.

Usage

```
## S4 method for signature 'CrunchExpr,CrunchLogicalExpr,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'CrunchVariable,CrunchExpr,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'CrunchVariable,numeric,ANY'
```

```
x[i, j, ..., drop = TRUE]

## S4 method for signature 'CrunchVariable,logical,ANY'
x[i, j, ..., drop = TRUE]
```

Arguments

x	a Variable
i	a CrunchExpr, logical, or numeric
j	Invalid
...	additional arguments, ignored
drop	Invalid

Value

a CrunchExpr containing references to the variable x and the filter logic contained in i

Index