Package ‘crunch’

September 12, 2018

**Type**   Package

**Title**  Crunch.io Data Tools

**Description**  The Crunch.io service [<http://crunch.io/>](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**  1.24.0


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

**License**  LGPL (>= 3)

**Depends**  R (>= 3.0.0)

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

**Suggests**  covr, crunchy, devtools, haven, httptest (>= 3.0.0), knitr, lintr, magrittr, miniUI, rmarkdown, rstudioapi, shiny, spelling, styler, testthat (>= 2.0.0), withr

**VignetteBuilder**  knitr

**Language**  en-US

**Encoding**  UTF-8

**RoxygenNote**  6.1.0

**LazyData**  true

**Collate**  'AllClasses.R' 'AllGenerics.R' 'variable-definition.R'
'R-to-variable.R' 'SO-survey.R' 'abstract-categories.R'
'abstract-category.R' 'add-subvariable.R' 'add-variable.R'
'api.R' 'append-dataset.R' 'as-data-frame.R' 'auth.R'
'batches.R' 'case-variables.R' 'categories.R' 'category.R'
'change-category-id.R' 'combine-categories.R'
'compare-categories.R' 'compare-datasets.R'
'compare-subvariables.R' 'compare-variables.R' 'compare.R'
R topics documented:

'conditional-transform.R' 'consent.R' 'context-manager.R'
'copy.R' 'crunch-data-frame.R' 'crunch.R' 'crunchbox.R'
'cube-collapse-dimensions.R' 'cube-dims.R' 'cube-index-table.R'
'cube-query.R' 'cube-residuals.R' 'cube-result.R'
'cube-subset.R' 'cube-transforms.R' 'cube-variables.R' 'cut.R'
'dataset-catalog.R' 'dataset-extract.R' 'dataset-order.R'
'dataset-stream.R' 'dataset-update.R' 'dataset.R'
'dichotomize.R' 'drop-rows.R' 'edit-lock.R' 'export-dataset.R'
'expressions.R' 'filters.R' 'folders-projects.R' 'folders.R'
'fork-and-merge.R' 'formula.R' 'gadgets.R' 'geo.R'
'get-datasets.R' 'github.R' 'hide-variables.R' 'insertions.R'
'interact.R' 'join-dataset.R' 'json.R' 'make-array.R'
'members.R' 'merge-crunch-data-frame.R' 'misc.R'
'multitables.R' 'new-dataset.R' 'ordering.R'
'permission-catalog.R' 'pretty-print.R' 'progress.R'
'project-folder.R' 'projects.R' 'search.R' 'session.R'
'share.R' 'shoji-catalog.R' 'shoji-entity.R' 'shoji-folder.R'
'shoji-order-slots.R' 'shoji-order.R' 'shoji.R' 'show.R'
'subtotals-and-headings.R' 'subvariables.R'
'summary-insertions.R' 'tab-book.R' 'teams.R'
'transform-calculate.R' 'transform.R' 'tuple.R' 'univariate.R'
'user-info.R' 'user.R' 'variable-as-methods.R'
'variable-catalog.R' 'variable-delete.R'
'variable-derivation.R' 'variable-folder.R'
'variable-metadata.R' 'variable-order.R' 'variable-summary.R'
'variable-to.R.R' 'variable-type.R' 'variable-update.R'
'variable.R' 'variables.R' 'versions.R' 'weight.R' 'zcl.R'

NeedsCompilation no

Author  Neal Richardson [aut, cre],
         Jonathan Keane [aut],
         Gordon Shotwell [aut]

Maintainer  Neal Richardson <neal@crunch.io>

Repository CRAN

Date/Publication  2018-09-12 21:00:03 UTC

R topics documented:

addGeoMetadata . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
addSubvariable . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
addSummaryStat . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
addVariables . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
appendDataset . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
appendStream . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
as.environment.CrunchDataset-method . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
as.vector.CrunchExpr-method . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
availableGeodataFeatures . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
### R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>batches</td>
<td>14</td>
</tr>
<tr>
<td>c-categories</td>
<td>15</td>
</tr>
<tr>
<td>catalog-dataframes</td>
<td>15</td>
</tr>
<tr>
<td>catalog-extract</td>
<td>17</td>
</tr>
<tr>
<td>catalog-length</td>
<td>20</td>
</tr>
<tr>
<td>Categories-class</td>
<td>21</td>
</tr>
<tr>
<td>categoriesFromLevels</td>
<td>22</td>
</tr>
<tr>
<td>cd</td>
<td>23</td>
</tr>
<tr>
<td>changeCategoryID</td>
<td>24</td>
</tr>
<tr>
<td>cleanseBatches</td>
<td>25</td>
</tr>
<tr>
<td>collapseCategories</td>
<td>26</td>
</tr>
<tr>
<td>combine</td>
<td>26</td>
</tr>
<tr>
<td>compareDatasets</td>
<td>27</td>
</tr>
<tr>
<td>conditionalTransform</td>
<td>28</td>
</tr>
<tr>
<td>consent</td>
<td>30</td>
</tr>
<tr>
<td>ContextManager</td>
<td>30</td>
</tr>
<tr>
<td>copyOrder</td>
<td>31</td>
</tr>
<tr>
<td>copyVariable</td>
<td>32</td>
</tr>
<tr>
<td>createWithPreparedData</td>
<td>33</td>
</tr>
<tr>
<td>crtabs</td>
<td>33</td>
</tr>
<tr>
<td>crunch</td>
<td>34</td>
</tr>
<tr>
<td>crunch-cut</td>
<td>34</td>
</tr>
<tr>
<td>crunch-uni</td>
<td>36</td>
</tr>
<tr>
<td>crunchBox</td>
<td>37</td>
</tr>
<tr>
<td>CrunchDataFrame</td>
<td>38</td>
</tr>
<tr>
<td>CrunchDataset-class</td>
<td>39</td>
</tr>
<tr>
<td>CrunchGeography-class</td>
<td>40</td>
</tr>
<tr>
<td>CrunchVariable-class</td>
<td>41</td>
</tr>
<tr>
<td>cube-methods</td>
<td>41</td>
</tr>
<tr>
<td>cube-missingness</td>
<td>43</td>
</tr>
<tr>
<td>cube-residuals</td>
<td>43</td>
</tr>
<tr>
<td>dashboard</td>
<td>44</td>
</tr>
<tr>
<td>dataset-extract</td>
<td>45</td>
</tr>
<tr>
<td>dataset-owner</td>
<td>46</td>
</tr>
<tr>
<td>dataset-to-R</td>
<td>47</td>
</tr>
<tr>
<td>dataset-update</td>
<td>48</td>
</tr>
<tr>
<td>DatasetOrder-class</td>
<td>50</td>
</tr>
<tr>
<td>datasets</td>
<td>50</td>
</tr>
<tr>
<td>dedupeOrder</td>
<td>51</td>
</tr>
<tr>
<td>delete</td>
<td>52</td>
</tr>
<tr>
<td>deleteDataset</td>
<td>53</td>
</tr>
<tr>
<td>deleteSubvariables</td>
<td>54</td>
</tr>
<tr>
<td>deleteVariables</td>
<td>54</td>
</tr>
<tr>
<td>derivations</td>
<td>55</td>
</tr>
<tr>
<td>describe</td>
<td>57</td>
</tr>
<tr>
<td>dichotomize</td>
<td>59</td>
</tr>
<tr>
<td>dim-dataset</td>
<td>60</td>
</tr>
<tr>
<td>dropRows</td>
<td>61</td>
</tr>
</tbody>
</table>
R topics documented:

duplicated ................................................................. 62
duplicated ................................................................. 62
duplicated ................................................................. 62
duplicated ................................................................. 62
duplicated ................................................................. 62
duplicated ................................................................. 62
duplicated ................................................................. 62
duplicated ................................................................. 62
duplicated ................................................................. 62

e-mail ................................................................. 63
email ................................................................. 63
embedCrunchBox ............................................................. 63
exclusion ................................................................. 64
exportDataset ............................................................... 65
expressions ............................................................... 66
expropriateUser ............................................................. 67
filter-catalog .............................................................. 68
flattenOrder ............................................................... 69
flipArrays ............................................................... 69
folder ................................................................. 70
forkDataset ............................................................... 71
fortify ............................................................... 72
getTeams ............................................................... 72
grouped ............................................................... 73
hiddenVariables ........................................................... 73
hide ............................................................... 74
hideVariables ............................................................. 74
http-methods .............................................................. 75
importMultitable .......................................................... 76
index.table .............................................................. 76
Insertions-class ........................................................... 77
interactVariables ........................................................... 80
is-na-categories ............................................................ 81
is-public ............................................................... 82
is-selected-categories ...................................................... 83
is.archived.DatasetCatalog-method ................................................ 83
is.dataset ............................................................. 85
is.editor ............................................................... 86
joinDatasets .............................................................. 87
listDatasets .............................................................. 88
loadDataset ............................................................. 89
locateEntity ............................................................. 90
lock ............................................................... 90
login ............................................................... 91
logout ............................................................... 91
makeArray .............................................................. 92
makeArrayGadget .......................................................... 92
makeCaseVariable .......................................................... 93
makeMRFromText ........................................................... 94
makeWeight .............................................................. 95
matchCatToFeat ............................................................ 96
me ............................................................... 97
members<-,.CrunchTeam, MemberCatalog-method .......................................... 97
merge ............................................................... 98
mergeFork ............................................................... 99
modifyWeightVariables .................................................... 101
moveToGroup ............................................................. 102
multitable-catalog .......................................................... 103
mv .......................................................... 103
na-omit-categories ......................................................... 105
name,AbstractCategory-method ........................................... 105
names,BatchCatalog-method ............................................... 106
newDataset .......................................................... 108
newFilter .......................................................... 109
newMultitable .......................................................... 110
newProject .......................................................... 111
noTransforms .......................................................... 112
ordering .......................................................... 113
owners .......................................................... 114
pendingStream .......................................................... 114
permissions .......................................................... 115
pk .......................................................... 115
pollProgress .......................................................... 116
population .......................................................... 116
preCrunchBoxCheck .......................................................... 117
prepareDataForCrunch ......................................................... 118
project-icon .......................................................... 118
projects .......................................................... 119
prop.table,CrunchCube-method ............................................... 119
refresh .......................................................... 121
removeEmptyGroups ......................................................... 121
resetPassword .......................................................... 122
restoreVersion .......................................................... 122
rmdir .......................................................... 123
saveVersion .......................................................... 124
scoreCatToFeat .......................................................... 125
searchDatasets .......................................................... 125
self .......................................................... 126
session .......................................................... 126
session-extract .......................................................... 127
setName .......................................................... 127
setNames .......................................................... 128
setOrder .......................................................... 129
settings .......................................................... 130
share .......................................................... 131
shoji-index .......................................................... 131
ShojiObject-class .......................................................... 132
ShojiOrder-extract .......................................................... 132
ShojiOrder-length .......................................................... 135
ShojiOrder-slots .......................................................... 135
show,ShojiFolder-method ......................................................... 137
showTransforms,CrunchCube-method ........................................... 138
SO_schema .......................................................... 139
SO_survey .......................................................... 140
Subtotal-class .......................................................... 142
addGeoMetadata

Add geodata metadata to a crunch variable

Description

If the variable matches a single geographic shapefile hosted by crunch, addGeoMetadata will make the appropriate CrunchGeography to add to a variable’s geo() metadata. It matches based on how well the contents of the variable match the feature properties that are in each shapefile.

Usage

addGeoMetadata(variable, ...)

Index

addGeoMetadata  Add geodata metadata to a crunch variable

Description

If the variable matches a single geographic shapefile hosted by crunch, addGeoMetadata will make the appropriate CrunchGeography to add to a variable’s geo() metadata. It matches based on how well the contents of the variable match the feature properties that are in each shapefile.

Usage

addGeoMetadata(variable, ...)
addSubvariable

Arguments

variable  a Crunch variable to use for matching. This must be either a text or a categorical variable.

...  arguments passed on to `matchCatToFeat()` for example a set of available geographic features as `all_features` if you want to limit the number of features to be considered.

Details

If more than one property of the same geographic shapefile has the same highest matching score, the first one will be used.

If more than one geographic shapefile has the same highest matching score, an error will be printed listing the geographic shapefiles that matched. Information from this error can be used to setup an appropriate `CrunchGeography` by hand to connect a variable with the metadata needed.

Value

a `CrunchGeography` object that can be assigned into `geo(variable)`

Examples

```r
## Not run:
geo(ds$state) <- addGeoMetadata(ds$state)
```

```
## End(Not run)
```
addSummaryStat

Value

variable with the indicated subvariables added.

See Also

subvariables

Examples

## Not run:
d$sallpets <- addSubvariable(d$sallpets, d$sallpets_4)
d$petloc <- addSubvariables(d$petloc, d[c("petloc_school", "petloc_daycare")])

## End(Not run)

addSummaryStat(a CrunchCube)

Description

Use addSummaryStat() to add a summary statistic to a CrunchCube object. If not otherwise specified, the summary statistic will be mean and be placed at the bottom of the cube. You can change those defaults by passing any value you can use with SummaryStat() (e.g. position, categories, after).

Usage

addSummaryStat(cube, stat = c("mean", "median"), var, ...)

Arguments

cube a CrunchCube to add stats to
stat a character with the summary statistic to include (default: "mean")
var a character with the name of the dimension variable to add the summary statistic for generally the alias of the variable in Crunch, but might include Crunch functions like rollup(), bin(), etc.
... options to pass to SummaryStat() (e.g., position, after, etc.)

Value

a CrunchCube with the summary statistic Insertion added to the transforms of the variable specified

See Also

SummaryStat
Examples

```r
## Not run:
pet_feelings
cats dogs
# extremely happy 9 5
# somewhat happy 12 12
# neutral 12 7
# somewhat unhappy 10 10
# extremely unhappy 11 12

# add a mean summary statistic to a CrunchCube
addSummaryStat(pet_feelings, stat = "mean", var = "feelings")
cats dogs
# extremely happy 9 5
# somewhat happy 12 12
# neutral 12 7
# somewhat unhappy 10 10
# extremely unhappy 11 12
# mean 4.90740740740741 4.34782608695652

# we can also store the CrunchCube for use elsewhere
pet_feelings <- addSummaryStat(pet_feelings, stat = "mean", var = "feelings")
pet_feelings
cats dogs
# extremely happy 9 5
# somewhat happy 12 12
# neutral 12 7
# somewhat unhappy 10 10
# extremely unhappy 11 12
# mean 4.90740740740741 4.34782608695652

# \texttt{addSummaryStat} returns a CrunchCube that has had the summary statistic
danced to it, so that you can still use the Crunch logic for multiple
# response variables, missingness, etc.
class(pet_feelings)
[1] "CrunchCube"
attr("package")
[1] "crunch"

# Since \texttt{pet_feelings} is a CrunchCube, although it has similar properties
# and behaviors to arrays, it is not a R array:
is.array(pet_feelings)
[1] FALSE

# cleanup transforms
transforms(pet_feelings) <- NULL

# add a median summary statistic to a CrunchCube
pet_feelings <- addSummaryStat(pet_feelings, stat = "median", var = "feelings")
pet_feelings
```

addSummaryStat
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

... VariableDefinitions or a list of VariableDefinitions.
appendDataset

Value

dataset with the new variables added (invisibly)

Append one Crunch dataset to another

Description

With Crunch, you can add additional rows to a dataset by appending a second dataset to the bottom of the original dataset. Crunch makes intelligent guesses to align the variables between the two datasets and to harmonize the categories and subvariables of variables, as appropriate.

Usage

appendDataset(dataset1, dataset2, autorollback)

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. If it is a Crunch-Dataset, it may be subsetted with a filter expression on the rows and a selection of variables on the columns.

autorollback Deprecated. This option no longer does anything.

Details

Variables are matched between datasets based on their aliases. Variables present in only one of the two datasets are fine; they’re handled by filling in with missing values for the rows corresponding to the dataset where they don’t exist. For variables present in both datasets, you will have best results if you ensure that the two datasets have the same variable names and types, and that their categorical and array variables have consistent categories. To preview how datasets will align when appended, see compareDatasets().

Particularly if you’re appending to datasets that are already shared with others, you may want to use the fork-edit-merge workflow when appending datasets. This allows you to verify your changes before releasing them to the other viewers of the dataset. To do this fork the dataset with forkDataset(), append the new data to the fork, ensure that the append worked as expected, and then merge the fork back to the original dataset with mergeFork(). For more, see vignette("fork-and-merge", package = "crunch").

Value

dataset1, updated with dataset2, potentially filtered on rows and variables, appended to it.
Examples

```r
## Not run:
ds <- loadDataset("Survey, 2016")
new_wave <- loadDataset("Survey, 2017")
ds <- appendDataset(ds, new_wave)

## End(Not run)
```

**Description**

Crunch allows you to stream data to a dataset. Streaming data is useful for datasets which have frequent updates (see the Crunch API documentation for more information). Crunch automatically appends streamed data periodically; however, if you would like to trigger appending pending streamed data to a dataset, you can call `appendStream()`.

**Usage**

```r
appendStream(ds)
```

**Arguments**

- `ds` a CrunchDataset

**Value**

the dataset with pending stream data appended.

---

**Description**

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

**Usage**

```r
## 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

Crunch Variables reside on the server, allowing you to work with datasets that are too big to bring into memory on your machine. Many functions, such as \code{max}, \code{mean}, and \code{crtabs()}, translate your commands into API queries and return only the result. But, not every operation you’ll want to perform has been implemented on the Crunch servers. If you need to do something beyond what is currently supported, you can bring a variable’s data into R with \code{as.vector(ds$var)} and work with it like any other R vector.

Usage

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

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

Arguments

- \code{x} a CrunchVariable
- \code{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 \code{NA} elements. Otherwise, missing categories will all be returned \code{NA}. For non-Categorical variables, the \code{mode} argument is ignored.

Details

\code{as.vector} transfers data from Crunch to a local R session. Note: \code{as.vector} returns the vector in the row order of the dataset. If filters are set that specify an order that is different from the row order of the dataset, the results will ignore that order. If you need the vector ordered in that way, use syntax like \code{as.vector(ds$var)[c(10, 5, 2)]} instead.

Value

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

Get the property features for available geographies

Description

Get the property features for available geographies

Usage

availableGeodataFeatures(x = getAPIRoot(), geodatum_fields = c("name", "description", "location"))

Arguments

x an API root address (default: the R-session default)

geodatum_fields character, what pieces of information about each geodatum should be retained? (default: `c("name", "description", "location")")

Value

a dataframe with all of the available features and geographies for matching

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
S3 method to concatenate Categories and Category objects

Description

S3 method to concatenate Categories and Category objects

Usage

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

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

Arguments

```r
... see c
```

Value

An object of class `Categories`

Examples

```r
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-dataframes

as.data.frame method for catalog objects

Description

This method gives you a view of a catalog, such as a `VariableCatalog`, as a `data.frame` in order to facilitate further exploration.
Usage

```r
## S3 method for class 'VariableCatalog'
as.data.frame(x, row.names = NULL,
               optional = FALSE, keys = c("alias", "name", "type"), ...)

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

## S3 method for class 'BatchCatalog'
as.data.frame(x, row.names = NULL,
               optional = FALSE, keys = c("id", "status"), ...)

## S3 method for class 'FilterCatalog'
as.data.frame(x, row.names = NULL,
               optional = FALSE, keys = c("name", "id", "is_public"), ...)

## S3 method for class 'ProjectCatalog'
as.data.frame(x, row.names = NULL,
               optional = FALSE, keys = c("name", "id", "description"), ...)

## S3 method for class 'UserCatalog'
as.data.frame(x, row.names = NULL,
               optional = FALSE, keys = c("name", "email", "teams", "collaborator"), ...)
```

Arguments

- `x`: A catalog object
- `row.names`: A character vector of elements to use as row labels for the resulting data.frame, or NULL, the default, adds no row labels.
- `optional`: part of as.data.frame signature. Ignored.
- `keys`: A character vector of the catalog attributes that you would like included in the data.frame. To include all attributes, set keys to TRUE, which is the default for some catalogs. Other catalog classes specify a narrower default:
  - VariableCatalog: c("alias", "name", "type")
  - BatchCatalog: c("id", "status")
  - FilterCatalog: c("name", "id", "is_public")
- `...`: Additional arguments passed to data.frame

Details

Modifying the data.frame produced by this function will not update the objects on the Crunch server. Other methods exist for updating the metadata in the variable catalog, for example. See vignette("variables", package = "crunch").
Value

A `data.frame` including metadata about each entity contained in the catalog. The fields in the `data.frame` match the `keys` argument provided to the function, and each row represents an entity.

Examples

```r
## Not run:
ds <- loadDataset("iris")
vars <- variables(ds)
var_df <- as.data.frame(vars, keys = TRUE)
# With row names
as.data.frame(vars, row.names = urls(vars))

## End(Not run)
```

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

Description

Extract and modify subsets of Catalog-type objects

Usage

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

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

## 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 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 'MultitableCatalog,numeric'
x[[i, j, ...]]

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

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

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

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

## S4 method for signature 'ProjectCatalog,numeric'
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]
```r
## 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 'ShojiFolder,numeric'
x[[1, ..., drop = FALSE]]

## S4 method for signature 'ShojiFolder,character'
x[[1, ..., drop = FALSE]]

## 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 'UserCatalog,character,ANY'
x[i, j, ..., drop = TRUE]

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

## S4 method for signature 'VariableCatalog,numeric'
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 (for example secondary a vector of elements to match
against like emails(x) or owners(x) by default this is names(x))
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

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 are themselves subclasses of lists and contain the following fields: "name", "id", "numeric_value", "missing", and optionally "selected".

Usage
Categories(..., data = NULL)
Category(..., data = NULL)

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

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

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

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

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

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

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

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

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

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

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

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

categoriesFromLevels  
> Convert a factor’s levels into Crunch categories.

### Description

Crunch categorical variables have slightly richer metadata than R’s factor variables. This function generates a list of category data from a factor’s levels which can then be further manipulated in R before being imported into Crunch.
cd

Usage

categoriesFromLevels(level_vect)

Arguments

level_vect A character vector containing the levels of a factor. Usually obtained by running 
\texttt{base::levels()}

Value

A list with each category levels id, name, numeric_value, and missingness.

Examples

categoriesFromLevels(levels(iris$Species))

cd

\textit{Change to different folder}

Description

Like \texttt{cd} in a file system, this function takes you to a different folder, given a relative path specification.

Usage

\texttt{cd(x, path, create = FALSE)}

Arguments

\begin{itemize}
  \item \texttt{x} A CrunchDataset or Folder (VariableFolder or ProjectFolder)
  \item \texttt{path} A character "path" to the folder: either a vector of nested folder names or a single string with nested folders separated by a delimiter ("/") default, configurable via \texttt{options(crunch.delimiter)). The path is interpreted as relative to the location of the folder \texttt{x} (when \texttt{x} is a dataset, that means the root, top-level folder). path may also be a Folder object.
  \item \texttt{create} logical: if the folder indicated by \texttt{path} does not exist, should it be created? Default is FALSE. Argument mainly exists for the convenience of \texttt{mv()}, which moves entities to a folder and ensures that the folder exists. You can call \texttt{cd} directly with \texttt{create=TRUE}, though that seems unnatural.
\end{itemize}

Value

A Folder (VariableFolder or ProjectFolder)
See Also

mv() to move entities to a folder; rmdir() to delete a folder; base::setwd() if you literally want to change your working directory in your local file system, which cd() does not do.

Examples

```r
# Not run:
ds <- loadDataset("Example survey")
demo <- cd(ds, "Demographics")
names(demo)
# Or with %>%
require(magrittr)
ds %<>% cd("Demographics") %>%
names()
# Can combine with mv() and move things with relative paths
ds %>%
  cd("Key Performance Indicators/Brand X") %>%
  mv("nps_x", "../Net Promoters")

# End(Not run)
```

**changeCategoryID**

*Change the id of a category for a categorical variable*

**Description**

Changes the id of a category from an existing value to a new one. The variable can be a categorical, categorical array, or multiple response variable. The category changed will have the same numeric value and missing status as before. The one exception to this is if the numeric value is the same as the id, then the new numeric value will be the same as the new id.

**Usage**

```
changeCategoryID(variable, from, to)
```

**Arguments**

- **variable**: the variable in a crunch dataset that will be changed (note: the variable must be categorical, categorical array, or multiple response)
- **from**: the (old) id identifying the category you want to change
- **to**: the (new) id for the category
Details

It is highly recommended to disable any exclusion filter before using changeCategoryID, especially if it is being called multiple times in quick succession (e.g. as part of an automated script). If a problematic exclusion is encountered changeCategoryID will attempt to disable and re-enable the exclusion, but that process will be repeated for every call made which could have adverse consequences (not to mention slow down processing time).

Value

variable with category from and all associated data values mapped to id to

Examples

```r
## Not run:
ds$country <- changeCategoryID(ds$country, 2, 6)

## End(Not run)
```

---

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

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

Arguments

dataset  CrunchDataset

keep  character the statuses that you want to keep. By default, batches that don’t have either "imported" or "appended" status will be deleted.

Value

dataset with the specified batches removed.
collapseCategories   Combine Categories in place

Description

This function allows you to combine the categories of a variable without making a copy of the variable.

Usage

collapseCategories(var, from, to)

Arguments

var      A categorical Crunch variable
from    A character vector of categories you want to combine.
to      A character string with the destination category.

Value

the variable duly modified

See Also

combine()

combine   Combine categories or responses

Description

Crunch allows you to create a new categorical variable by combining the categories of another variable. For instance, you might want to recode a categorical variable with three categories small, medium, and large to one that has just small and large.

Usage

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

**compareDatasets**

**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 copy() for a more natural way to copy variables).

... Additional variable metadata for the new derived variable

**Value**

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

**Examples**

```r
## 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"))))

d$s$pets_owned2 <- combine(ds$allpets, name="Pets owned (collapsed)",
combinations=list(list(name="Mammals", responses=c("Cat", "Dog"))))

## End(Not run)
```

**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**

```r
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

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

## End(Not run)
```

conditionalTransform  Conditional transformation

Description

Create a new variable that has values when specific conditions are met. Conditions are specified using a series of formulas: the left-hand side is the condition that must be true (a `CrunchLogicalExpr`) and the right-hand side is where to get the value if the condition on the left-hand side is true. This is commonly a Crunch variable but may be a string or numeric value, depending on the type of variable you’re constructing.

Usage

```r
conditionalTransform(..., data, else_condition = NA, type = NULL,
    categories = NULL, formulas = NULL)
```
conditionalTransform

Arguments

... a list of conditions to evaluate (as formulas, see Details) as well as other properties to pass to the new conditional variable (i.e. alias, description)

data a Crunch dataset object to use

else_condition a default value to use if none of the conditions are true (default: NA)

type a character that is either "categorical", "text", "numeric" what type of output should be returned? If NULL, the type of the source variable will be used. (default: NULL) The source variables will be converted to this type if necessary.

categories a vector of characters if type="categorical", these are all of the categories that should be in the resulting variable, in the order they should be in the resulting variable or a set of Crunch categories.

formulas a list of conditions to evaluate (as formulas, see Details). If specified, ... must not contain other formulas specifying conditions.

Details

The type of the new variable can depend on the type(s) of the source variable(s). By default (type=NULL), the type of the new variable will be the type of all of the source variables (that is, if all of the source variables are text, the new variable type will be text, if all of the source variables are categorical, the new variable will be categorical). If there are multiple types in the source variables, the result will be a text variable. The default behavior can be overridden by specifying type = "categorical", "text", or "numeric".

conditionalTransform is similar to makeCaseVariable; however, conditionalTransform can use other Crunch variables as a source of a variable, whereas, makeCaseVariable can only use characters. This additional power comes at a cost: makeCaseVariable can be executed entirely on Crunch servers, so no data needs to be downloaded or uploaded to/from the local R session. conditionalTransform on the other hand will download the data necessary to construct the new variable.

Value

a Crunch VariableDefinition

Examples

```r
### Not run:

d$s$cat_opinion <- conditionalTransform(pet1 = 'Cat' - Opinion1,
                                       pet2 = 'Cat' - Opinion2,
                                       pet3 = 'Cat' - Opinion3,
                                       data = ds,
                                       name = "Opinion of Cats")

### End(Not run)
```
consent

Give consent to do things that require permission

Description

Potentially destructive actions require that you confirm that you really want to do them. If you’re running a script and you know that you want to perform those actions, you can preemptively provide consent.

Usage

consent()

with_consent(expr)

Arguments

expr
Code to evaluate with consent

Value

consent returns an S3 class "contextManager" object, which you can use with with. with_consent evaluates its arguments inside the consent context.

See Also

with-context-manager ContextManager

Examples

## Not run:
with(consent(), delete(ds))
# Equivalent to:
with_consent(delete(ds))

## End(Not run)

ContextManager Context managers

Description

Context managers
Usage

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

Arguments

- `enter`: function to run before taking actions
- `exit`: function to run after taking actions
- `error`: optional function to run if an error is thrown
- `as`: character, optional way to specify a default name for assigning the return of the enter function.

Value

an S3 class "contextManager" object

See Also

with-context-manager

---

**copyOrder**

*Copy the variable order from one dataset to another.*

Description

Copy the variable order from one dataset to another.

Usage

```r
copyOrder(source, target)
```

Arguments

- `source`: the dataset you want to copy the order from
- `target`: the dataset you want to copy the order to

Value

returns an object of class `VariableOrder` (which can be assigned to a dataset with `ordering`)

Examples

```r
## Not run:
ordering(ds) <- copyOrder(ds1, ds)
## End(Not run)
```
Description

Makes a copy of a Crunch variable on the server.

Usage

\[
copy\text{variable}(x, \text{deep} = \text{FALSE}, \ldots)
\]

\[
copy(x, \text{deep} = \text{FALSE}, \ldots)
\]

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.
- **\ldots**: 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 is preferable unless a true hard copy is required. Shallow copies are effectively pointers to the original variable, and then 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 independently in the copy, but you can alter Subvariable names and ordering within arrays.

Value

a VariableDefinition for the copied variable. Assign into a Dataset to make the copy happen.
createWithPreparedData

Upload a prepared data.frame with metadata to Crunch

Description
If you have manually created a Crunch dataset object with `prepareDataForCrunch()` this function allows you to upload it to the app.

Usage
`createWithPreparedData(data, metadata = attr(data, "metadata"))`

Arguments
- `data`: a data.frame that meets the Crunch API specification, as returned by `prepareDataForCrunch()`, or a character path to a file or URL where such data has been written as CSV.
- `metadata`: list of Crunch metadata that corresponds to `data`. Default is the "metadata" attribute of `data`, as returned by `prepareDataForCrunch`, or a character path to a file where such metadata has been written as JSON.

Value
A CrunchDataset.

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 side of the "~". If aggregating by functions other than counts, include the aggregation expression on the left-hand side. Compare to `stats::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().

useNA

whether to include missing values in tabular results. See base::table().

Value

an object of class CrunchCube

See Also

weight()

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(“crunch”). To sign up for a Crunch.io account, visit https://app.crunch.io/.

crunch-cut

Cut a numeric Crunch variable

Description

crunch::cut() is equivalent to base::cut() except that it operates on Crunch variables instead of in-memory R objects. The function takes a numeric variable and derives a new categorical variable from it based on the breaks argument. You can either break the variable into evenly spaced categories by specifying the number of breaks, or specify a numeric vector identifying the start and end point of each category. For example, specifying breaks = 5 will break the numeric data into five evenly spaced portions while breaks = c(1, 5, 10) will recode the data into two groups based on whether the numeric vector falls between 1 and 5 or 5 and 10.
Usage

```r
## S4 method for signature 'NumericVariable'
cut(x, breaks, labels = NULL, name,
     include.lowest = FALSE, right = TRUE, dig.lab = 3,
     ordered_result = FALSE, ...)  
```

Arguments

- **x**: A Crunch NumericVariable
- **breaks**: Either a numeric vector of two or more unique cut points or a single number giving the number of intervals into which `x` is to be cut. If specifying cut points, values that are less than the smallest value in `breaks` or greater than the largest value in `breaks` will be marked missing in the resulting categorical variable.
- **labels**: A character vector representing the labels for the levels of the resulting categories. The length of the labels argument should be the same as the number of categories, which is one fewer than the number of breaks. If not specified, labels are constructed using interval notation. For example, `[1, 5)` indicates that the category goes from 1 to 5. The bracket shape indicates whether the boundary value is included in the category, i.e. whether it is "closed". `[1, 5)` indicates that the interval includes (is closed on) 1 but does not include (is open on) 5. If `labels = FALSE`, simple integer codes are returned instead of a factor.
- **name**: The name of the resulting Crunch variable as a character string.
- **include.lowest**: logical, indicating if an `x[i]` equal to the lowest (or highest, for `right = FALSE`) breaks value should be included.
- **right**: logical, indicating if the intervals should be closed on the right (and open on the left) or vice versa.
- **dig.lab**: integer which is used when labels are not given. It determines the number of digits used in formatting the break numbers.
- **ordered_result**: Ignored.
- **...**: further arguments passed to `makeCaseVariable`

Value

a Crunch `VariableDefinition`. Assign it into the dataset to create it as a derived variable on the server.

Examples

```r
## Not run:
ds <- loadDataset("mtcars")
d$s$cat_var <- cut(d$s$mpg, breaks = c(10, 15, 20),
                   labels = c("small", "medium"), name = "Fuel efficiency")
d$s$age <- sample(1:100, 32)
d$s$age4 <- cut(d$s$age, c(0, 30, 45, 65, 100),
                labels = c("Youth", "Adult", "Middle-aged", "Elderly"),
                name = "Age (4 category)"
)
## End(Not run)
Univariate statistics on Crunch objects

Description

Univariate statistics on Crunch objects

Usage

```r
## 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'
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`, a NumericVariable or DatetimeVariable
- `...` additional arguments to summary statistic function
- `na.rm` logical: exclude missings?
See Also

base::mean() stats::sd() stats::median() base::min() base::max()

---

**crunchBox**

**Make a CrunchBox**

**Description**

CrunchBoxes allow you to publish results to the world.

**Usage**

```r
crunchBox(dataset, filters = crunch::filters(dataset),
weight = crunch::weight(dataset), brand_colors, static_colors,
category_color_lookup, ...)
```

```r
CrunchBox(dataset, filters = crunch::filters(dataset),
weight = crunch::weight(dataset), brand_colors, static_colors,
category_color_lookup, ...)
```

**Arguments**

- **dataset** A CrunchDataset, potentially a selection of variables from it
- **filters** FilterCatalog, or NULL for no filters. Default all filters in your catalog, `filters(dataset)`.
- **weight** a CrunchVariable that has been designated as a potential weight variable for `dataset`, or NULL for unweighted results. Default is the currently applied `weight()`.
- **brand_colors** an optional color vector of length 3 or less, or a named list with names 'primary', 'secondary', and 'message'. See "Details" for more about color specification.
- **static_colors** an optional vector of colors to use for categorical plots. Bars and lines are colored in the order of `static_colors`. See "Details" for more about color specification.
- **category_color_lookup** an optional list of category names to colors to use for that category, wherever it appears in the data. This allows you to always see a category displayed in a specific color. See "Details" for more about color specification.
- **...** additional metadata for the box, such as "title", "header", etc.

**Details**

In addition to specifying the variables and filters to include in your CrunchBox, you can provide custom color palettes. The arguments `brand_colors`, `static_colors`, and `category_color_lookup` allow you to provide color lists to use. Colors should be either a valid hexadecimal string representation, like "#fa1af1", or they may also be an R named color, such as "darkgreen".
The URL to the newly created box.

See Also

`preCrunchBoxCheck()` to provide guidance on what you’re including in the

Examples

```r
## Not run:
# Creating a CrunchBox with three variables
crunchBox(ds[c("var1", "var2", "var3")], title="New CrunchBox")

# Creating a CrunchBox changing primary, secondary, and message brand colors
crunchBox(ds[c("var1", "var2", "var3")],
title="Branded CrunchBox",
brand_colors = c("#ff0aa4", "#af17ff", "#260aff"))

# Creating a CrunchBox changing category-specific colors
crunchBox(ds[c("var1", "var2", "var3")],
title="CrunchBox with category colors",
category_color_lookup = list("agree" = "#ff0aa4",
                           "disagree" = "#af17ff",
                           "don't know" = "#260aff"))

## End(Not run)
```

CrunchDataFrames are designed to mimic the ways that `data.frame`s are used. They should be a drop-in replacement in many places where `data.frame`s are used.

Usage

```r
## S3 method for class 'CrunchDataFrame'
dim(x)

## S3 method for class 'CrunchDataFrame'
names(x)

## S3 method for class 'CrunchDataFrame'
x[i, j, drop = TRUE]
```
Crunch Datasets

Description

Crunch Datasets

Arguments

x: a CrunchDataFrame
i: indicators for rows (integers or logicals)
j: indicators for columns (names, integers, or logicals)
drop: logical. If TRUE the result is coerced to the lowest possible dimension. The default is to drop if only one column is left, but not to drop if only one row is left.
value: value to place (or replace) in the CrunchDataFrame

Details

CrunchDataFrames are generated not by downloading all of the variables from a dataset, but rather only the variables that are needed by subsequent functions. So, if you create a CrunchDataFrame, and then run a linear model using `lm()`, only the variables used by the linear model will be downloaded.

CrunchDataFrames can be altered (that is: adding more columns, removing columns, subsetting rows, etc.) with the same `[`, `[[`, and `$` syntax as data.frames.
Crunch stores geographic data as variable metadata. There are a number of functions that help access and change this metadata.

### Usage

```r
CrunchGeography(..., data = NULL)
```

```r
## S4 method for signature 'CrunchVariable'
geo(x)
```

```r
## S4 replacement method for signature 'CrunchVariable,CrunchGeography'
geo(x) <- value
```

```r
## S4 replacement method for signature 'CrunchVariable,'NULL''
geo(x) <- value
```

```r
availableGeodata(x = getAPIRoot())
```

### Arguments

- `...` for CrunchGeography, named arguments from which to construct a CrunchGeography:
  - `geodatum` an object of class CrunchGeodata which stores references to the Crunch-hosted (geo|topo)json to use
  - `feature_key` a character string representing the feature inside of the (geo|topo)json which is used to match `match_field` (e.g. properties.name)
  - `match_field` a character string representing the variable metadata information which is used to match `feature_key` to (e.g. name)
- `data` for CrunchGeography, list of named arguments from which to construct a CrunchGeography:
  - `geodatum` an object of class CrunchGeodata which stores references to the Crunch-hosted (geo|topo)json to use
  - `feature_key` a character string representing the feature inside of the (geo|topo)json which is used to match `match_field` (e.g. properties.name)
  - `match_field` a character string representing the variable metadata information which is used to match `feature_key` to (e.g. name)
- `x` a crunch variable
- `value` value of the geography property to set

### Details

`geo` retrieves the geographic information associate with a variable. If there is geographic information it returns an object of class CrunchGeography otherwise it returns NULL.

CrunchGeography objects store geography metadata from a variable. There are three slots:

- `geodatum` an object of class CrunchGeodata which stores references to the Crunch-hosted (geo|topo)json to use
- `feature_key` a character string representing the feature inside of the (geo|topo)json which is used to match `match_field` (e.g. properties.name)
- `match_field` a character string representing the variable metadata information which is used to match `feature_key` to (e.g. name)
CrunchVariable-class

Value

georaphic information of class CrunchGeography (NULL if there is none)

Examples

```r
## Not run:
geo(ds$location)

geo(ds$location)$feature_key <- "properties.name"
geo(ds$location)$match_field <- "name"

## End(Not run)
```

__CrunchVariable-class__  __Variables in Crunch__

Description

Variables are S4 objects. All inherit from the base class CrunchVariable.

Slots

- `filter` either NULL or CrunchLogicalExpr
- `tuple` VariableTuple

__cube-methods__  __Methods on Cube objects__

Description

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

Usage

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

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

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

## S4 method for signature 'CrunchCube'
dimensions(x)
```
## S4 replacement method for signature 'CrunchCube,CubeDims'

dimensions(x) <- value

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

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

## S4 method for signature 'CrunchCube'
dimnames(x)

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

## S4 method for signature 'CrunchCube'
measures(x)

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

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

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

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

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

### Arguments

- **x**: a CrunchCube or its CubeDims component.
- **value**: for dimensions<- a CubeDims object to overwrite a CrunchCube dimensions
- **i**: used with [ to extract a dimension
- **j**: not used
- **...**: not used
- **drop**: not used

### Value

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

cube-computing base::array

cube-missingness Modify cube missing behavior

Description

By default, CrunchCubes do not show entries for missing categories. You can include missing values in a cube with showMissing(cube) and hide them again with hideMissing(cube).

Usage

```r
## S4 method for signature 'CrunchCube'
showMissing(cube)

## S4 method for signature 'CrunchCube'
hideMissing(cube)

## S4 method for signature 'CrunchCube'
showIfAny(cube)
```

Arguments

cube a CrunchCube

---

cube-residuals Calculate standardized residuals from a CrunchCube

Description

Standardized residuals, \( \frac{\text{observed} - \text{expected}}{\sqrt{V}} \), where \( V \) is the residual cell variance (Agresti, 2007, section 2.4.5). Special care is taken for multiple-response variables which are in effect a series of separate tables where ‘not selected’ cells for each item are hidden.

Usage

```r
## S4 method for signature 'CrunchCube'
rstandard(model)
```

Arguments

model A CrunchCube representing a contingency table
an array of standardized residuals or Z-scores from the hypothesis being tested. The default method is that the joint distributions of (weighted) counts are equal to the marginal distributions of the table.

References

See Also
stats::chisq.test

dashboard View or set a dashboard URL

Description
You can designate a dashboard that will show when the dataset is loaded in the Crunch web app. This dashboard could be a Crunch Shiny ("Crunchy") app, a CrunchBox, an RMarkdown website or something else.

Usage
dashboard(x)

setDashboardURL(x, value)
dashboard(x) <- value

Arguments
x CrunchDataset
value For the setter, a URL (character) or NULL to unset the dashboard.

Value
The getter returns a URL (character) or NULL. The setter returns the dataset (x).

Examples
## Not run:
dashboard(ds) <- "https://shiny.crunch.io/example/"

## End(Not run)
**dataset-extract**

**Subset datasets and extract variables**

**Description**

Subset datasets and extract variables

**Usage**

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

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

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

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

## S4 method for signature 'CrunchDataset,VariableOrder,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.
2. If j is specified as either x[i, j] or x[i, j], i is an object of class CrunchLogicalExpr that will define a subset of rows.

j columnar extraction, as described above
...
additional arguments
drop logical: automatically 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

Description

Get and set 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.
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 stats::lm()) without needing to download the whole dataset. You can, however, choose to download a true data.frame.

Usage

```r
## S3 method for class 'CrunchDataset'
as.data.frame(x, row.names = NULL,
               optional = FALSE, force = FALSE, categorical.mode = "factor",
               row.order = NULL, include.hidden = TRUE, ...)

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

Arguments

- `x`: a CrunchDataset or CrunchDataFrame
- `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.
- `categorical.mode`: what mode should categoricals be pulled as? One of factor, numeric, id (default: factor)
- `row.order`: vector of indices. Which, and their order, of the rows of the dataset should be presented as (default: NULL). If NULL, then the Crunch Dataset order will be used.
- `include.hidden`: logical: should hidden variables be included? (default: TRUE)
- `...`: additional arguments passed to as.data.frame (default method).

Details

By default, the as.data.frame method for CrunchDataset does not return a data.frame but instead CrunchDataFrame, which behaves like a data.frame without bringing the whole dataset into memory. When you access the variables of a CrunchDataFrame, you get an R vector, rather than a CrunchVariable. 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.
If you call `as.data.frame()` on a CrunchDataset with `force = TRUE`, you will instead get a true data.frame. You can also get this data.frame by calling `as.data.frame()` on a CrunchDataFrame (effectively calling `as.data.frame()` on the dataset twice).

When a data.frame is returned, the function coerces Crunch Variable values into their R equivalents using the following rules:

- Numeric variables become numeric vectors
- Text variables become character vectors
- Datetime variables become either Date or POSIXt vectors
- Categorical variables become either factors with levels matching the Crunch Variable's categories (the default), or, if `categorical.mode` is specified as "id" or "numeric", a numeric vector of category ids or numeric values, respectively
- Array variables (Categorical Array, Multiple Response) are decomposed into their constituent categorical subvariables. An array with three subvariables, for example, will result in three columns in the data.frame.

Column names in the data.frame are the variable/subvariable aliases.

Value

When called on a CrunchDataset, the method returns an object of class CrunchDataFrame unless `force = TRUE`, in which case the return is a data.frame. For CrunchDataFrame, the method returns a data.frame.

See Also

`as.vector()`

---

dataset-update

Update a variable or variables in a dataset

Description

Update a variable or variables in a dataset

Usage

```r
## 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
```
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 ex-
           traction, columns of the dataset rather than rows.

Value

x, modified.
DatasetOrder-class  Organize Datasets

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

Crunch datasets are associated with catalogs. A project catalog will have a set of datasets associated with it, as will a user or team. This function allows you to get or modify the datasets associated with a catalog.

Usage

datasets(x = getAPIRoot())

datasets(x) <- value

Arguments

x a ShojiObject, such as a CrunchProject. If omitted, the function 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

```r
## 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)
```

### dedupeOrder

**Remove duplicated entities from an order/group**

**Description**

This function recurses through a ShojiOrder or OrderGroup and removes any duplicate entities. As with `base::duplicated`, the first appearance of an entity is kept, and subsequent occurrences are marked as duplicated and removed. (Unlike duplicated, there is no option to reverse that order.) The first occurrence of an entity is determined by the function’s recursion: within each group, nested groups are processed first, in order, and their nested groups are processed recursively. See the test suite, in test-variable-order.R, for an example that illustrates which entities are dropped as duplicate.

**Usage**

```r
dedupeOrder(x)
```

**Arguments**

- `x` VariableOrder, DatasetOrder, VariableGroup, or DatasetGroup

**Value**

- `x` with duplicate entities removed.

**See Also**

`duplicates`, which when set to FALSE also calls this function.
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

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

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

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

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

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

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

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

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

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

Arguments

```
x       a Crunch object
...
    additional arguments, in the generic
```

Details

Deleting requires confirmation. In an interactive session, you will be asked to confirm. To avoid that prompt, or to delete objects from a non-interactive session, wrap the call in `with_consent()` to give your permission to delete.
deleteDataset

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 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

Deleting variables requires confirmation. In an interactive session, you will be asked to confirm. To avoid that prompt, or to delete subvariables from a non-interactive session, wrap the call in `with_consent()` to give your permission to delete.

Usage

deleteSubvariables(variable, to.delete)

deleteSubvariable(variable, to.delete)

Arguments

variable  the array variable

to.delete  aliases (following `crunch.namekey.dataset`) or indices of variables to delete.

Details

To delete the subvariables the function unbinds the array, deletes the subvariable, and then binds the remaining subvariables into a new array.

Value

a new version of variable without the indicated subvariables

deleteVariables  Delete Variables Within a Dataset

Description

This function permanently deletes a variable from a dataset. For a non-destructive alternative see `hide()`.

Usage

deleteVariables(dataset, variables)

deleteVariable(dataset, variables)
Arguments

- **dataset**: the Dataset to modify
- **variables**: aliases (following `crunch.namekey.dataset`) or indices of variables to delete.

Details

In an interactive session, you will be prompted to confirm that you wish to delete the variable. To avoid that prompt, or to delete variables from a non-interactive session, wrap the call in `with_consent()` to give your permission to delete.

Value

(invisibly) dataset with the specified variables deleted

See Also

`hide`

---

**derivations**

*Get or set a derived variable’s expression*

Description

Get a derived variable’s derivation formula as a `CrunchExpr` with `derivation(variable)`. Set (change) a derived variable’s derivation with `derivation(variable) <- expression`.

Usage

```r
## S4 method for signature 'CrunchVariable'
derivation(x)

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

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

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

## S4 replacement method for signature 'CrunchVariable,logical'
is.derived(x) <- value
```
Arguments

- **x**: a variable
- **value**: a CrunchExpr to be used as the derivation (for the setter only) or NULL to integrate a derived variable. For is.derived, FALSE can be used to integrate a derived variable.

Details

To break a derivation link between a derived variable and the originating variable, set the derivation value of the derived variable to NULL with derivation(variable) <- NULL.

is.derived can be used to see if a variable is derived or not. Additionally setting a derived variable’s is.derived to FALSE will break the derivation link between two variables.

Value

- a CrunchExpr of the derivation for derivation; a logical for is.derived; the variable given in x for is.derived <- returns

Examples

```r
## Not run:

d$sderived_v1 <- d$V1 + 5

derivation(d$sderived_v1)
# Crunch expression: V1 + 5

derivation(d$sderived_v1) <- d$V1 + 10
derivation(d$sderived_v1)
# Crunch expression: V1 + 10

is.derived(d$sderived_v1)
# TRUE

# to integrate or instantiate the variable in place (remove the link between
# variable V1 and the derivation) you can:
derivation(d$sderived_v1) <- NULL

# after integrating, the derived variable is no longer derived.
is.derived(d$sderived_v1)
# FALSE

## End(Not run)
```
**Description**

Name, alias, and description for Crunch objects

**Usage**

```r
## 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 'Geodata'
description(x)

## S4 replacement method for signature 'Multitable'
name(x) <- value
```
```r
## S4 replacement method for signature 'ShojiFolder'
name(x) <- value

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

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

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

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

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

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

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

## 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

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

## S4 replacement method for signature 'NumericVariable'
digits(x) <- value
```
dichotomize

## S4 replacement method for signature 'CrunchVariable'

digits(x) <- value

## S4 method for signature 'MultipleResponseVariable'

uniformBasis(x)

## S4 replacement method for signature 'MultipleResponseVariable'

uniformBasis(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 and digits are only defined for Variables.

### Value

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

### See Also

- Categories describe-catalog

---

### Description

Multiple Response variables are essentially Categorical Arrays in which one or more categories are set as "selected". These methods allow 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)
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 does the reverse.

Value

Categories or the Variable, (un)dichotomized accordingly

See Also

describe-category

dim-dataset

Dataset dimensions

Description

Dataset dimensions
dropRows

Usage

```r
## 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

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

---

duplicated "duplicated" method for Crunch objects

Description

"duplicated" method for Crunch objects

Usage

```r
# S4 method for signature 'CrunchVariable'
duplicated(x, incomparables = FALSE, ...)
```

```r
# S4 method for signature 'CrunchExpr'
duplicated(x, incomparables = FALSE, ...)
```

Arguments

- `x` CrunchVariable or CrunchExpr
- `incomparables` Ignored
- `...` Ignored

Value

A CrunchLogicalExpr that evaluates TRUE for all repeated entries after the first occurrence of a value.

See Also

base::duplicated()
Extract the email from a User Entity

**Description**

Extract the email from a User Entity

**Usage**

```
email(x)
```

```R
## S4 method for signature 'UserEntity'
email(x)
```

**Arguments**

- `x`: a UserEntity returned from `me()`

**Value**

a character string of the user's email

---

Get HTML for embedding a CrunchBox

**Description**

`crunchBox()` returns a URL to the box data that it generates, but in order to view it in a CrunchBox or to embed it on a website, you'll need to translate that to the Box’s public URL and wrap it in some HTML. This function takes a CrunchBox and returns the HTML which you can embed in a website.

**Usage**

```
embedCrunchBox(box, title = NULL, logo = NULL, ...)
```

**Arguments**

- `box`: character URL of the box data, as returned by `crunchBox()`
- `title`: character title for the Box, to appear above the iframe. Default is NULL, meaning no title shown
- `logo`: character URL of a logo to show instead of a title. Default is NULL, meaning no logo shown. If both logo and title are provided, only the logo will be shown. Note also that logo must be a URL of a hosted image: it cannot be a path to a local file.
- `...`: Additional arguments, not currently used.
exclusion

Value

Prints the HTML markup to the screen and also returns it invisibly.

See Also

crunchBox()

Examples

```r
## Not run:
box <- crunchBox(ds)
embedCrunchBox(box, logo="/myco.example/img/logo_200px.png")

## End(Not run)
```

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

```r
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

This function allows you to write a CrunchDataset to a .csv or SPSS .sav file.

Usage

```
exportDataset(dataset, file, format = c("csv", "spss"),
               categorical = c("name", "id"), na = NULL, varlabel = c("name",
                                                                        "description"), include.hidden = FALSE, ...)
```

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

Arguments

- `dataset` CrunchDataset, which may have been subsetted with a filter expression on the rows and a selection of variables on the columns.
- `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.
- `na` Similar to the argument in `utils::write.table()`, 'na' lets you control how missing values are written into the CSV file. Supported values are:
  1. NULL, the default, which means that categorical variables will have the category name or id as the value, and numeric, text, and datetime variables will have the missing reason string;
  2. A string to use for missing values.
  3. "" means that empty cells will be written for missing values for all types.
- `varlabel` For SPSS export, which Crunch metadata field should be used as variable labels? Default is "name", but "description" is another valid value.
- `include.hidden` logical: should hidden variables be included? (default: FALSE)
- `x` (for write.csv) CrunchDataset, which may have been subsetted with a filter expression on the rows and a selection of variables on the columns.

Value

Invisibly, file.
Crunch Expressions, i.e. CrunchExpr and CrunchLogicalExpr, encapsulate 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

```r
## 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,numeric'
e1 == e2

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

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

- `x`: an input
- `table`: For `%in%`. See `base::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`: for the `rollupResolution()` setter, the default resolution for this variable.

**Value**

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

---

**Description**

If you want to transfer all teams, projects, and datasets owned by one user to another you can with `expropriateUser`. To use `expropriateUser` you must be an account admin and be from the same account as the user who is being expropriated. This is useful if a user leaves your organization and you want to transfer all of the teams, projects, and datasets they own to someone else.
**Usage**

```r
expropriateUser(from, to)
```

**Arguments**

- `from` | a character of the email address of the user to expropriate from
- `to` | a character of the email address of the user who should be the new owner

**Details**

Expropriating requires confirmation. In an interactive session, you will be asked to confirm. To avoid that prompt, or to expropriate datasets from a non-interactive session, wrap the call in `with_consent()` to give your permission to expropriate.

**Value**

None

---

**filter-catalog**

*Get or set a dataset’s filters*

**Description**

You can build and save filters in the Crunch web app, and these filters are stored in a `FilterCatalog`. This function allows you to retrieve and modify those filters.

**Usage**

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

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

**Arguments**

- `x` | a `CrunchDataset`
- `value` | for the setter, a `FilterCatalog`

**Value**

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

Remove nesting of groups within an order/group

Description

This function reduces a potentially nested order to its flattened representation, containing no nested groups. Entities are ordered in the result by their first appearance in the order object.

Usage

flattenOrder(x)

Arguments

x: VariableOrder, DatasetOrder, VariableGroup, or DatasetGroup; or a Crunch-Dataset or catalog that has an ordering property.

Value

x, or its order resource, flattened.

flipArrays

Rearrange array subvariables

Description

Sometimes it is useful to group subvariables across arrays in order to compare them more easily. This function generates a set of derived views of common subvariables across arrays. Because they are derived, they share data with the underlying array variables, and they are thus automatically updated when new data is appended.

Usage

flipArrays(variables, suffix = "", flipped")

Arguments

variables: List of variables, a variable catalog, or a dataset subset containing the categorical array or multiple response variables you want to rearrange.
suffix: character string to append to the new variable names. Pass "" if you don’t want it to append anything.

Value

A list of derived VariableDefinitions, one per unique subvariable name across all variables. Each variable in variables that contains this subvariable will appear as a subvariable in these new derived array definitions. Use addVariables to add these to your dataset.
Examples

```r
## Not run:
ds <- addVariables(ds, flipArrays(ds[c("petloc", "petloc2")], suffix="", rearranged"))
## End(Not run)
```

---

**folder**

*Find and move entities to a new folder*

**Description**

Find and move entities to a new folder

**Usage**

```r
folder(x)
folder(x) <- value
```

**Arguments**

- `x` For `folder`, a Variable to find. For `folder` assignment, a Variable, selection of variables in a Dataset, or any other object that can be moved to a folder.
- `value` For assignment, a character "path" to the folder: either a vector of nested folder names or a single string with nested folders separated by a delimiter ("/") default

**Value**

`folder` returns the parent folder of `x`, or `NULL` if the `x` is the root level. `folder<-` returns the `x` input, having been moved to the requested location.

**See Also**

`mv()` `cd()`

**Examples**

```r
## Not run:
ds <- loadDataset("Example survey")
folder(ds$income) <- "Demographics/Economic"
folder(ds$income)
## [1] "Demographics" "Economic"

## End(Not run)
```
for Dataset

Create a fork of a dataset

Description

Forking a dataset makes a copy of the data that is linked by Crunch’s version control system to the original dataset. When you make edits to a fork, users of the original dataset do not see the changes.

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, visible only to those with edit permissions? Default is FALSE.
- ... Additional dataset metadata to provide to the fork

Details

A common strategy for revising a dataset that has been shared with others is to fork it, make changes to the fork, and then merge those changes back into the original dataset. This workflow allows you to edit a dataset and review changes before publishing them, so that you don’t accidentally send your clients incorrect data. For more on this workflow, see vignette(“fork-and-merge”, package = “crunch”).

Value

The new fork, a CrunchDataset.

See Also

```
mergeFork()
```
fortify

Fortify crunch objects for use with ggplot

Description

Fortify crunch objects for use with ggplot

Usage

fortify.CrunchDataFrame(model, data, ...)

fortify.CrunchDataset(model, data, ...)

Arguments

model  model or other R object to convert to data frame
data   original dataset, if needed
...    other arguments passed to methods

getTeams

Retrieve your teams

Description

Retrieve your teams

Usage

getTeams()

Value

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

See Also

tools
grouped

Get grouped or ungrouped OrderGroups

Description
"ungrouped" is an 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 a dataset’s hidden variables

Description

Show the names of a dataset’s hidden variables

Usage

hiddenVariables(dataset, key = namekey(dataset))

Arguments

dataset  the Dataset

key       the Variable attribute to return. Default is "alias", following getOption("crunch.namekey.dataset").

Value

a vector of the names of Variables marked as hidden.
**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

**See Also**

- `hideVariables`

---

**Description**

Hide and unhide variables within a dataset

**Usage**

```
hideVariables(dataset, variables)
hiddenVariables(x) <- value
unhideVariables(dataset, variables)
```
Arguments

- **dataset**: the Dataset to modify
- **variables**: names or indices of variables to (un)hide
- **x**: same as dataset, for hidden variables
- **value**: same as variables, for hidden variables

Value

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

See Also

- **hide**

---

**http-methods**

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

Description

These methods let you communicate with the Crunch API, for more background see Crunch Internals.

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.
importMultitable  Import a Multitable

Description

Allows you to import a multitable from a different dataset

Usage

importMultitable(data, multitable, ...)

Arguments

data a CrunchDataset in which to create the multitable
multitable an object of class Multitable that you want copied to the new dataset
... Additional multitable attributes to set. Options include name and is_public.

Value

An object of class Multitable

Examples

## Not run:
m <- newMultitable(~ gender + age4 + marstat, data=ds)
copied_m <- importMultitable(another_ds, m)
name(copied_m) # [1] "gender + age4 + marstat"

## End(Not run)

index.table  Calculate an index table for a CrunchCube

Description

Index tables are percentages of percentages. They take the percentage from prop.table(cube, margin) and, by default, divide that by the proportions of the other margin. The baseline argument can be used to provide baseline proportions to compare against.

Usage

index.table(x, margin, baseline)
Arguments

- **x**: A CrunchCube to calculate index table for
- **margin**: which margin to index against (1 for rows, 2 for columns)
- **baseline**: an arbitrary set of proportions to compare the table given in x to. Useful for comparing two separate cubes. baseline must have the same length as the extent of the dimension given in margin.

Details

`index.table()` is only implemented for 2 dimensional cubes. If you need to calculate indexes for a higher dimension Cube, please slice the cube first.

Value

an array of percentages indexed to the margin provided

Examples

```r
## Not run:
cube_object
# v7
# v4  C  E
# B 5 2
# C 5 3
index.table(cube_object, 1)
# v7
# v4  C  E
# B 107.1429 85.71429
# C 93.7500 112.50000
index.table(cube_object, 2)
# v7
# v4  C  E
# B 100 80
# C 100 120
index.table(cube_object, 2, c(0.6, 0.4))
# v7
# v4  C  E
# B 83.33333 66.66667
# C 125.00000 150.00000

## End(Not run)
```
Description

Insertions allow you to insert new categories into a categorical-like response on a variable’s transformations.

Usage

Insertions(..., data = NULL)

Insertion(...)

.insertion(..., data = NULL)

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

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

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

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

## S4 replacement method for signature 'Insertion,ANY'
subtotals(x) <- value

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

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

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

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

## S4 method for signature 'Insertion'
arguments(x)

## S4 method for signature 'Subtotal'
arguments(x, var_categories)

## S4 method for signature 'Heading'
arguments(x)
Arguments

... additional arguments to \( \_ \), ignored

data For the constructor functions Insertion and Insertions, 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 Insertion or Insertions
value For \( [<-, \) the replacement Insertion to insert

var_categories categories (from \texttt{categories()}) to used by the arguments and anchor methods when needed to translate between category names and category ids.

Working with Insertions

Insertions are used to add information about a variable or CrunchCube that extends the data in the dataset but does not alter it. This new data includes: aggregations like \texttt{subtotals} that sum the count
of more than one category together or headings which can be added between categories.

Insertions objects are containers for individual Insertion objects. The individual Insertions contain all the information needed to calculate, apply, and display insertions to CrunchCubes and categorical variables.

An Insertion must have two properties:

- anchor - which is the id of the category the insertion should follow
- name - the string to display

Additionally, Insertions may also have the following two properties (though if they have one, they must have the other):

- function - the function to use to aggregate (e.g. "subtotal")
- args - the category ids to use as operands to the function above.

Although it is possible to make both subtotals and headings using Insertion alone, it is much easier and safer to use the functions Subtotal() and Heading() instead. Not only are they more transparent, they also are quicker to type, accept both category names as well as ids, and have easier to remember argument names.

---

**interactVariables**

Create a variable by interacting categorical variables

**Description**

interactVariables takes two or more variables and creates a new one that is the cartesian product expansion of their unique values. For example, if we cross ethnicity (with 2 categories) and race (with 4 categories), the new variable would have 8 valid categories (e.g. black:hispanic, white:hispanic, black:non-hispanic, etc.) and 7 categories where at least one of the variables is missing (e.g. white:No Data).

**Usage**

interactVariables(..., name, sep = ":")

**Arguments**

... a sequence of categorical variables to make an interaction from as well as other properties to pass about the case variable (i.e. alias, description)

name a character to use as the name for the interaction variable

sep a character to separate the values of the individual variables (default: :)

**Value**

A VariableDefinition that creates the new interaction variable.
### Examples

```r
## Not run:
ds$ethn_race <- interactVariables(ds$ethnicity, ds$race, name="Interaction of ethnicity and race")
## End(Not run)
```

### Description
Crunch categorical variables allow you to set multiple categories as missing. For instance, you might have "not answered" and "doesn’t know" both coded as missing. This function returns a logical vector of all dataset entries that fall into any of the missing categories. It also allows you to append additional categories to the list of missing categories using the setter.

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

## 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-public View and modify "public" attribute

Description

View and modify whether all dataset viewers have access to the dataset. This will return FALSE if the dataset is in draft.

Usage

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

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

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

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

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

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

Arguments

- x: a Crunch object
- value: an attribute to set

Value

For is.public, a logical value for whether the object is flagged as shared with all dataset viewers. (Its setter thus takes a logical value as well.) Catalogs of datasets return a vector of logicals corresponding to the length of the catalog, while entities return a single value.
Description

Crunch Multiple Response variables identify one or more categories as "selected". These methods allow you to get or set which categories should indicate a selection.

Usage

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

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

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

Arguments

- `x` Categories or a single Category
- `value` A logical vector indicating whether the category should be selected. For a single category the value should be either TRUE or FALSE to change the selection status for a Categories object. Supply a logical vector which is the same length as the number of categories.

Value

Getters return a logical vector indicating selection status. Setters return the Categories or Category object, 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, while published datasets are available to all viewers. A dataset can either be published or in draft, but not both. These properties are accessed and set with the "is" methods. You can also set the properties by assigning into the function. The verb functions archive and publish are alternate versions of the setters.
Usage

```r
## 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
is.dataset

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.

Examples

```r
## Not run:
ds <- LoadDataset("mtcars")
is.draft(ds) # FALSE
is.published(ds) # TRUE
identical(is.draft(ds), !is.published(ds))
# Can make a dataset a "draft" by:
is.draft(ds) <- TRUE
is.published(ds) # FALSE
# Could also have set is.published(ds) <- FALSE
# Now, can go the other way by setting is.draft, is.published, or:
ds <- publish(ds)
is.published(ds) # TRUE

is.archived(ds) # FALSE
is.archived(ds) <- TRUE
is.archived(ds) # TRUE
# Could have achieved the same effect by:
ds <- archive(ds)

## End(Not run)
```

is.dataset Test whether a Crunch object belongs to a class

Description

Test whether a Crunch object belongs to a class

Usage

is.dataset(x)

is.CrunchExpr(x)

is.Expr(x)

is.Geodata(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.

Description

As `base::merge()` does for dataframes, this function takes two datasets, matches rows based on a specified key variable, and adds columns from one to the other.

Usage

joinDatasets(x, y, by = intersect(names(x), names(y)), by.x = by, by.y = by, all = FALSE, all.x = TRUE, all.y = FALSE, copy = TRUE)

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

## S3 method for class 'CrunchDataset'
merge(x, y, by = intersect(names(x), names(y)), by.x = by, by.y = by, 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
character, optional shortcut for specifying by.x and by.y by alias if the key variables have the same alias in both datasets.

by.x
CrunchVariable in x on which to join, or the alias (following `crunch.namekey.dataset`) of a variable. Must be type numeric or text and have all unique, non-missing values.

by.y
CrunchVariable in y on which to join, or the alias (following `crunch.namekey.dataset`) of a variable. Must be type numeric or text and 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.

copy logical: make a virtual or materialized join. Default is TRUE, which means materialized. Virtual joins are experimental and not advised.

... additional arguments, ignored

Details
Since joining two datasets can sometimes produce unexpected results if the keys differ between the two datasets, you may want to follow the fork-edit-merge workflow for this operation. To do this, fork the dataset with forkDataset(), join the new data to the fork, ensure that the resulting dataset is correct, and merge it back to the original dataset with mergeFork(). For more, see vignette("fork-and-merge", package = "crunch").

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

listDatasets
Show the names of all Crunch datasets associated with a catalog

Description
If shiny is TRUE the function launches a shiny gadget which allows you to navigate your Crunch projects and datasets. This is useful if you can’t remember a dataset’s project and also saves typing long dataset names.

Usage
listDatasets(kind = c("active", "all", "archived"), project = NULL, refresh = FALSE, shiny = 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.

shiny logical: launch a shiny gadget to help select the right dataset. The gadget will return a valid loadDataset() call which loads the selected dataset.
loadDataset

Value

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

Description

Load a Crunch Dataset

Usage

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

Arguments

dataset character, the name of a Crunch dataset that 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

Examples

```r
## Not run:
dsName <- listDatasets()[1]
ds <- loadDatasets(dsName)

## End(Not run)
```
locateEntity  

*Find an entity in an order object*

**Description**

Find an entity in an order object

**Usage**

`locateEntity(x, ord)`

**Arguments**

- `x`  
  Variable or Dataset, depending on the type of order, or URL for it
- `ord`  
  ShojiOrder (VariableOrder or DatasetOrder)

**Value**

If `x` is found in `ord`, a character vector of group names that provide the "path" to the entity. The length of the vector corresponds to the depth of nesting. If not found, `NA` is returned

---

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 = envOrOption("crunch.email"),
     password = envOrOption("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

Additionally, your email and password can be stored in and read from the environmental variables R_CRUNCH_EMAIL and R_CRUNCH_PW respectively.

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()
makeArrayGadget

### Description

Launch array builder gadget

### Usage

makeArrayGadget()
makeCaseVariable

Details

Categorical Array and Multiple Response variables can be difficult to construct without being able to investigate the available variables, and their categories. This shiny gadget lets you select subvariables from the dataset list, and ensures that those variables have consistent categories. To use the gadget you must have at least one CrunchDataset loaded into the global environment.

Value

a valid call to makeArray() or makeMR()

makeCaseVariable Make a case variable

Description

The makeCaseVariable function derives a variable using values from other variables. These are evaluated in the order they are supplied in the list as the cases argument (they proceed in an IF, ELSE IF, ELSE IF, ..., ELSE fashion); the first one that matches selects the corresponding value from the case list.

Usage

makeCaseVariable(..., cases, data = NULL, name)

Arguments

... a sequence of named expressions to use as cases as well as other properties to pass about the case variable (i.e. alias, description)
cases a list of lists with each case condition to use each must include at least a name and an expression element. Cases may also include missing (logical) and numeric_value (numeric).
data (optional) a crunch dataset to use. Specifying this means you don't have to put dataset$ in front of each variable name.
name a character to use as the name of the case variable to create

Details

There are two ways to specify cases, but you must pick only one (note these two will produce the same case variable):

1. When you just want to specify conditions, you can use named conditions: makeCaseVariable(case1=ds$v1 == 1, case2=ds$v2 == 2, ..., name="case")
2. You can also use the cases argument, which is useful when you want to provide category ids, numeric values, or missingness: makeCaseVariable(cases=list(list(expression=ds$v1 == 1, name="case1"), list(expression=ds$v2 == 2, name="case2")), name="case")

Rows in the dataset that do not match any of the provided "cases" will be assigned to an "else" category. By default, Crunch will use the system missing "No Data" category. Alternatively, you can provide an else case definition for these rows by including as the last "case" you provide one with its expression set to the string "else". See the examples for details.
Value

A **VariableDefinition** that will create the new case variable when assigned into the Dataset.

Examples

```r
## Not run:
makeCaseVariable(case1=ds$v1 == 1, case2=ds$v2 == 2, name="new case")
makeCaseVariable(cases=list(list(expression=ds$v1 == 1, name="case1"),
                      list(expression=ds$v2 == 2, name="case2")),
                name="new case")

# different ways to specify else cases
makeCaseVariable(cases=list(list(expression=ds$v1 == 1, name="case1"),
                           list(expression=ds$v2 == 2, name="case2"),
                           list(expression="else", name="other")),
                name="new case")
makeCaseVariable(case1=ds$v1 == 1, case2=ds$v2 == 2, other="else", name="new case")

# the dataset can be specified with data=
makeCaseVariable(case1=v1 == 1, case2=v2 == 2, data=ds, name="new case")

##(Not run)
```

---

**makeMRFromText**

*Create Multiple Response Variable from Delimited lists*

**Description**

Surveys often record multiple response questions in delimited lists where each respondent’s selections are separated by a delimiter like ; or |. This function breaks the delimited responses into subvariables, uploads those subvariables to Crunch, and finally creates a multiple response variable from them.

**Usage**

```r
makeMRFromText(var, delim, name, selected = "selected",
               not_selected = "not_selected", unanswered = NA, ...)
```

**Arguments**

- `var` : The variable containing the delimited responses
- `delim` : The delimiter separating the responses
- `name` : The name of the resulting MR variable
- `selected` : A character string used to indicate a selection, defaults to "selected"
- `not_selected` : Character string identifying non-selection, defaults to "not_selected"
- `unanswered` : Character string indicating non-response, defaults to NA.
- `...` : Other arguments to be passed on to `makeMR()`
**makeWeight**

Value

a Multiple response variable definition

---

**Description**

This function allows you to generate a weight variable by supplying a set of categorical variables and the target distribution for each of the variables’ categories. Weights are computed by iteratively ‘raking’ conditional ‘cells’ to the provided marginal targets.

**Usage**

```r
makeWeight(..., name)
```

**Arguments**

- `...` A series of expressions of the form `variable ~ target_weights`. The variable must be a categorical Crunch variable, and the target weights must be a numeric vector whose length should be equal to the number of categories contained in the variable, and whose sum is equal to 100 or 1. If you supply fewer target weights than there are categories `makeWeight` will pad the target weight vector with 0s.

- `name` The name of the resulting variable

**Details**

For instance, if you wanted to create a weight variable which equally weighted four categories stored in `ds$var` you would call `ds$weight1 <- makeWeight(ds$var ~ c(25, 25, 25, 25), name = "weight1").` Note that `makeWeight` returns a `VariableDefinition`, an expression that when assigned into your Dataset becomes a derived variable. This does not on its own set the new variable as “the weight” for your dataset. To set that attribute, use `weight()`. Alternatively, you can also create the variable and set the weight attribute in one step with `weight(ds) <- makeWeight(ds$var ~ c(25, 25, 25, 25), name = "weight1").` 

**Value**

A crunch `VariableDefinition()` of the weight variable

**See Also**

`weight<-(); settings()` for the "default weight" for other dataset viewers.
Examples

```r
## Not run:
mtcars$cyl <- as.factor(mtcars$cyl)
mtcars$gear <- as.factor(mtcars$gear)
ds <- newDataset(mtcars)
# Create a new "raked" variable
ds$weight <- makeWeight(ds$cyl ~ c(30, 30, 40, 0),
                        ds$gear ~ c(20, 20, 60, 0), name = "weight")
summary(ds$weight)
# ds$weight is not "the weight" for the dataset unless you set it:
weight(ds) <- ds$weight
# Or, you can create the variable and set as weight in one step:
weight(ds) <- makeWeight(ds$var ~ c(25, 25, 25, 25), name = "weight2")
## End(Not run)
```

---

**matchCatToFeat**

*Match categories with features from geodata*

**Description**

Match categories with features from geodata

**Usage**

```r
matchCatToFeat(categories, all_features = availableGeodataFeatures())
```

**Arguments**

- `categories` : a vector of categories to match
- `all_features` : a dataframe of all available geodata features. (default: downloaded from Crunch servers)

**Value**

geodatum to associate with the variable that produced categories
### My user entity

**Description**

Get the user entity of the currently authenticated user.

**Usage**

```r
me()
```

**Value**

A `UserEntity`

---

### 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 set of datasets with a user all at once by adding the user to a team that contains those datasets.

**Usage**

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

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

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

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

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

## S4 method for signature 'CrunchProject'
members(x)
```
## S4 replacement method for signature 'CrunchProject,MemberCatalog'

```r
members(x) <- value
```

## S4 replacement method for signature 'CrunchProject,character'

```r
members(x) <- value
```

## S4 method for signature 'CrunchTeam'

```r
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. You can extract an individual team by name, or create a team by assigning into the function. To create a team by assignment, assign a list to `teams("myteam") <- value_list`, the `value_list` can either empty (to just create a team with that name), or can contain a "members" element with the emails or URLs of users to add to the team. Users can be also 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`

---

## merge

### Merge a CrunchDataFrame

**Description**

Merging a CrunchDataFrame with a local dataframe is useful in situations where you have new information in your local R session that you want to connect with Crunch data. For example, for making plots with Crunch and non-Crunch data. It produces a hybrid CrunchDataFrame that has the local data attached to it, but like normal CrunchDataFrames it is still judicious about downloading data from the server only when it is needed.

**Usage**

```r
## S3 method for class 'CrunchDataFrame'
merge(x, y, by = intersect(names(x), names(y)),
       by.x = by, by.y = by, sort = c("x", "y"), ...)
```
mergeFork

Arguments

- **x**: a CrunchDataFrame
- **y**: a standard data.frame
- **by**: name of the variable to match in both data sources (default: the intersection of the names of x and y)
- **by.x**: name of the variable to match in x
- **by.y**: name of the variable to match in y
- **sort**: character, either "x" or "y" (default: "x"). Which of the inputs should be used for the output order. Unlike merge.data.frame, merge.CrunchDataFrame will not re-sort the order of the output. It will use the order of either x or y.
- **...**: ignored

Details

Merging a CrunchDataFrame with a local dataframe does not allow specifying all rows from both sources. Instead, the resulting CrunchDataFrame will include all of the rows in whichever source is used for sorting (x or y). So if you specify sort="x" (the default) all rows of x will be present but rows in y that do not match with rows in x will not be present.

Merging a CrunchDataFrame with a local dataframe is experimental and might result in unexpected results. One known issue is that using merge on a CrunchDataFrame will change the both the CrunchDataFrame used as input as well as create a new CrunchDataFrame.

Value

a CrunchDataFrame with columns from both x and y

mergeFork

Merge changes to a dataset from a fork

Description

Crunch datasets include information about the dataset’s revision history. This function takes the changes made on a dataset fork and adds them to the revision history of the parent dataset, like a merge of branches in a version control system.

Usage

mergeFork(dataset, fork, autorollback = TRUE, force = FALSE)
mergeFork

Arguments

- **dataset**: The CrunchDataset to merge to
- **fork**: The CrunchDataset, which must be a fork 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.
- **force**: logical Attempt to push through merge conflicts by dropping all changes to dataset that occurred after fork diverged from and take only the changes from fork? Default is FALSE. You should only use force=TRUE after first attempting and failing to merge without forcing.

Details

All modifications of a dataset record actions in its revision history. For example, if you add a variable to the dataset, that action is recorded. The sum of these records is a dataset's revision history, and it is possible to merge in the revision history of a dataset that has been forked.

This function is most often used in conjunction with `forkdataset()` to create a copy of a dataset, make some changes to that copy, and then merge the changes back into the original dataset. For more on this workflow, see vignette("fork-and-merge", package = "crunch").

Value

dataset with changes from fork merged to it.

See Also

- `forkdataset()`

Examples

```r
# Not run:
ds <- loadDataset("My survey")
fork <- forkdataset(ds)
# Do stuff to fork
ds <- mergeFork(ds, fork)
# Now the changes you did to fork are also on ds

# End(Not run)
```
**modifyWeightVariables**  \( \textit{Change which variables can be set as a dataset's weight.} \)

**Description**

modifyWeightVariables allows you to change the variables which are eligible to be used as a dataset's weight. You can also add variables to the weight variables catalog by assignment with 

```r
weightVariables(ds) <- "weight" or is.weightVariable(ds$weight) <- TRUE.
```

**Usage**

```r
modifyWeightVariables(x, vars, type = "append")
```

```r
## S4 replacement method for signature 'CrunchDataset'
weightVariables(x) <- value
is.weightVariable(x)
```

```r
## S4 replacement method for signature 'NumericVariable'
is.weightVariable(x) <- value
```

**Arguments**

- **x**  
  a CrunchDataset
- **vars**  
  Variables to add or remove this can be a numeric Crunch variable, list of numeric Crunch variables or a character vector with the aliases of numeric Crunch variables. Setting vars to NULL clears a datasets weightVariables
- **type**  
  a character string determining how the weightVariables will be modified:
  - "append": add vars to the current weight variables
  - "remove": remove vars from the current list of weight variables
  - "replace": replace the current weight variables with vars
- **value**  
  For the weightVariables() and is.weightVariable setters the variables to append to a dataset's weightVariables.

**Details**

Editors can change which variables can be set as the weighting variable for a dataset. For instance if several weights have been calculated they can let the user choose which of those variables to use a weight, but prevent the user from choosing other variables as weight. This function allows you to change the weightVariables of a dataset.

**Value**

- a CrunchDataset
Examples

```r
## Not run:
modifyWeightVariables(ds, "weight", "append")
weightVariables(ds) <- list(ds$weight, ds$weight2)
weightVariables(ds) <- NULL
weightVariables(ds) <- c("weight", "weight2")
is.weightVariables(ds$weight) <- TRUE

## End(Not run)
```

moveToGroup

Move entities to a group

Description

Shoji entities can be placed into groups, this is mostly used for grouping variables for display in the app, but is technically possible for any of the order catalogs. This function moves an entity to one of these groups.

Usage

```r
moveToGroup(x, value)
moveToGroup(x) <- value
```

Arguments

- `x` : VariableGroup
- `value` : Variable, VariableCatalog subset, or Dataset subset

Details

The function has two versions: a regular function and a setter. They do the same thing, but the setter is probably more succinct.

Value

`x` with the entities in `value` appended to it. If the containing order object has `duplicates=FALSE`, the entities will be "moved" to this group. Otherwise, their references will be copied to the group.

Examples

```r
## Not run:
moveToGroup(ordering(ds)[["Demographics"]]) <- ds[c("gender", "age")]

## End(Not run)
```
Multitable Catalog

Description

Multitable entities for a dataset

Usage

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

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

Arguments

- `x`: a CrunchDataset
- `value`: for the assignment method, a MultitableCatalog

Value

an object of class MultitableCatalog containing references to Multitable entities. (Setter returns the Dataset.)

---

mv

Functions to manipulate variables’ or project’s folder structure

Description

Variables in Crunch datasets are organized into folders, like in a file system. Datasets are similarly organized into hierarchical Projects. These functions allow you to create new folders and move objects into folders. Their names, `mv` and `mkdir`, suggest their Unix file utility inspiration.

Usage

```
mv(x, what, path)

mkdir(x, path)
```
Arguments

- **x**: A CrunchDataset or Folder (VariableFolder or ProjectFolder)
- **what**: A Variable, selection of variables from dataset, or any other object that can be moved to a folder (e.g., a dataset when organizing projects).
- **path**: A character "path" to the folder: either a vector of nested folder names or a single string with nested folders separated by a delimiter ("/") default, configurable via options(crun.ch.delimiter). The path is interpreted as relative to the location of the folder x (when x is a dataset, that means the root, top-level folder). path may also be a Folder object.

Details

The functions have some differences from the strict behavior of their Unix ancestors. For one, they work recursively, without additional arguments: mkdir will make every directory necessary to construct the requested path, even if all parent directories didn’t already exist; and mv doesn’t require that the directory to move to already exist—it will effectively call mkdir along the way.

Value

x, with the folder at path guaranteed to be created, and for mv, containing what moved into it.

See Also

cd() to select a folder by path; rmdir() to delete a folder; folder() to identify and set an object’s parent folder; base::dir.create() if you literally want to create a directory in your local file system, which mkdir() does not do

Examples

```r
# Not run:
ds <- loadDataset("Example survey")
ds <- mv(ds, c("gender", "age", "educ"), "Demographics")
ds <- mkdir(ds, "Key Performance Indicators/Brand X")
# These can also be chained together
require(magrittr)
ds %>%
  mv(c("aware_x", "nps_x"), "Key Performance Indicators/Brand X") %>
  mv(c("aware_y", "nps_y"), "Key Performance Indicators/Brand Y")
# Can combine with cd() and move things with relative paths
ds %>%
  cd("Key Performance Indicators/Brand X") %>
  mv("nps_x", "../Net Promoters")
# Can combine with folder() to move objects to the same place as something else
ds %>% mv("nps_y", folder(ds$nps_x))
# Now let's put ds in a Project
projects() %>%
  mv(ds, "Brand Tracking Studies")

# End(Not run)
```
**Description**

Omit missing categories

**Usage**

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

**Arguments**

- `object` : Categories
- `...` : additional arguments, ignored

**Value**

`object` with any categories that have missing: `TRUE` excluded

---

**name,AbstractCategory-method**

*Category attributes*

**Description**

Functions to access and set category attributes.

**Usage**

```r
## S4 method for signature 'AbstractCategory'
name(x)

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

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

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

## S4 method for signature 'Category'
value(x)
```
names,BatchCatalog-method

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

## 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`

---

**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

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

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

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

## 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 'ShojiFolder'
types(x)

## 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 'VariableCatalog'
ids(x)

## S4 method for signature 'VariableFolder'
aliases(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.
For the setters, an appropriate-length character vector to assign.

Details

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

Value

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

See Also

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

Description

This function creates a new dataset on the Crunch server with either a data.frame or similar object in your R session, a file, or a URL to a file. It captures available metadata from your R object and translates it into Crunch types.

Usage

newDataset(x, name = NULL, ...)

Arguments

x a data.frame or other rectangular R data object, or a string file name or URL to upload to create a dataset. The file may be a compressed Zip file containing a single file in CSV or SPSS format.

name character name to give the new Crunch dataset. By default the function uses the name of the R object, or, if passing a file, the file name.

... additional arguments passed to createDataset()

Details

If you have an SPSS file, it is better specify the file name directly rather than first reading it into R. Uploading SPSS files directly to Crunch will preserve metadata that is stripped by the R import, regardless of the library used to read it into R.

Value

If successful, an object of class CrunchDataset.
newFilter

See Also

c有针对性FromFile(); newDatasetByColumn() for an alternate upload method.

Examples

```r
## Not run:
dse <- newDataset(mtcars, "cars")
dse <- newDataset("mySurvey.sav")

## End(Not run)
```

newFilter Create a new filter

Description

This function creates a new filter for a CrunchDataset. You can achieve the same results by assigning into a dataset’s filters catalog using filters(), but this may be a more natural way to think of the action, particularly when you want to do something with the filter entity after you create it.

Usage

```r
newFilter(name, expression, catalog = NULL, ...)
```

Arguments

- **name**: character name for the filter
- **expression**: CrunchLogicalExpr with which to make a filter entity
- **catalog**: FilterCatalog in which to create the new filter. May also provide a dataset entity. If omitted, the function will attempt to infer the dataset (and thus its FilterCatalog) from the contents of expression.
- **...**: Additional filter attributes to set, such as is_public.

Value

A CrunchFilter object.
newMultitable  

Create a new Multitable

Description

Multitables, or "banners" or "crossbreaks", define a set of variables or or query expressions to crosstab with as a unit. They are used in the Crunch web app to display tables side by side, as well as to define one dimension of a tab book.

Usage

newMultitable(formula, data, name, ...)

Arguments

formula an object of class 'formula' object with the cross-classifying variables separated by '+' on the right-hand side. Following how stats::formula() works in R, it should start with "~". Variables on left-hand side of the formula have no meaning in this function.

data an object of class CrunchDataset in which to create the multitable, and to which the variables referenced in formula belong.

name character name to give the new multitable object. If omitted, a default name will be derived from formula.

... Additional multitable attributes to set. Options include is_public.

Value

An object of class Multitable

See Also

stats::formula

Examples

## Not run:
m <- newMultitable(~ gender + age4 + marstat, data=ds)
name(m) # [1] "gender + age4 + marstat"

## End(Not run)
newProject

Create a new project

Description

This function creates a new project. You can achieve the same results by assigning into the projects catalog, but this may be a more natural way to think of the action, particularly when you want to do something with the project entity after you create it.

Usage

newProject(name, members = NULL, catalog = projects(), ...)

Arguments

name character name for the project

members Optional character vector of emails or user URLs to add as project members.

catalog ProjectCatalog in which to create the new project. There is only one project catalog currently, projects(), but this is left here so that all new* functions follow the same pattern.

... Additional project attributes to set

Value

A CrunchProject object.

Examples

## Not run:
proj <- newProject("A project name")
# That is equivalent to doing:

p <- projects()
p["A project name"] <- list() proj <- p["A project name"]

proj2 <- newProject("Another project", members="you@yourco.com")
# That is equivalent to doing:
p["Another project"] <- list(members="you@yourco.com")
proj <- p["Another project"]

## End(Not run)
Description

Remove transformations from a CrunchCube

Usage

noTransforms(cube)

Arguments

cube a CrunchCube

Value

the CrunchCube with no transformations

Removing transforms

noTransforms() is useful if you don’t want to see or use any transformations like Subtotals and Headings. This action only applies to the CrunchCube object in R: it doesn’t actually change the variables on Crunch servers or the query that generated the CrunchCube.

Examples

```r
## Not run:
# A CrunchCube with a heading and subtotals
crtabs(~opinion, ds)
#   All opinions
#   Strongly Agree 23
#   Somewhat Agree 24
#   Agree 47
# Neither Agree nor Disagree 18
#   Somewhat Disagree 16
#   Strongly Disagree 19
#   Disagree 35

noTransforms(crtabs(~opinion, ds))
#   Strongly Agree Somewhat Agree Neither Agree nor Disagree
#   23 24 18
#   Somewhat Disagree Strongly Disagree
#   16 19

## End(Not run)
```
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

```r
## 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 method for signature 'CrunchProject'
ordering(x)

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

## S4 replacement method for signature 'CrunchProject'
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
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.

pendingStream
Get the pending streams for a dataset

Description
Retrieves the number of pending messages. Use appendStream() to append all pending streamed rows to the dataset.

Usage
pendingStream(ds)

Arguments
ds          a CrunchDataset

Value
number of pending messages in the stream for the dataset
permissions

See who has access to a dataset

Description
See who has access to a dataset

Usage
```r
## 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.

pk

Get and set the primary key for a Crunch dataset

Description
A primary key is a variable in a dataset that has a unique value for every row. A variable must be either numeric or text type and have no duplicate or missing values. A primary key on a dataset causes updates to that dataset that have the rows with the same primary key value(s) as the first dataset to update the existing rows rather than inserting new ones.

Usage
```r
## S4 method for signature 'CrunchDataset'
pk(x)

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

Arguments
- `x` a Dataset
- `value` For the setter, a single Variable to use as the primary key or NULL to remove the primary key.

Value
- Getter returns the Variable object that is used as the primary key (NULL if there is no primary key);
- setter returns x duly modified.
pollProgress

*Check a Crunch progress URL until it finishes*

**Description**

You'll probably only call this function if progress polling times out and its error message tells you to call `pollProgress` to resume.

**Usage**

```r
pollProgress(progress_url, wait = 0.5)
```

**Arguments**

- `progress_url` : A Crunch progress URL
- `wait` : Number of seconds to wait between polling. This time is increased 20 percent on each poll.

**Value**

The percent completed of the progress. Assuming the `options(crunch.timeout)` (default: 15 minutes) hasn’t been reached, this will be 100. If the timeout is reached, it will be the last reported progress value.

---

population

*Get and set the market size for Crunch datasets*

**Description**

Crunch Datasets allow you to set a target population size in order to extrapolate population estimates from survey percentages. These functions let you work with the population size and magnitude.

**Usage**

```r
## S4 method for signature 'CrunchDataset'
popSize(x)

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

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

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

### Description

CrunchBoxes allows you to share data with the world in a simple, easy to embed format. However, not all datasets naturally translate to the CrunchBox format. This function checks your dataset to see if it

### Usage

`preCrunchBoxCheck(dataset)`

### Arguments

- **dataset**: CrunchDataset, potentially subsetted on variables

### Value

Invisibly, the dataset. Called for side-effect of printing things.

### See Also

CrunchBox
prepareDataForCrunch  Translate a data.frame to Crunch format

Description
This is called within newDataset to extract the Crunch metadata from the data and to transform the
data to match the extracted metadata. You can call this directly in order to tailor the data import
flow more finely.

Usage
prepareDataForCrunch(data, ...)

Arguments
data  A data.frame or other rectangular R object
...  additional arguments passed to createDataset. "name" will be required by the
      Crunch server but is not required by this function.

Value
A data.frame that is a transformation of data suitable for uploading to Crunch, also containing a
"metadata" attribute that is the associated Crunch metadata.

See Also
createWithPreparedData writePreparedData

project-icon  Get or set a project’s icon

Description
This function allows you to change a project’s icon. Note that you must first upload the icon to a
website, as this function does not allow you to upload a local file.

Usage
icon(x)
icon(x) <- value

Arguments
x  a CrunchProject
value  character 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 an associated project catalog. 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)

prop.table,CrunchCube-method

Work with CrunchCubes, MultitableResults, and TabBookResults

Description

These functions provide an interface like base::margin.table() and base::prop.table() for the CrunchCube object. CrunchCubes contain richer metadata than standard R array objects, and they also conceal certain complexity in the data structures from the user. In particular, multiple-response variables are generally represented as single dimensions in result tables, but in the actual data, they may comprise two dimensions. These methods understand the subtleties in the Crunch data types and correctly compute margins and percentages off of them.
Usage

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

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

## S4 method for signature 'CrunchCube'
bases(x, margin = NULL)

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

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

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

## S4 method for signature 'TabBookResult'
bases(x, margin = NULL)

## S4 method for signature 'MultitableResult'
bases(x, margin = NULL)
```

Arguments

- **x**: a CrunchCube
- **margin**: index, or vector of indices to generate margin for. See `base::prop.table()`. `bases()` accepts 0 as an additional valid value for margin, which yields the unweighted counts for the query.
- **digits**: For `round`, the number of decimal places to round to. See `base::round()`

Details

These functions also generalize to MultitableResults and TabBookResults, which are returned from a `tabBook()` request. When called on one of those objects, they effectively apply over each CrunchCube contained in them.

`bases` is an additional method for CrunchCubes. When making weighted requests, `bases` allows you to access the unweighted counts for every cell in the resulting table (array). The `bases` function takes a "margin" argument to work like `margin.table`, or with `margin=0` gives all cell counts.

Value

When called on CrunchCubes, these functions return an array. Calling `prop.table` on a MultitableResult returns a list of `prop.tables` of the CrunchCubes it contains. Likewise, `prop.table` on a TabBookResult returns a list of lists of `prop.tables`.
See Also

base::margin.table() base::prop.table()

Description

Crunch objects generally keep themselves in sync with the server when you manipulate them, but some operations cause the local version to diverge from the version on the server. For instance, someone else may have 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

```r
## 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`

Description

This function recurses through a ShojiOrder/OrderGroup and removes any groups that contain no entities.

Usage

```r
removeEmptyGroups(x)
```
resetPassword

Description
Trigger the password reset process. Password reset instructions will be emailed to you.

Usage
resetPassword(email)

Arguments
email Your email

Value
NULL, invisibly. Called for its side effects.

Examples
```r
## Not run:
resetPassword("me@example.com")

## End(Not run)
```

restoreVersion

Description
You can save a version of a dataset using `saveVersion()`. Savepoints are also created automatically by certain Crunch functions that make major changes to the dataset. You can get the list of saved versions with the `versions()` function.

Usage
restoreVersion(dataset, version)
**Description**

Like `rmdir` in a file system, this function removes a folder. Unlike the file-system version, it does not require the folders to be empty.

**Usage**

```r
rmdir(x, path)
```

**Arguments**

- `x` A CrunchDataset or Folder (VariableFolder or ProjectFolder)
- `path` A character "path" to the folder: either a vector of nested folder names or a single string with nested folders separated by a delimiter ("/") default, configurable via options(crunch.delimiter)). The path is interpreted as relative to the location of the folder `x` (when `x` is a dataset, that means the root, top-level folder). `path` may also be a Folder object.

**Value**

NULL

**See Also**

`mv()` to move entities to a folder; `cd()` to select a folder; `base::file.remove()` if you literally want to delete a directory from your local file system, which `rmdir()` does not do.
saveVersion

Examples

```r
## Not run:
ds <- loadDataset("Example survey")
rmdir(ds, "Demographics")
# Or with %>%
require(magrittr)
d <- ds %>%
  rmdir("Demographics")

## End(Not run)
```

Description

Crunch datasets can be saved and restored using `saveVersion` and `restoreVersion()`. Some Crunch functions, such as `appendDataset()` create new savepoints automatically. To see the list of savepoints use `versions()`.

Usage

```r
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`
scoreCatToFeat

Score similarity between a feature dataframe and categories

Description

Implemented using the Jaccard index, where a number closer to 1 is more similar.

Usage

scoreCatToFeat(features, categories)

Arguments

features a vector of features to match (usually from a subset of the output [availableGeodataFeatures]) with a single property for a single geodatum.
categories a vector of categories to match

Value

the Jaccard index for the values of the property given in feat_df and the vector of categories

searchDatasets

Search Crunch for datasets.

Description

searchDatasets searches datasets’ metadata for matches to the query argument. This search will include variable names, aliases, categories, but not the content of text variables. See the API Documentation for more information about searching Crunch.

Usage

searchDatasets(query, ...)

Arguments

query the text to search for in datasets and their variables (note: only alpha characters will be used, numbers and other characters will be discarded.)

Value

If successful, an object of class SearchResults
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`

Description

Get various catalogs for your Crunch session

Usage

```
session()
```

Value

A Session object. Access dataset and project catalogs from it.

Examples

```
## Not run:
cr <- session()
cr$datasets
cr$projects

## End(Not run)
```
Extract catalogs from a Session object

Description

Extract catalogs from a Session object

Usage

```r
## S4 method for signature 'Session,ANY'
[[i, ..., drop = FALSE]]

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

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

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

Arguments

- `x` a Session object
- `i` which catalog to load. Supported values are "datasets" and "projects"
- `...` additional arguments, ignored.
- `drop` invalid
- `name` for $, the same as `i` for 
- `value` For updating, an object of the appropriate class and size to insert. In practice value is ignored; Session objects hold no state and it is assumed that any state modification on the server happens in other methods.

Value

The requested catalog when extracting; a Session object if assigning.

See Also

session
### setName

**Change the name of the current folder**

**Description**

If you just need to change the name of the folder you are currently in, you can use `setName()`. It doesn’t move variables or change anything other than the name of the current folder.

**Usage**

```
setName(object, nm)
```

**Arguments**

- `object`: A Folder
- `nm`: A character that is the new name the folder should have

**Value**

`object`, with its name duly changed

**See Also**

`cd()` and `mv()`

**Examples**

```r
## Not run:
ds <- ds %>%
   cd("Demographics") %>%
   setName("Key Demos.")

## End(Not run)
```

### setNames

**Change the name of the entities in a catalog**

**Description**

This is an alternative to assigning `names(catalog) <- something`, suitable for inclusion in a pipeline.

**Usage**

```
setNames(object, nm)
```

**Examples**

```r
## S4 method for signature 'ShojiCatalog'
setNames(object, nm)
```
setOrder

Arguments

object A catalog object, such as VariableFolder
nm A character vector of new names of the same length as the number of entities in the index

Value

object, with the names of its children duly changed

See Also

cd() and mv()

Examples

## Not run:
ds <- ds %>%
cd("Demographics") %>%
setNames(c("Gender (4 category)", "Birth year", "Race (5 category)"))

## End(Not run)
These methods allow access and control over dataset settings. Currently supported settings include:

- User Authorizations for view-only users ('viewers_can_export', 'viewers_can_share', and 'viewers_can_change_weight'); and
- 'weight', which determines the default weighting variable for the dataset. Additional settings will be added in the future. See http://docs.crunch.io/#fragments, under 'Settings', for an up-to-date list of settings supported throughout the Crunch system. Clients may also provide and use custom settings if they choose.

**Usage**

```r
settings(x)
```

```r
settings(x) <- value
```

**Arguments**

- `x`: CrunchDataset
- `value`: A settings object (ShojiEntity), for the setter

**Value**

The getter returns a settings object (ShojiEntity). The setter returns the dataset (x), duly modified.

**Examples**

```r
## Not run:
settings(ds)
settings(ds)$viewers_can_export <- TRUE
settings(ds)$weight <- ds$myWeightVariable

## End(Not run)
```
**share**  

*Share a dataset*

**Description**

Share a dataset

**Usage**

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

**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.
- **message**: character: a message to send to the users who are receiving new privileges.

**Value**

Invisibly, the dataset.

**See Also**

unshare

---

**shoji-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.

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 VariableOrders and VariableGroups

Description

Extract and update in VariableOrders and VariableGroups

Usage

```r
## 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'
\( x[i, j] \leftarrow \text{value} \)

## S4 replacement method for signature 'ShojiOrder,character,missing,list'
\( x[[i, j]] \leftarrow \text{value} \)

## S4 replacement method for signature 'ShojiOrder,character,missing,character'
\( x[[i, j]] \leftarrow \text{value} \)

## S4 replacement method for signature 'ShojiOrder,character,missing,OrderGroup'
\( x[[i, j]] \leftarrow \text{value} \)

## S4 replacement method for signature 'ShojiOrder,ANY,missing,OrderGroup'
\( x[[i, j]] \leftarrow \text{value} \)

## S4 replacement method for signature 'ShojiOrder,ANY,missing,ANY'
\( x[[i, j]] \leftarrow \text{value} \)

## S4 replacement method for signature 'ShojiOrder,ANY,missing,'NULL''
\( x[[i, j]] \leftarrow \text{value} \)

## S4 replacement method for signature 'ShojiOrder,character,missing,'NULL''
\( x[[i, j]] \leftarrow \text{value} \)

## S4 replacement method for signature 'ShojiOrder,character,missing,ShojiOrder'
\( x[[i, j]] \leftarrow \text{value} \)

## S4 replacement method for signature 'ShojiOrder'
\( x$\text{name} \leftarrow \text{value} \)

## S4 method for signature 'OrderGroup,ANY,ANY'
\( x[i, j, \ldots, \text{drop = TRUE}] \)

## S4 method for signature 'OrderGroup,character,ANY'
\( x[i, j, \ldots, \text{drop = TRUE}] \)

## S4 method for signature 'OrderGroup,character'
\( x[[i, j, \ldots]] \)

## S4 method for signature 'OrderGroup,ANY'
\( x[[i, j, \ldots]] \)

## S4 method for signature 'OrderGroup'
\( x$\text{name} \)

## S4 replacement method for signature 'OrderGroup,character,missing,list'
\( x[[i, j]] \leftarrow \text{value} \)
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**

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

```r
# S4 method for signature 'OrderGroup'
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**

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

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

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

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

```r
# S4 replacement method for signature 'ShojiOrder'
entities(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 being 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

Description

Show methods for Crunch objects

Usage

```r
## S4 method for signature 'ShojiFolder'
show(object)

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

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

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

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

```r
show(object)
```

## S4 method for signature 'CrunchCube'

```r
show(object)
```

## S4 method for signature 'OrderGroup'

```r
show(object)
```

## S4 method for signature 'CrunchGeography'

```r
show(object)
```

## S4 method for signature 'MultitableResult'

```r
show(object)
```

### Arguments

- `object` the object

### Value

- `invisibly` invisibly

### See Also

- `methods::show`

---

### Description

`showTransforms(x)` shows a summary of a categorical variable that has transforms with the transforms calculated and applied. This is useful to see what kind transforms exist before including the variable in a CrunchCube.

### Usage

```r
## S4 method for signature 'CrunchCube'
showTransforms(x)
```

```r
## S4 method for signature 'CategoricalVariable'
showTransforms(x)
```

### Arguments

- `x` a Categorical variable or CrunchCube
SO_schema

Details

`showTransforms([CrunchCube])` shows the CrunchCube with all transforms calculated and applied. This is the default display method for cubes, so should not be frequently needed.

In both cases, an array is returned that includes the values of both the underlying data (either category counts or CrunchCube cell values) as well as the transformations applied.

Value

summary of the variable, or the full CrunchCube with transforms applied

Examples

```r
# Not run:
showTransforms(ds$variable)

# End(Not run)
```

---

SO_schema  Schema for the 2017 Stack Overflow developer survey

Description

Survey questions and variable names for the 2017 Stack Overflow Developers Survey #'

Usage

SO_schema

Format

A data frame with 23 rows and 2 variables.

- **Column**  The column name of the survey data frame
- **Question**  Question asked of respondents
SO_survey

R users who responded to the 2017 Stack Overflow developer survey

Description
A slightly modified version of the 2017 Stack Overflow developer survey. The dataset is filtered to only include respondents who have used R before, and to include illustrative variable types.

Usage
SO_survey

Format
A data frame with 1634 rows and 25 variables.

Respondent  Respondent ID number
Professional  Which of the following best describes you?
Country  In which country do you currently live?
CompanySize  In terms of the number of employees, how large is the company or organization you work for?
CareerSatisfaction  Career satisfaction rating
JobSatisfaction  Job satisfaction rating
ImportantHiringAlgorithms  Congratulations! You’ve just been put in charge of technical recruiting at Globex, a multinational high-tech firm. This job comes with a corner office, and you have an experienced staff of recruiters at your disposal. They want to know what they should prioritize when recruiting software developers. How important should each of the following be in Globex’s hiring process? Knowledge of algorithms and data structures
ImportantHiringTechExp  Congratulations! You’ve just been put in charge of technical recruiting at Globex, a multinational high-tech firm. This job comes with a corner office, and you have an experienced staff of recruiters at your disposal. They want to know what they should prioritize when recruiting software developers. How important should each of the following be in Globex’s hiring process? Experience with specific tools (libraries, frameworks, etc.) used by the employer
ImportantHiringCommunication  Congratulations! You’ve just been put in charge of technical recruiting at Globex, a multinational high-tech firm. This job comes with a corner office, and you have an experienced staff of recruiters at your disposal. They want to know what they should prioritize when recruiting software developers. How important should each of the following be in Globex’s hiring process? Communication skills
ImportantHiringOpenSource  Congratulations! You’ve just been put in charge of technical recruiting at Globex, a multinational high-tech firm. This job comes with a corner office, and you have an experienced staff of recruiters at your disposal. They want to know what they should prioritize when recruiting software developers. How important should each of the following be in Globex’s hiring process? Contributions to open source projects
Congratulations! You’ve just been put in charge of technical recruiting at Globex, a multinational high-tech firm. This job comes with a corner office, and you have an experienced staff of recruiters at your disposal. They want to know what they should prioritize when recruiting software developers. How important should each of the following be in Globex’s hiring process? Experience with specific project management tools & techniques

Previous companies worked at

Previous job titles held

Educational credentials (e.g. schools attended, specific field of study, grades earned)

Stack Overflow reputation

Track record of getting things done

Gender Which of the following do you currently identify as?

Race Which of the following do you identify as?

Salary What is your current annual base salary, before taxes, and excluding bonuses, grants, or other compensation?

ExpectedSalary You said before that you are currently learning how to program. When you have completed your studies, what annual salary do you expect to earn in your first job after graduation?

TabsSpaces Tabs or spaces?

WantWorkLanguage Which of the following languages have you done extensive development work in over the past year, and which do you want to work in over the next year?

HaveWorkedLanguage Which of the following languages have you done extensive development work in over the past year, and which do you want to work in over the next year?
Subtotal-class

Source

https://insights.stackoverflow.com/survey/

---

Subtotal-class Subtotals and headings

Description

Subtotals and headings for categorical Variables and CrunchCubes. These are especially useful for making aggregates across multiple categories (sometimes referred to as nets, top box, or top 2 box).

Usage

Subtotal(name, categories, position = c("relative", "top", "bottom"),
          after = NULL)

Heading(name, position = c("relative", "top", "bottom"), after = NULL)

Subtotal(name, categories, position = c("relative", "top", "bottom"),
          after = NULL)

is.Subtotal(x)

is.Heading(x)

are.Subtotals(x)

are.Heading(x)

Heading(name, position = c("relative", "top", "bottom"), after = NULL)

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

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

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

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

Arguments

name character the name of the subtotal or heading
### Adding Subtotals and Headings

Subtotals and headings can be added either by passing a list of Subtotals or Headings, or they can be added one at a time by passing `Subtotal` or `Heading` to `subtotals(variable)` alone.

Adding subtotals or headings is additive; meaning that subtotals or headings that are already set on the variable are not removed when new subtotals or headings are added. To remove all subtotals and headings, set `subtotals(variable)` to `NULL`.

To get an array of just the subtotal rows from a CrunchCube, use the function `subtotalArray(CrunchCube)`.

### Working with Subtotals and Headings

When interacting programmatically with Subtotals and Headings, it can be useful to be able to tell if something is a Subtotal or a Heading. The `is.*` family of methods are useful here: the singular versions (`is.Subtotal` and `is.Heading`) take a single object and returns `TRUE` if the object is either a Subtotal or a Heading and `FALSE` if not; the plural versions (`are.Subtotals` and `are.Headings`) take a list of objects (including an Insertions object) and returns a vector of `TRUE/FALSEs`.

### Removing transforms

`noTransforms()` is useful if you don’t want to see or use any transformations like Subtotals and Headings. This action only applies to the CrunchCube object in R: it doesn’t actually change the variables on Crunch servers or the query that generated the CrunchCube.

### Examples

```r
## Not run:
# given a variable ds$opinion, with categories: Strongly Agree, Somewhat Agree, Neither Agree nor Disagree, Somewhat Disagree, and Strongly Disagree,
# to make two subtotals for Agree and Disagree:
subtotals(ds$opinion) <- list(
    Subtotal(name = "Agree",
    Subtotal(name = "Disagree",
```

### Details

To see the subtotals or headings set for a variable, use `subtotals(variable)`

---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>categories</strong></td>
<td>character or numeric the category names or ids for subtotal only</td>
</tr>
<tr>
<td><strong>position</strong></td>
<td>character one of &quot;relative&quot;, &quot;top&quot;, or &quot;bottom&quot;. Determines the position of the subtotal or heading, either at the top, bottom, or relative to another category in the cube (default).</td>
</tr>
<tr>
<td><strong>after</strong></td>
<td>character or numeric if position is &quot;relative&quot;, then the category name or id to position the subtotal or heading after. If not supplied this defaults to the last of the categories supplied to Subtotal.</td>
</tr>
<tr>
<td><strong>x</strong></td>
<td>either a variable or CrunchCube object to add or get subtotal transforms for, for is.Subtotal() and is.Heading() an object to test if it is either a Subtotal or Heading</td>
</tr>
<tr>
<td><strong>value</strong></td>
<td>For <code>&lt;</code>-, the replacement Subtotal to insert</td>
</tr>
<tr>
<td><strong>...</strong></td>
<td>additional arguments to <code>[]</code>, ignored</td>
</tr>
</tbody>
</table>
categories = c("Strongly Agree", "Somewhat Agree"),
after = "Somewhat Agree",
Subtotal(name = "Disagree",
categories = c("Strongly Disagree", "Somewhat Disagree"),
after = "Strongly Disagree")
)

# headings can also be added:
subtotals(ds$opinion) <- Heading(name = "All opinions", position = "top")

# to see the subtotals and headings associated with a variable
subtotals(ds$opinion)

# anchor name func args
# 1 2 Agree subtotal 1 and 2
# 2 4 Disagree subtotal 4 and 5
# 3 0 All opinions <NA> NA

# when you use a variable with subtotals and headings in a cube, you see them
# by default
opinion_cube <- ctabs(~opinion, ds)
opinion_cube

# All opinions
# Strongly Agree 23
# Somewhat Agree 24
# Agree 47
# Neither Agree nor Disagree 18
# Somewhat Disagree 16
# Strongly Disagree 19
# Disagree 35

# to get just the subtotals,
subtotalArray(opinion_cube)

# Agree Disagree
# 47 35

# to remove all subtotals and headings
subtotals(ds$opinion) <- NULL
ctabs(~opinion, ds)

# Strongly Agree 23
# Somewhat Agree 24
# Neither Agree nor Disagree 18
# Somewhat Disagree 16
# Strongly Disagree 19

# if you want to temporarily remove subtotals and headings, you can with `noTransforms`
noTransforms(ctabs(~opinion, ds))

# Strongly Agree Somewhat Agree Neither Agree nor Disagree
# 23 24 18
# Somewhat Disagree Strongly Disagree
# 16 19

## End(Not run)
**subtotalArray, CrunchCube-method**

*Calculate the transforms from a CrunchCube*

**Description**
applyTransforms calculates transforms (e.g. Subtotals) on a CrunchCube. Currently only the row transforms are supported. This is useful if you want to use the values from the subtotals of the CrunchCube in an analysis.

**Usage**
```r
## S4 method for signature 'CrunchCube'
subtotalArray(x, headings = FALSE)
```

```r
applyTransforms(x, array = cubeToArray(showMissing(x)),
   transforms_list = transforms(x), dims_list = dimensions(x),
   useNA = x@useNA, ...)
```

**Arguments**
- `x` : a CrunchCube
- `headings` : for subtotalArray: a logical indicating if the headings should be included with the subtotals (default: FALSE)
- `array` : an array to use, if not using the default array from the cube itself. (Default: not used, pulls an array from the cube directly)
- `transforms_list` : list of transforms to be applied (default: transforms(x))
- `dims_list` : list of dimensions that correspond to array (default: dimensions(x))
- `useNA` : useNA parameter from the CrunchCube to use (default: x@useNA)
- `...` : arguments to pass to calcTransforms, for example include

**Details**
Including the include argument allows you to specify which parts of the CrunchCube to return. The options can be any of the following: "cube_cells" for the untransformed values from the cube itself, "subtotals" for the subtotal insertions, and "headings" for any additional headings. Any combination of these can be given, by default all will be given.

subtotalArray(cube) is a convenience function that is equivalent to applyTransforms(cube, include = c("subtotals"))

**Value**
an array with any transformations applied
Examples

```r
## Not run:
# to get an array of just the subtotals
subtotalArray(crtabs(~opinion, ds))
#   Agree Disagree
#    47    35

# to get the full array with the subtotals but not headings
applyTransforms(crtabs(~opinion, ds), include = c("cube_cells", "subtotals"))
#                Strongly Agree   Somewhat Agree Agree
#               23                24     47
# Neither Agree nor Disagree Strongly Disagree Disagree
#               18                19     35
#     Somewhat Disagree
#               16

# to get the full array with the headings but not subtotals
applyTransforms(crtabs(~opinion, ds), include = c("cube_cells", "headings"))
#                All opinions Strongly Agree   Somewhat Agree
#               NA                23     24
# Neither Agree nor Disagree Strongly Disagree Somewhat Disagree
#               18                19     16

## End(Not run)
```

Subvariables-class  

Subvariables in Array Variables

Description

Multiple-response and categorical-array variables are higher order variables which are made up of sets of subvariables. These methods allow you to retrieve and change the subvariables of a multiple-response or categorical-array variable.

Usage

```r
## 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")`

---

```r
subvars-extract Extract and modify subsets of subvariables
```

Description

Extract and modify subsets of subvariables

Usage

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

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

## S4 method for signature 'CategoricalArrayVariable,missing,character'
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 containing subvariables
- **i**: Subvariables to extract
- **j**: 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

Description

Just like subtotals(), summary statistics can be inserted into cubes. `SummaryStat()` makes an object of type `SummaryStat` which can be added on to a CrunchCube's insertions to add the specified summary statistic. Currently only mean and median are supported; both use weighted algorithms to go from counts and numeric values of categories to the expected statistic. Although `SummaryStat` objects can be made by hand, it is recommended instead to use the `addSummaryStat()` function which is much quicker and easier to simply add a summary statistic to an existing CrunchCube.

Usage

```r
SummaryStat(name = "name", stat = "mean", categories = NULL, position = c("relative", "top", "bottom"), after = NULL, includeNA = FALSE)
```

```r
SummaryStat(name = "name", stat = "median", categories = NULL, position = c("relative", "top", "bottom"), after = NULL, includeNA = FALSE)
```

```r
is.SummaryStat(x)
```

```r
are.SummaryStats(x)
```

Arguments

- **name**: character the name of the summary statistic
- **stat**: a function to calculate the summary (e.g. mean or median)
- **categories**: character or numeric the category names or ids to be included in the summary statistic, if empty all categories
position character one of "relative", "top", or "bottom". Determines the position of the subtotal or heading, either at the top, bottom, or relative to another category in the cube (default)

after character or numeric if position is "relative", then the category name or id to position the subtotal or heading after

includeNA should missing categories be included in the summary?

x for is.SummaryStat() only, an object to test if it is a SummaryStat object

Details

Summary statistics are intended only for CrunchCube objects, and are not able to be set on Crunch variables.

Removing transforms

noTransforms() is useful if you don’t want to see or use any transformations like Subtotals and Headings. This action only applies to the CrunchCube object in R: it doesn’t actually change the variables on Crunch servers or the query that generated the CrunchCube.

Description

This function allows you to generate a tab book from a multitable and data. As with other functions, you can select the rows and columns you want to work with by subsetting the dataset you pass into the function.

Usage

```r
tabBook(multitable, dataset, weight = crunch::weight(dataset),
          format = c("json", "xlsx"), file, ...)
```

Arguments

- **multitable** a Multitable object
- **dataset** CrunchDataset, which may have been subsetted with a filter expression on the rows and a selection of variables on the columns.
- **weight** a CrunchVariable that has been designated as a potential weight variable for dataset, or NULL for unweighted results. Default is the currently applied weight.
- **format** character export format: currently supported values are "json" (default) and "xlsx".
- **file** character local filename to write to. A default filename will be generated from the multitable’s name if one is not supplied and the "xlsx" format is requested. Not required for "json" format export.
- **...** Additional "options" passed to the tab book POST request.
Details

By specifying a "json" format, instead of generating an Excel workbook, you'll get a TabBookResult object, containing nested CrunchCube results. You can then further format these and construct custom tab reports.

Value

If "json" format is requested, the function returns an object of class TabBookResult, containing a list of MultitableResult objects, which themselves contain CrunchCubes. If "xlsx" is requested, the function invisibly returns the filename (file, if specified, or the the autogenerated file name). If you request "json" and wish to access the JSON data underlying the TabBookResult, pass in a path for file and you will get a JSON file written there as well.

Examples

```r
## Not run:
m <- newMultitable(~ gender + age4 + marstat, data=ds)
tabBook(m, ds[ds$income > 100000], format="xlsx", file="wealthy-tab-book.xlsx")
book <- tabBook(m, ds)  # Returns a TabBookResult
tables <- prop.table(book, 2)

## End(Not run)
```

Description

TabBookResult and MultitableResult methods

Usage

```r
## S4 method for signature 'TabBookResult'
length(x)

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

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

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

## S4 method for signature 'TabBookResult'
names(x)
```
## Arguments

- **x**: a `TabBookResult` or `MultitableResult`
- **i**: an index into `x`
- **j**: an index into `x`, ignored
- **...**: also ignored

## Value

Returns what you’d expect.

---

**table** | **Table function for Crunch objects**

---

**Description**

Table function for Crunch objects

**Usage**

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

Arguments

... CrunchVariables to tabulate
exclude see base::table
useNA see base::table
dnn see base::table
deparse.level see base::table

Value

a table object

See Also

base::table

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
### Description

`crunch` uses the `jsonlite` package for JSON serialization and deserialization. Unfortunately, `jsonlite::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, so you can define methods for it if you have other objects that you want to successfully serialize to JSON.

### Usage

```r
jsonprep(x, ...)
```

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

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

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

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

```r
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

`jsonlite::toJSON()`
**toVariable**

*Generic method for converting objects to Crunch representations*

**Description**

R objects are converted to Crunch objects using the following rules:

**Usage**

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

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

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

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

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

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

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

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

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

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

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

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

toVariable(x, ...)

**Arguments**

- **x**: An R vector you want to turn into a Crunch variable
- **...**: Additional metadata fields for the variable, such as "name" and "description". See the API documentation for a complete list of valid attributes.

**Details**

- Character vectors are converted into Crunch text variables
- Numeric vectors are converted into Crunch numeric variables
- Factors are converted to categorical variables
- Date and POSIXt vectors are converted into Crunch datetime variables
- Logical vectors are converted to Crunch categorical variables
- `VariableDefinition()`s are not converted, but the function can still append additional metadata

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

**Value**

A `VariableDefinition` object. To add this to a dataset, either assign it into the dataset (like `ds$newvar <- toVariable(...)`) or call `addVariables()`. If you're adding a column of data to a dataset, it must be as long as the number of rows in the dataset, or it may be a single value to be recycled for all rows.

**See Also**

`VariableDefinition()` `addVariables()`

**Examples**

```r
var1 <- rnorm(10)
toVariable(var1)
toVariable(var1, name="Random", description="Generated in R")
## Not run:
ds$random <- toVariable(var1, name="Random")
# Or, this way:
ds <- addVariables(ds, toVariable(var1, name="Random"))
```

## End(Not run)
Transformations allow you to change how a variable or cube is displayed without changing the underlying data.

Usage

transforms(..., data = NULL)

## S4 method for signature 'CrunchCube'
transforms(x)

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

## S4 replacement method for signature 'CrunchCube,list'
transforms(x) <- value

## S4 replacement method for signature 'CrunchCube,`NULL`'
transforms(x) <- value

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

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

## S4 replacement method for signature 'CrunchVariable,Transforms'
transforms(x) <- value

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

Arguments

... For the constructor function Transforms you can pass in attributes via ...
data For the constructor function Transforms 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 Transforms
value For [<-, the replacement Transforms to insert
Getting transformations

The \texttt{transforms(x)} methods can be used with Variables and CrunchCubes to get what transformations are currently set. For variables, they return a single \texttt{Transforms} object that includes all transformations for the variable. For CrunchCubes, it returns a named list with the same length as the number of dimensions of the cube with each dimension’s transformations.

Currently, \texttt{Insertions} (e.g. \texttt{Subtotal()} and \texttt{Heading()}) are the only type of transformations that are supported.

Setting transformations on a variable

The \texttt{transforms(x) <- value} methods can be used to assign transformations for a specific variable. \texttt{value} must be a \texttt{Transforms} object. This allows you to set transformations on categorical variables. These transformations will automatically show up in any new CrunchCubes that contain this variable.

Setting transformations on a CrunchCube

The \texttt{transforms(x) <- value} methods can also be used to assign transformations to a CrunchCube that has already been calculated. \texttt{value} must be a named list of \texttt{Transforms} objects. The names of this list must correspond to dimensions in the cube (those dimensions correspondences are matched based on variable aliases). You don’t have to provide an entry for each dimension, but any dimension you do provide will be overwritten fully.

Removing transformations

To remove transformations from a variable or CrunchCube, use \texttt{transforms(x) <- NULL}.

\begin{center}
\begin{tabular}{ll}
\texttt{type} & \textit{Change Crunch variable types} \\
\end{tabular}
\end{center}

Description

Numeric, text, and categorical variables can be cast to one another by assigning them a new "type". This modifies 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

\begin{verbatim}
## S4 method for signature 'CrunchVariable'
type(x)

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

## S4 replacement method for signature 'CrunchVariable'
type(x) <- value
\end{verbatim}
unbind

Arguments

\[
\begin{align*}
x & \quad \text{a Variable} \\
\text{value} & \quad \text{For the setter, a character value in c("numeric", "text", "categorical")}
\end{align*}
\]

Value

Getter returns character; setter returns x duly modified.

Description

Split an array or multiple-response variable into its CategoricalVariables

Usage

unbind(x)

Arguments

\[
\begin{align*}
x & \quad \text{a CategoricalArrayVariable or MultipleResponseVariable}
\end{align*}
\]

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 \quad \text{a CrunchDataset}

users \quad \text{character: email address(es) or URLs of the users or teams to unshare with.}
Value

Invisibly, the dataset.

See Also

share

users

Get information about users who have access to a dataset

Description

Get user metadata about all of the users that have access to a particular Crunch object like a dataset or project. Returns a UserCatalog object which can be translated into a data.frame with catalogToDataFrame() if information needs to be extracted, queried, transformed, etc.

Usage

```r
## S4 method for signature 'CrunchDataset'
users(x)

## S4 method for signature 'DatasetTuple'
users(x)

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

Arguments

- `x` a CrunchDataset, DatasetTuple, or CrunchProject object to get users from

Value

a UserCatalog with information about users who have access to the dataset
Usage

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

## 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-as-methods  
as.* methods for variables

Description

Use the as.* family of functions to make a derived copy of a variable that has been converted into a new type.

Usage

```r
## S4 method for signature 'CrunchVariable'
as.Numeric(x)

## S4 method for signature 'CrunchVariable'
as.Text(x, format)

## S4 method for signature 'CrunchVariable'
as.Categorical(x, format)

## S4 method for signature 'CrunchVariable'
as.Datetime(x,    
    format = "%Y-%m-%d %H:%M:%S", resolution, offset)

## S3 method for class 'CrunchVariable'
as.double(x, ...)

## S3 method for class 'CrunchVariable'
as.character(x, ...)
```

Arguments

- **x**  
  a Crunch variable to derive and convert to a new type

- **format**  
  for as.Datetime, when the variable in x is a text or categorical variable, format is the typographical format that the datetime is already formatted in that needs to be parse from (default: "%Y-%m-%d %H:%M:%S"); for as.Text and as.Categorical, is the typographical format that the datetime is to be formatted as (e.g. "%Y-%m-%d %H:%M:%S" for "2018-01-08 12:39:57", the default if no rollup resolution is specified on the source variable. If a rollup resolution is specified, a reasonable default will be used.).

- **resolution**  
  for as.Datetime, when the variable in x is a numeric variable, the resolution of the number (e.g. "ms" for milliseconds, "s" for seconds, etc. see expressions for more information about valid values.)
offset for as.Datetime, when the variable in x is a numeric the, a character of the offset to count from in the shape "2018-01-08 12:39:57". If not supplied, Crunch’s default of 1970-01-01 00:00:00 will be used.

... additional arguments for as.character and as.numeric, ignored when used with Crunch variables

Details

Each type of Crunch variable (text, numeric, categorical, etc.) has an as.* function (as.Text, as.Numeric, and as.Categorical respectively) that takes the input given as x, and makes a new derived variable that is now of the type specified. See below for detailed examples.

For as.Text and as.Numeric, aliases to the R-native functions as.character and as.numeric are provided for convenience.

Value

a VariableDefinition to be used as the derivation

Examples

```r
## Not run:
# ds$v1 is of type Text
is.Text(ds$v1)
# [1] TRUE

# that has strings of numbers
as.vector(ds$v1)
# [1] "32" "8" "4096" "1024"

# convert this to a numeric variable with the alias `v1_numeric`
ds$v1_numeric <- as.Numeric(ds$v1)

# the values are the same, but are now numerics and the type is Numeric
as.vector(ds$v1_numeric)
# [1] 32 8 4096 1024
is.Numeric(ds$v1_numeric)
# [1] TRUE

# this new variable is derived, so if new data is appended or streamed, the
# new rows of data will be updated.
is.derived(ds$v1_numeric)
# [1] TRUE

## End(Not run)
```
variable-update

Update variables with expressions or values

Description
Update 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,ANY,missing,CrunchVariable'
x[i, j] <- value

## S4 replacement method for signature 'CrunchVariable,ANY,missing,CrunchLogicalExpr'
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
```
VariableCatalog-class

\begin{verbatim}
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
\end{verbatim}

Arguments

\begin{verbatim}
x            a Variable
i            a CrunchLogicalExpr or R index, optionally
j            Invalid
value        an R vector or a CrunchExpr with which to update
\end{verbatim}

Value

\begin{verbatim}
x duly modified
\end{verbatim}

---

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. Each VariableCatalog also contains a VariableOrder that governs how variables within it are organized.
Construct a variable definition

**Description**

Crunch variables are created by posting a `VariableDefinition` to the Crunch server. The `VariableDefinition` contains the information the server requires to calculate the variable. This can include information in the form of the actual data which you would like to include in the variable, or a derivation which tells the server how to derive the new variable from existing ones. This function converts an R vector or set of attributes into a variable definition which can be posted to the server.

**Usage**

```r
VariableDefinition(data, ...)  
VarDef(data, ...)
```

**Arguments**

- `data`: an R vector of data to convert to the Crunch payload format. See `toVariable` for how R data types are converted. This function can also be used to construct a `VariableDefinition` directly by passing attributes to `...`. This is only recommended for advanced users who are familiar with the Crunch API.
- `...`: additional attributes to be included in the `VariableDefinition`.

**Value**

A `VariableDefinition` object, ready to POST to Crunch.

**See Also**

toVariable

**Examples**

```r
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
Crunch stores variable information in several catalogs containing information about the variable class, its missingness and subvariables. This function allows you to access that information.

Usage
variableMetadata(dataset)

Arguments
dataset  CrunchDataset

Value
A VariableCatalog with all variable properties, including categories and subvariables.

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.
Access a catalog of variables

Description

Datasets contain collections of variables. For some purposes, such as editing variables’ metadata, it is helpful to access these variable catalogs more directly. Other objects, such as cubes and folders, also define variables() methods that expose variable metadata.

Usage

variables(x)

variables(x) <- value

allVariables(x)

allVariables(x) <- value

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

## S4 method for signature 'CrunchCube'
variables(x)

## 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

## S4 method for signature 'SearchResults'
variables(x)

## S4 method for signature 'VariableFolder'
variables(x)

Arguments

x a Dataset

value For the setters, a VariableCatalog to assign.
Details

For datasets, variables() returns only the active variables in the dataset, while allVariables() returns all variables, including hidden variables. allVariables() is not defined for other objects.

Value

All methods return a VariableCatalog except the VariableFolder method, which returns a subset of x containing only variable references. Assignment functions return x with the changes made.

versions

Access the saved versions of a dataset

Description

This function allows you to see a dataset’s savepoints. These can then be passed to restoreVersion() to load the previously saved version 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

webApp

View a Crunch Object in the Web Application

Description

Convenience function that will use your system’s "open" command to open a Crunch object in our web application in your default browser.

Usage

webApp(x)
Arguments

\texttt{x} \hspace{1cm} \text{a Crunch Dataset or Variable}

Value

Nothing; called for side effect of opening your web browser.

\begin{center}
\begin{tabular}{ll}
\texttt{weight} & \textit{Dataset weights} \\
\end{tabular}
\end{center}

Description

\texttt{weight} lets you view and set your user’s currently applied weight on the server. \texttt{weightVariables} lets you view all of the variables that have been designated as valid to use as weights.

Usage

\begin{verbatim}
weight(x)

weight(x) <- value

is.weight(x)

## S4 replacement method for signature 'NumericVariable'

is.weight(x) <- value
\end{verbatim}

Arguments

\begin{verbatim}
  x \hspace{1cm} \text{a Dataset}
  value \hspace{1cm} \text{a Variable, VariableDefinition, or NULL. If a VariableDefinition is passed, the variable will first be created and then set as the datasets weight. Set to NULL to remove existing weights from the dataset.}
\end{verbatim}

Value

For the weight getter, a Variable if there is a weight, else NULL. For the setter, \texttt{x}, modified accordingly. \texttt{weightVariables} returns the aliases (or names, according to \texttt{options(crunch.namekey.dataset)}), of the variables designated as weights.

See Also

\texttt{weightVariables()} \texttt{makeWeight()}

weightVariables

Get a dataset's weightVariables

Description

Get a dataset's weightVariables

Usage

```r
## S4 method for signature 'CrunchDataset'
weightVariables(x)

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

Arguments

- `x`: a CrunchDataset

Value

weightVariables returns a character vector of the aliases of the variables that are eligible to be used as weights.

See Also

weight(), makeWeight(), modifyWeightVariables()

which

"which" method for CrunchLogicalExpr

Description

"which" method for CrunchLogicalExpr

Usage

```r
## S4 method for signature 'CrunchLogicalExpr'
which(x, arr.ind = FALSE, useNames = TRUE)
```

Arguments

- `x`: CrunchLogicalExpr
- `arr.ind`: Ignored
- `useNames`: Ignored
Value

Integer row indices where \( x \) is true. Note that this does not return a Crunch expression. Use this when you need to translate to R values. For filtering a Crunch expression by \( x \), don’t use \texttt{which}.

\[
\text{with-context-manager} \quad \text{Context manager’s "with" method}
\]

Description

Context manager’s "with" method

Usage

\[
\texttt{## S3 method for class 'contextManager'} \\
\texttt{with(data, expr, \ldots)}
\]

Arguments

\[
\begin{align*}
\text{data} & \quad \texttt{contextManager} \\
\text{expr} & \quad \text{code to evaluate within that context} \\
\ldots & \quad \text{additional arguments. One additional supported argument is "as", which lets you assign the return of your "enter" function to an object you can access.}
\end{align*}
\]

Value

Nothing.

See Also

\texttt{ContextManager}

\[
\texttt{write.csv.gz} \quad \text{Write CSV to a compressed file}
\]

Description

Write CSV to a compressed file

Usage

\[
\texttt{write.csv.gz(x, file, na = "", row.names = FALSE, \ldots)}
\]
Arguments

- **x**: A data.frame or similar CSV-writable object
- **file**: character destination to write the gzipped CSV to
- **na**: See `utils::write.csv`. This just changes the default to a Crunch-friendly empty string.
- **row.names**: logical: write out row names? See `utils::write.csv`.
- **...**: Additional arguments passed to `write.csv`.

Value

A csv file written to dist

Description

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

Usage

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

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

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

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

## 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`

---

```
$.AbstractCategory-method

Access Category fields directly
```

**Description**

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

**Usage**

```r
## S4 method for signature 'AbstractCategory'
x$name

## S4 replacement method for signature 'AbstractCategory'
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`
Index

!,CrunchExpr-method (expressions), 66
!=,CategoricalVariable, character-method (expressions), 66
!=,CategoricalVariable, factor-method (expressions), 66
!=,CategoricalVariable, numeric-method (expressions), 66

*Topic datasets

SO_schema, 139
SO_survey, 140
.
.Insertion (Insertions-class), 77
==,CategoricalVariable, character-method (expressions), 66
==,CategoricalVariable, factor-method (expressions), 66
==,CategoricalVariable, numeric-method (expressions), 66
[],AbstractCategories, ANY, ANY-method (Categories-class), 21
[],AbstractCategories, character, ANY-method (Categories-class), 21
[],AbstractCategories, numeric, ANY-method (Categories-class), 21
[],CategoricalArrayVariable, character, ANY-method (subvars-extract), 147
[],CategoricalArrayVariable, missing, ANY-method (Subvars-extract), 147
[],CategoricalArrayVariable, missing, character-method (Subvars-extract), 147
[],CrunchCube, ANY, ANY-method (cube-methods), 41
[],CrunchDataset, ANY, ANY-method (dataset-extract), 45
[],CrunchDataset, CrunchLogicalExpr, ANY-method (dataset-extract), 45
[],CrunchDataset, CrunchLogicalExpr, missing-method (dataset-extract), 45
[],CrunchDataset, VariableGroup, ANY-method (dataset-extract), 45

[],CrunchExpr, CrunchLogicalExpr, ANY-method, 173
[],CrunchExpr, logical, ANY-method ([,CrunchExpr, CrunchLogicalExpr, ANY-method), 173
[],CrunchExpr, numeric, ANY-method ([,CrunchExpr, CrunchLogicalExpr, ANY-method), 173
[],CrunchVariable, CrunchExpr, ANY-method ([,CrunchExpr, CrunchLogicalExpr, ANY-method), 173
[],CrunchVariable, logical, ANY-method ([,CrunchExpr, CrunchLogicalExpr, ANY-method), 173
[],CrunchVariable, numeric, ANY-method ([,CrunchExpr, CrunchLogicalExpr, ANY-method), 173

[],CrunchVariable, numeric, ANY-method ([,CrunchExpr, CrunchLogicalExpr, ANY-method), 173

[,ShojiCatalog, ANY, ANY-method (catalog-extract), 17
[,ShojiCatalog, character, ANY-method (catalog-extract), 132
[,ShojiCatalog, logical, ANY-method (catalog-extract), 17

(catalog-extract), 17
[[-, ProjectCatalog, numeric-method (catalog-extract), 17
[[-, Session, ANY-method (session-extract), 127
[[-, ShojiCatalog, ANY-method (catalog-extract), 17
[[-, ShojiCatalog, character-method (catalog-extract), 17
[[-, ShojiFolder, character-method (catalog-extract), 17
[[-, ShojiFolder, numeric-method (catalog-extract), 17
[[-, ShojiOrder, ANY-method (ShojiOrder-extract), 132
[[-, ShojiOrder, character-method (ShojiOrder-extract), 132
[[-, Subvariables, character-method (subvars-extract), 147
[[-, Subvariables, numeric-method (subvars-extract), 147
[[-, TabBookResult, character-method (tabbook-methods), 151
[[-, TabBookResult, numeric-method (tabbook-methods), 151
[[-, TeamCatalog, numeric-method (catalog-extract), 17
[[-, UserCatalog, character-method (catalog-extract), 17
[[-, VariableCatalog, numeric-method (catalog-extract), 17
[[-, CrunchDataFrame (CrunchDataFrame), 38
[[-, CrunchDataFrame, character, ANY, ANY-method (Categories-class), 21
[[-, CategoricalArrayVariable, ANY, missing, ANY-method (shojiOrder-extract), 132
[[-, CategoricalArrayVariable, character, missing, ANY-method (shojiOrder-extract), 132
[[-, CrunchDataset, ANY, ANY, ANY-method (dataset-update), 48
[[-, CrunchDataset, ANY, missing, CrunchGeography-method (dataset-update), 48
[[-, CrunchDataset, character, missing, CrunchLogicalExpr-method (dataset-update), 48
[[-, CrunchDataset, character, missing, CrunchVariable-method (dataset-update), 48
[[-, CrunchDataset, character, missing, NULL-method (dataset-update), 48
[[-, DatasetCatalog, character, missing, DatasetTuple-method (catalog-extract), 17
[[-, FilterCatalog, character, missing, CrunchFilter-method (catalog-extract), 17
[[-, FilterCatalog, character, missing, CrunchLogicalExpr-method (catalog-extract), 17
[[-, FilterCatalog, numeric, missing, CrunchFilter-method (catalog-extract), 17
[[-, FilterCatalog, numeric, missing, CrunchLogicalExpr-method (catalog-extract), 17
[[-, MemberCatalog, ANY, missing, ANY-method (catalog-extract), 17
[[-, MemberCatalog, character, missing, NULL-method (catalog-extract), 17
[[-, MultitableCatalog, ANY, missing, NULL-method (catalog-extract), 17
[[-, MultitableCatalog, character, missing, formula-method (catalog-extract), 17
[[-, MultitableCatalog, numeric, missing, formula-method (catalog-extract), 17
[[-, OrderGroup, ANY, missing, OrderGroup-method (ShojiOrder-extract), 132
[[-, OrderGroup, character, missing, NULL-method (ShojiOrder-extract), 132
[[-, OrderGroup, character, missing, list-method (ShojiOrder-extract), 132
[[-, OrderGroup, numeric, missing, NULL-method (ShojiOrder-extract), 132
[[-, OrderGroup, numeric, missing, list-method (ShojiOrder-extract), 132
[[-, ProjectCatalog, character, missing, list-method (catalog-extract), 17
[[-, ProjectCatalog, character, missing, list-method (catalog-extract), 17
[[-, Session, ANY, ANY, ANY-method (session-extract), 127
[[<-, ShojiOrder ANY, missing, ANY-method
(ShojiOrder-extract), 132
[[<-, ShojiOrder ANY, missing, NULL-method
(ShojiOrder-extract), 132
[[<-, ShojiOrder ANY, missing, OrderGroup-method
(ShojiOrder-extract), 132
[[<-, ShojiOrder character, missing, NULL-method
(ShojiOrder-extract), 132
[[<-, ShojiOrder character, missing, OrderGroup-method
(ShojiOrder-extract), 132
[[<-, ShojiOrder character, missing, character-method
(ShojiOrder-extract), 132
[[<-, ShojiOrder character, missing, list-method
(ShojiOrder-extract), 132
[[<-, Subvariables ANY, missing, any-method
(subvars-extract), 147
[[<-, Subvariables ANY, missing, CrunchVariable-method
(subvars-extract), 147
[[<-, Subvariables ANY, missing, NULL-method
(subvars-extract), 147
[[<-, Subvariables character, missing, CrunchVariable-method
(subvars-extract), 147
[[<-, TeamCatalog character, missing, CrunchTeam-method
(catalog-extract), 17
[[<-, TeamCatalog character, missing, list-method
(catalog-extract), 17
[[<-, VariableCatalog character, missing, CrunchVariable-method
(catalog-extract), 17
[[<-, VariableCatalog character, missing, VariableTuple-method
(catalog-extract), 17
[[<-, VariableGroup character, missing, CrunchDataset-method
(ShojiOrder-extract), 132
[[<-, VariableOrder character, missing, CrunchDataset-method
(ShojiOrder-extract), 132
[[<-. CrunchDataFrame (CrunchDataFrame), 38
$,. AbstractCategory-method, 174
$,. CategoricalArrayVariable-method
(subvars-extract), 147
$,. CrunchDataset-method
(dataset-update), 48
$,. OrderGroup-method
(ShojiOrder-extract), 132
$,. Session-method (session-extract), 127
$,. ShojiCatalog-method
(catalog-extract), 17
$,. ShojiOrder-method
(ShojiOrder-extract), 132
$,. CrunchDataFrame (CrunchDataFrame), 38
$,-, AddGeoMetadata, 6
$,-, AddSubvariable, 7
$,-, AddSumStat, 8
$,-, AddVariables, 10, 69
$,-, Alias (describe), 57
$,-, Alias, CrunchVariable-method (describe), 57
$,-, Alias, VariableTuple-method (describe), 57
$,-, Alias, (describe), 57


alias<-CrunchVariable-method 
(describe), 57
aliases(names,BatchCatalog-method), 106
aliases,CrunchCube-method 
(cube-methods), 41
aliases,MultiTableResult-method 
(tabbook-methods), 151
aliases,TabBookResult-method 
(tabbook-methods), 151
aliases,VariableCatalog-method 
(names,BatchCatalog-method), 106
aliases,VariableFolder-method 
(names,BatchCatalog-method), 106
aliases<-(names,BatchCatalog-method), 106
allVariables(variables), 168
allVariables,CrunchDataset-method 
(variables), 168
allVariables<-(variables), 168
anchor(Insertions-class), 77
anchor,Heading-method 
(Insertions-class), 77
anchor,Insertion-method 
(Insertions-class), 77
anchor,Subtotal-method 
(Insertions-class), 77
anchor,SummaryStat-method 
(Insertions-class), 77
anchor<-(Insertions-class), 77
anchor<-,Heading-method 
(Insertions-class), 77
anchor<-,Insertion-method 
(Insertions-class), 77
anchor<-,Subtotal-method 
(Insertions-class), 77
anchor<-,SummaryStat-method 
(Insertions-class), 77
anchors(Insertions-class), 77
appendDataset, 11
appendDataset(), 124
appendStream, 12
appendStream(), 114
applyTransforms 
(subtotalArray,CrunchCube-method), 145
archive 
(is.archived,DatasetCatalog-method), 83
archive-and-publish 
(is.archived,DatasetCatalog-method), 83
are.Headings(Subtotal-class), 142
are.Subtotals(Subtotal-class), 142
are.SummaryStats(SummaryStat-class), 149
arguments(Insertions-class), 77
arguments,Heading-method 
(Insertions-class), 77
arguments,Insertion-method 
(Insertions-class), 77
arguments,Subtotal-method 
(Insertions-class), 77
arguments,SummaryStat-method 
(Insertions-class), 77
as.Categorical(variable-as-methods), 162
as.Categorical,CrunchVariable-method 
(variable-as-methods), 162
as.character.CrunchVariable 
(variable-as-methods), 162
as.data.frame, 14
as.data.frame.BatchCatalog 
(catalog-dataframes), 15
as.data.frame.CrunchDataFrame 
(dataset-to-R), 47
as.data.frame.CrunchDataset 
(dataset-to-R), 47
as.data.frame.FilterCatalog
(catalog-dataframes), 15
as.data.frame.ProjectCatalog
catalog-dataframes), 15
as.data.frame.ShojiCatalog
catalog-dataframes), 15
as.data.frame.UserCatalog
catalog-dataframes), 15
as.data.frame.VariableCatalog
catalog-dataframes), 15
as.Datetime (variable-as-methods), 162
as.Datetime, CrunchVariable-method
(variable-as-methods), 162
as.double, CrunchVariable
(variable-as-methods), 162
as.environment, CrunchDataset-method,
12
as.Numeric (variable-as-methods), 162
as.Numeric, CrunchVariable-method
(variable-as-methods), 162
as.Text (variable-as-methods), 162
as.Text, CrunchVariable-method
(variable-as-methods), 162
as.vector(), 48
as.vector, CrunchExpr-method, 13
as.vector, CrunchVariable-method
(as.vector, CrunchExpr-method), 13
availableGeodata
crunchgeography-class), 40
availableGeodataFeatures, 14
base::array, 43
base::dim(), 61
base::dir.create(), 104
base::duplicate, 51
base::duplicate(), 62
base::file.remove(), 123
base::levels(), 23
base::margin.table(), 119, 121
base::match(), 67
base::max(), 37
base::mean(), 37
base::merge(), 87
base::min(), 37
base::names(), 108
base::prop.table(), 119–121
base::round(), 120
base::setwd(), 24
base::table, 153
base:::table(), 34
bases (prop.table, CrunchCube-method),
119
bases, CrunchCube-method
(prop.table, CrunchCube-method),
119
bases, MultitableResult-method
(prop.table, CrunchCube-method),
119
bases, TabBookResult-method
(prop.table, CrunchCube-method),
119
batches, 14
bin (expressions), 66
c, 15
c-categories, 15
c.Categories (c-categories), 15
c.Category (c-categories), 15
catalog-dataframes, 15
catalog-extract, 17
catalog-length, 20
catalogToDataFrame(), 160
CategoricalArrayVariable
crunchvariable-class), 41
CategoricalArrayVariable-class
(CrunchVariable-class), 41
CategoricalVariable
(CrunchVariable-class), 41
CategoricalVariable-class
(CrunchVariable-class), 41
Categories, 15, 59, 106, 108
Categories (Categories-class), 21
categories (var-categories), 160
categories(), 79
categories, CategoricalArrayVariable-method
(var-categories), 160
categories, CategoricalVariable-method
(var-categories), 160
categories, CrunchVariable-method
(var-categories), 160
categories, VariableEntity-method
(var-categories), 160
categories, VariableTuple-method
(var-categories), 160
categories-class, 21
categories<= (var-categories), 160
categories<=, CategoricalArrayVariable, ANY-method
(var-categories), 160
INDEX

181

categories<-.CategoricalArrayVariable, Category <-.CategoricalArrayVariable, Category <-.CategoricalArrayVariable, Character <-.CategoricalArrayVariable, CategoricalArrayVariable, Numeric <-.CategoricalArrayVariable, CategoricalVariable, ANY <-.CategoricalArrayVariable, Character <-.CategoricalVariable, Categories <-.CategoricalVariable, Categories <-.CrunchVariable, ANY <-.CrunchVariable, Categories <-.CrunchVariable, Categories <-.CrunchVariable, Categories <-.CrunchVariable, CategoriesFromLevels, 22

Category (Categories-class), 21
Category-class (Categories-class), 21

category-extract

$(\$, AbstractCategory-method), 174
cd, 23
cd(), 70, 104, 123, 128, 129
changeCategoryID, 24
cleanseBatches, 25
collapseCategories, 26
combine, 26
combine(), 26
compareDatasets, 27
compareDatasets(), 11
conditionalTransform, 28
consent, 30
ContextManager, 30, 30, 153, 172
contextManager, 172
contextManager (ContextManager), 30
copy (copyVariable), 32
copy(), 27
copyOrder, 31
copyVariable, 32
crDELETE (http-methods), 75
createDataset, 118
createDataset(), 108
createWithPreparedData, 33
crGET (http-methods), 75
crPATCH (http-methods), 75
crPOST (http-methods), 75
crPUT (http-methods), 75
cube-computing
cube-method

crtabs(), 13
crmethod

crunch-cut, 34
crunch-package (crunch), 34
crunch-uni, 36
crunchAPI, 75
CrunchBox, 117

methodBox (crunchBox), 37
methodBox, 37

CrunchDataFrame, 38

CrunchDataset, 113
CrunchDataset (CrunchDataset-class), 39
CrunchDataset-class, 39
CrunchExpr, 55
CrunchGeography, 6, 7
CrunchGeography

(CrunchGeography-class), 40
CrunchGeography-class, 40

CrunchVariable-class, 41
cut, NumericVariable-method

cut (crunch-cut), 34
dashboard, 44
dashboard<-(dashboard), 44
data.frame, 38
dataset-extract, 45
dataset-owner, 46
dataset-to-R, 47
dataset-update, 48
dataset-variables (variables), 168
DatasetGroup (DatasetOrder-class), 50
DatasetGroup-class

(DatasetOrder-class), 50
DatasetOrder (DatasetOrder-class), 50
DatasetOrder-class, 50
datasets, 50
datasets<-(datasets), 50
DateTimeVariable

(CrunchVariable-class), 41
DateTimeVariable-class

(CrunchVariable-class), 41
dedupeOrder, 51
delete, 52, 53
delete(), 53
delete, ANY-method (delete), 52
delete, CrunchDataset-method (delete), 52
delete, CrunchProject-method (delete), 52
delete, CrunchTeam-method (delete), 52
delete, CrunchVariable-method (delete), 52
delete, Multitable-method (delete), 52
delete, ShojiFolder-method (delete), 52
delete, ShojiObject-method (delete), 52
delete, VariableTuple-method (delete), 52
deleteDataset, 53, 53
deleteSubvariable, 147
deleteSubvariable (deleteSubvariables), 54
deleteSubvariables, 54
deleteVariable (deleteVariables), 54
deleteVariables, 54
 derivation (derivations), 55
derivation, CrunchVariable-method (derivations), 55
derivation<- (derivations), 55
derivation<-, CrunchVariable, ANY-method (derivations), 55
derivation<-, CrunchVariable, NULL-method (derivations), 55
derivations, 55
deriveArray (makeArray), 92
describe, 57
describe= catalog
(names, BatchCatalog-method), 106
describe= category
(name, AbstractCategory-method), 105
description (describe), 57
description, CrunchDataset-method (describe), 57
description, CrunchVariable-method (describe), 57
description, Geodata-method (describe), 57
description, VariableTuple-method (describe), 57
description<-, CrunchDataset-method (describe), 57
description<-, CrunchVariable-method (describe), 57
description<-, NumericVariable-method (describe), 57
dim, CrunchCube-method (cube-methods), 41
dim, CrunchDataset-method (dim-dataset), 60
dim, CubeDims-method (cube-methods), 41
dim, TabBookResult-method
(tabbook-methods), 151
dim-dataset, 60
dim.CrunchDataFrame (CrunchDataFrame), 38
dimensions (cube-methods), 41
dimensions, CrunchCube-method (cube-methods), 41
dimensions<- (cube-methods), 41
dimensions<-, CrunchCube, CubeDims-method (cube-methods), 41
dimnames, CrunchCube-method (cube-methods), 41
dimnames, CubeDims-method (cube-methods), 41
dropRows, 61
duplicated, 62
duplicated, CrunchExpr-method (duplicated), 62
duplicated, CrunchVariable-method (duplicated), 62
duplicates, 51
duplicates (ShojiOrder-slots), 135
duplicates, OrderGroup-method (ShojiOrder-slots), 135
duplicates, ShojiOrder-method (ShojiOrder-slots), 135
duplicates, VariableCatalog-method (ShojiOrder-slots), 135
duplicates<-(ShojiOrder-slots), 135
duplicates<-, OrderGroup, logical-method (ShojiOrder-slots), 135
duplicates<-, ShojiOrder, logical-method (ShojiOrder-slots), 135
duplicates<-, VariableCatalog, logical-method (ShojiOrder-slots), 135
e-mail, 63
e-mail, UserEntity-method (email), 63
emails (names, BatchCatalog-method), 106
emails, ShojiCatalog-method (names, BatchCatalog-method), 106
embedCrunchBox, 63
endDate (describe), 57
endDate, CrunchDataset-method (describe), 57
endDate<-(describe), 57
endDate<-, CrunchDataset-method (describe), 57
entities (ShojiOrder-slots), 135
entities, list-method (ShojiOrder-slots), 135
entities, OrderGroup-method (ShojiOrder-slots), 135
entities, ShojiOrder-method (ShojiOrder-slots), 135
entities<-(ShojiOrder-slots), 135
entities<-, OrderGroup-method (ShojiOrder-slots), 135
dimensions<-, ShojiOrder-method (ShojiOrder-slots), 135
exclusion, 61, 64
exclusion<-(exclusion), 64
exportDataset, 65
exportDataset(), 14
expressions, 66, 162
expropriateUser, 67
extendDataset (joinDatasets), 87
filter-catalog, 68
filters (filter-catalog), 68
filters(), 109
filters, CrunchDataset-method (filter-catalog), 68
filters<-(filter-catalog), 68
filters<-, CrunchDataset-method (filter-catalog), 68
flattenOrder, 69
flipArrays, 69
folder, 70
folder(), 104
folder<-(folder), 70
forkDataset, 71
forkDataset(), 11, 88, 100
fortify, 72
func (Insertions-class), 77
func, Heading-method (Insertions-class), 77
func, Insertion-method (Insertions-class), 77
func, Subtotal-method (Insertions-class), 77
func, SummaryStat-method (Insertions-class), 77
func<-(Insertions-class), 77
funcs (Insertions-class), 77
funcs, Insertions-method (Insertions-class), 77
geo (CrunchGeography-class), 40
geo(), 6
geo, CrunchVariable-method
(CrunchGeography-class), 40
geo <- (CrunchGeography-class), 40
geo <-, CrunchVariable, CrunchGeography-method
(CrunchGeography-class), 40
geo <-, CrunchVariable, NULL-method
(CrunchGeography-class), 40
Geodata (CrunchGeography-class), 40
Geodata-class (CrunchGeography-class), 40
getTeams, 72, 98
getTeams(), 98
grouped, 73, 137

Heading (Subtotal-class), 142
Heading(), 80, 158
Heading-class (Subtotal-class), 142
headings, 80
hiddenVariables, 73
hiddenVariables <- (hideVariables), 74
hide, 53, 55, 74, 75
hide(), 54
hide, CrunchVariable-method (hide), 74
hide, VariableCatalog-method (hide), 74
hideMissing (cube-missingness), 43
hideMissing, CrunchCube-method
(cube-missingness), 43
hideVariables, 74, 74
http-methods, 75

icon (project-icon), 118
icon <- (project-icon), 118
id (name, AbstractCategory-method), 105
id, AbstractCategory-method
(name, AbstractCategory-method), 105
id, CrunchDataset-method (describe), 57
id, CrunchVariable-method (describe), 57
id, AbstractCategory (Categories-class), 21
ids (Categories-class), 21
ids, AbstractCategories-method
(Categories-class), 21
ids, VariableCatalog-method
(names, BatchCatalog-method), 106
ids, AbstractCategories
(Categories-class), 21
ids <- (Categories-class), 21
ids <-, Categories-method
(Categories-class), 21
importMultitable, 76
index (shoji-index), 131
index, ShojiCatalog-method
(shoji-index), 131
index.table, 76
index <- (shoji-index), 131
index <-, ShojiCatalog-method
(shoji-index), 131
Insertion (Insertions-class), 77
Insertion-class (Insertions-class), 77
Insertions, 158
Insertions (Insertions-class), 77
Insertions-class, 77
interactVariables, 80
is-na-categories, 81
is-public, 82
is-selected-categories, 83
is.archived
(is.archived, DatasetCatalog-method), 83
is.archived, CrunchDataset-method
(is.archived, DatasetCatalog-method), 83
is.archived, DatasetCatalog-method, 83
is.archived <-
(is.archived, DatasetCatalog-method), 83
is.archived <-, CrunchDataset, logical-method
(is.archived, DatasetCatalog-method), 83
is.archived <-, DatasetCatalog, logical-method
(is.archived, DatasetCatalog-method), 83
is.Array (is.dataset), 85
is.CA (is.dataset), 85
is.Categorical (is.dataset), 85
is.CategoricalArray (is.dataset), 85
is.CrunchExpr (is.dataset), 85
is.dataset, 85
is.Datetime (is.dataset), 85
is.derived (derivations), 55
is.derived, CrunchVariable-method
(derivations), 55
is-derived<- (derivations), 55
is-derived<-, CrunchVariable, logical-method (derivations), 55
is.dichotomized (dichotomize), 59
is.dichotomized, Categories-method (dichotomize), 59
is.draft
(is.archived, DatasetCatalog-method), 83
is.draft, CrunchDataset-method
(is.archived, DatasetCatalog-method), 83
is.draft, DatasetCatalog-method
(is.archived, DatasetCatalog-method), 83
is.draft<-
(is.archived, DatasetCatalog-method), 83
is.draft<-, CrunchDataset, logical-method
(is.archived, DatasetCatalog-method), 83
is.draft<-, DatasetCatalog, logical-method
(is.archived, DatasetCatalog-method), 83
is.editor, 86
is.editor, MemberCatalog-method
(is.editor), 86
is.editor, PermissionCatalog-method
(is.editor), 86
is.editor, PermissionTuple-method
(is.editor), 86
is.editor<-(is.editor), 86
is.editor<-, MemberCatalog, logical-method
(is.editor), 86
is.Expr (is.dataset), 85
is.Geodata (is.dataset), 85
is.Heading (Subtotal-class), 142
is.MR (is.dataset), 85
is.Multiple (is.dataset), 85
is.MultipleResponse (is.dataset), 85
is.na, Categories-method
(is-na-categories), 81
is.na, Category-method
(is-na-categories), 81
is.na, CrunchVariable-method
(expressions), 66
is.na, CubeDims-method (cube-methods), 41
is.na<-, Categories, character-method
(is-na-categories), 81
is.na<-, Categories, logical-method
(is-na-categories), 81
is.na<-, Category, logical-method
(is-na-categories), 81
is.na<-, CrunchVariable, ANY-method
(variable-update), 164
is.numeric (is.dataset), 85
is.public (is-public), 82
is.public, CrunchFilter-method
(is-public), 82
is.public, Multitable-method
(is-public), 82
is.public<-(is-public), 82
is.public<-, CrunchFilter-method
(is-public), 82
is.public<-, Multitable-method
(is-public), 82
is.public<-, MultatableCatalog-method
(is-public), 82
is.published
(is.archived, DatasetCatalog-method), 83
is.published, CrunchDataset-method
(is.archived, DatasetCatalog-method), 83
is.published, DatasetCatalog-method
(is.archived, DatasetCatalog-method), 83
is.published<-
(is.archived, DatasetCatalog-method), 83
is.published<-, CrunchDataset, logical-method
(is.archived, DatasetCatalog-method), 83
is.published<-, DatasetCatalog, logical-method
(is.archived, DatasetCatalog-method), 83
is.selected
(name, AbstractCategory-method), 105
is.selected, Categories-method
(is-selected-categories), 83
is.selected, Category-method
(name, AbstractCategory-method), 105
is.selected<-(is-selected-categories), 83
is.selected<-,Categories-method
(is-selected-categories), 83
is.selected<-,Category-method
(is-selected-categories), 83
is.shoji(is.dataset), 85
is.Subtotal(Subtotal-class), 142
is.SummaryStat(SummaryStat-class), 149
is.Text(is.dataset), 85
is.variable(is.dataset), 85
is.weight(weight), 170
is.weight<-(weight), 170
is.weight<-,NumericVariable-method
(weight), 170
is.weightVariable
(modifyWeightVariables), 101
is.weightVariable<-
(modifyWeightVariables), 101
is.weightVariable<-,NumericVariable-method
(modifyWeightVariables), 101
joinDatasets, 87
jsonlite::toJSON(), 154
jsonprep(tojson-crunch), 154
jsonprep,AbstractCategories-method
(tojson-crunch), 154
jsonprep,ANY-method(tojson-crunch), 154
jsonprep,OrderGroup-method
(tojson-crunch), 154
jsonprep,ShojiOrder-method
(tojson-crunch), 154
length, MultitableResult-method
(tabbook-methods), 151
length, OrderGroup-method
(ShojiOrder-length), 135
length, ShojiCatalog-method
(catalog-length), 20
length, ShojiOrder-method
(ShojiOrder-length), 135
length, TabBookResult-method
(tabbook-methods), 151
listDatasets, 88
listDatasets(), 53
loadDataset, 89
loadDataset(), 89
locateEntity, 90
lock, 90
login, 91
logout, 91
makeArray, 92
makeArrayGadget, 92
makeCaseVariable, 35, 93
makeMR(makeArray), 92
makeMR(), 94
makeMRFromText, 94
makeWeight, 95
makeWeight(), 170, 171
margin.table
(prop.table, CrunchCube-method), 119
margin.table, CrunchCube-method
(prop.table, CrunchCube-method), 119
matchCatToFeat, 96
matchCatToFeat(), 7
max (crunch-uni), 36
max, CrunchVariable-method (crunch-uni), 36
max, DatetimeVariable-method
(crunch-uni), 36
max, NumericVariable-method
(crunch-uni), 36
me, 97
mean (crunch-uni), 36
mean, CrunchVariable-method
(crunch-uni), 36
mean, NumericVariable-method
(crunch-uni), 36
measures (cube-methods), 41
measures, CrunchCube-method
(cube-methods), 41
median (crunch-uni), 36
members
(members<-, CrunchTeam, MemberCatalog-method), 97
members, CrunchProject-method
(members<-, CrunchTeam, MemberCatalog-method), 97
members, CrunchTeam-method
(members<-, CrunchTeam, MemberCatalog-method), 97
members, ProjectFolder-method
(members<-, CrunchTeam, MemberCatalog-method), 97
members<-., CrunchTeam, MemberCatalog-method, 97
   members<- (members<-., CrunchTeam, MemberCatalog-method), 97
members<-., CrunchProject, character-method
   (members<-., CrunchTeam, MemberCatalog-method), 97
members<-., CrunchProject, MemberCatalog-method
   (members<-., CrunchTeam, MemberCatalog-method), 97
members<-., CrunchTeam, character-method
   (members<-., CrunchTeam, MemberCatalog-method), 97
members<-., ProjectFolder, character-method
   (members<-., CrunchTeam, MemberCatalog-method), 97
members<-., ProjectFolder, MemberCatalog-method
   (members<-., CrunchTeam, MemberCatalog-method), 97
merge, 98
merge.CrunchDataset (joinDatasets), 87
mergeFork, 99
methods::show, 138
min (crunch-unil, 36
   min, CrunchVariable-method (crunch-unil), 36
min, DatetimeVariable-method (crunch-unil), 36
min, NumericVariable-method (crunch-unil), 36
mkdir (mv), 103
modifyWeightVariables, 101
modifyWeightVariables(), 171
moveToGroup, 102
moveToGroup<- (moveToGroup), 102
MultipleResponseVariable
   (CrunchVariable-class), 41
MultipleResponseVariable-class
   (CrunchVariable-class), 41
multitable-catalog, 103
multitables (multitable-catalog), 103
multitables, CrunchDataset-method
   (multitable-catalog), 103
multitables<- (multitable-catalog), 103
multitables<-., CrunchDataset-method
   (multitable-catalog), 103
mv, 103
mv(), 24, 70, 123, 128, 129
na.omit-categories, 105
na.omit, Categories-method
   (na.omit-categories), 105
name, AbstractCategory-method, 105
name, CrunchDataset-method (describe), 57
name, CrunchVariable-method (describe), 57
name, OrderGroup-method
   (ShojiOrder-slots), 135
name-, ShojoFolder-method (describe), 57
name<- (describe), 57
names, AbstractCategory-method
   (name, AbstractCategory-method), 105
names, CrunchDataset-method (describe), 57
names, CrunchVariable-method
   (describe), 57
names, Multitable-method (describe), 57
name, NULL-method
   (name, AbstractCategory-method), 105
name, OrderGroup-method
   (ShojiOrder-slots), 135
name, ShojoFolder-method (describe), 57
names, AbstractCategories-method
   (Categories-class), 21
names, BatchCatalog-method, 106
names, CategoricalArrayVariable-method
   (names, BatchCatalog-method), 106
names, CrunchCube-method (cube-methods), 41
names, CrunchDataset-method
   (names, BatchCatalog-method), 106
names, MultitableResult-method
   (tabbook-methods), 151
names, OrderGroup-method
   (ShojiOrder-slots), 135
names, ShojoCatalog-method
   (names, BatchCatalog-method), 106
names, ShojoOrder-method
   (ShojiOrder-slots), 135
names, TabBookResult-method
  (tabbook-methods), 151
names, VersionCatalog-method
  (names, BatchCatalog-method), 106
names, CrunchDataFrame
  (CrunchDataFrame), 38
names<-, AbstractCategories-method
  (Categories-class), 21
names<-, MultitableCatalog-method
  (names, BatchCatalog-method), 106
names<-, ShojiCatalog-method
  (names, BatchCatalog-method), 106
names<-, ShojiOrder-method
  (ShojiOrder-slots), 135
ncol, CrunchDataset-method
  (dim-dataset), 60
newDataset, 108
newDatasetByColumn(), 109
newDatasetFromFile(), 109
newFilter, 109
newMultitable, 110
newProject, 111
notes (describe), 57
notes, CrunchCube-method (cube-methods), 41
notes, CrunchDataset-method (describe), 57
notes, CrunchVariable-method (describe), 57
notes, VariableCatalog-method
  (names, BatchCatalog-method), 106
notes, VariableTuple-method (describe), 57
notes<- (describe), 57
notes<-, CrunchDataset-method
  (describe), 57
notes<-, CrunchVariable-method
  (describe), 57
notes<-, VariableCatalog-method
  (names, BatchCatalog-method), 106
noTransforms, 112
NumericVariable (CrunchVariable-class), 41

NumericVariable-class
  (CrunchVariable-class), 41
ordering, 37, 113
ordering, CrunchDataset-method
  (ordering), 113
ordering, CrunchProject-method
  (ordering), 113
ordering, DatasetCatalog-method
  (ordering), 113
ordering, VariableCatalog-method
  (ordering), 113
ordering<-, (ordering), 113
ordering<-, CrunchDataset-method
  (ordering), 113
ordering<-, CrunchProject-method
  (ordering), 113
ordering<-, DatasetCatalog-method
  (ordering), 113
ordering<-, VariableCatalog-method
  (ordering), 113
owner (dataset-owner), 46
owner, CrunchDataset-method
  (dataset-owner), 46
owner<-, (dataset-owner), 46
owner<-, CrunchDataset-method
  (dataset-owner), 46
ownerNames (owners), 114
owners, 114
pendingStream, 114
permissions, 115
permissions, CrunchDataset-method
  (permissions), 115
pk, 115
pk, CrunchDataset-method (pk), 115
pk<-, (pk), 115
popMagnitude (population), 116
popMagnitude, CrunchDataset-method
  (population), 116
popMagnitude<-, (population), 116
popMagnitude<-, CrunchDataset-method
  (population), 116
popSize (population), 116
popSize, CrunchDataset-method
  (population), 116
popSize<-, (population), 116

owner (dataset-owner), 46
owner, CrunchDataset-method
  (dataset-owner), 46
owner<-, (dataset-owner), 46
owner<-, CrunchDataset-method
  (dataset-owner), 46
ownerNames (owners), 114
owners, 114
pendingStream, 114
permissions, 115
permissions, CrunchDataset-method
  (permissions), 115
pk, 115
pk, CrunchDataset-method (pk), 115
pk<-, (pk), 115
popMagnitude (population), 116
popMagnitude, CrunchDataset-method
  (population), 116
popMagnitude<-, (population), 116
popMagnitude<-, CrunchDataset-method
  (population), 116
popSize (population), 116
popSize, CrunchDataset-method
  (population), 116
popSize<-, (population), 116

owner (dataset-owner), 46
owner, CrunchDataset-method
  (dataset-owner), 46
owner<-, (dataset-owner), 46
owner<-, CrunchDataset-method
  (dataset-owner), 46
ownerNames (owners), 114
owners, 114
pendingStream, 114
permissions, 115
permissions, CrunchDataset-method
  (permissions), 115
pk, 115
pk, CrunchDataset-method (pk), 115
pk<-, (pk), 115
popMagnitude (population), 116
popMagnitude, CrunchDataset-method
  (population), 116
popMagnitude<-, (population), 116
popMagnitude<-, CrunchDataset-method
  (population), 116
popSize (population), 116
popSize, CrunchDataset-method
  (population), 116
popSize<-, (population), 116
popSize<-CrunchDataset-method (population), 116
preCrunchBoxCheck, 117
prepareDataForCrunch, 118
project-icon, 118
projects, 119
projects(), 119
prop.table (prop.table, CrunchCube-method), 119
prop.table, CrunchCube-method, 119
prop.table, MultitableResult-method (prop.table, CrunchCube-method), 119
publish (is.archived, DatasetCatalog-method), 83
refresh, 121
refresh(), 159
refresh, CrunchDataset-method (refresh), 121
refresh, CrunchVariable-method (refresh), 121
refresh, ShojiObject-method (refresh), 121
removeEmptyGroups, 121
resetPassword, 122
restoreVersion, 122, 124, 169
restoreVersion(), 124, 169
rmdir, 123
rmdir(), 24, 104
rollup(expressions), 66
rollupResolution(expressions), 66
rollupResolution<- (expressions), 66
round (prop.table, CrunchCube-method), 119
round, CrunchCube-method (prop.table, CrunchCube-method), 119
rstandard, CrunchCube-method (cube-residuals), 43
saveVersion, 123, 124, 169
saveVersion(), 122
scoreCatToFeat, 125
sd (crunch-uni), 36
sd, CrunchVariable-method (crunch-uni), 36
sd, NumericVariable-method (crunch-uni), 36
searchDatasets, 125
session, 126
setdashboardURL (dashboard), 44
setName, 128
setNames, 128
setNames, ShojiCatalog-method (setNames), 128
setOrder, 129
setPopulation (population), 116
setPopulation, CrunchDataset-method (population), 116
setNames, ShojiObject-method (setNames), 128
setOrder, 129
setPopulation (population), 116
settings, 130
settings(), 95
settings<- (settings), 130
show, 131, 160
show, Shoji-index, 131
ShojiObject (ShojiObject-class), 132
ShojiObject-class, 132
ShojiOrder-extract, 132
ShojiOrder-length, 135
ShojiOrder-slots, 135
show, Categories-method (show, ShojiFolder-method), 137
show, Category-method (show, ShojiFolder-method), 137
show, CrunchCube-method (show, ShojiFolder-method), 137
show, CrunchExpr-method (show, ShojiFolder-method), 137
show, CrunchGeography-method (show, ShojiFolder-method), 137
show, CrunchLogicalExpr-method (show, ShojiFolder-method), 137
show, CrunchProject-method  
(show, ShojiFolder-method), 137
show, CrunchVariable-method  
(show, ShojiFolder-method), 137
show, Insertion-method  
(show, ShojiFolder-method), 137
show, Insertions-method  
(show, ShojiFolder-method), 137
show, MultitableResult-method  
(show, ShojiFolder-method), 137
show, OrderGroup-method  
(show, ShojiFolder-method), 137
show, ShojiFolder-method, 137
show, ShojiObject-method  
(show, ShojiFolder-method), 137
show-crunch (show, ShojiFolder-method), 137
showIfAny (cube-missingness), 43
showIfAny, CrunchCube-method  
(cube-missingness), 43
showMissing (cube-missingness), 43
showMissing, CrunchCube-method  
(cube-missingness), 43
showTransforms  
(showTransforms, CrunchCube-method), 138
showTransforms, CategoricalVariable-method  
(showTransforms, CrunchCube-method), 138
showTransforms, CrunchCube-method, 138
SO_schema, 139
SO_survey, 140
startDate (describe), 57
startDate, CrunchDataset-method  
(describe), 57
startDate<= (describe), 57
startDate<=, CrunchDataset-method  
(describe), 57
stats::chisq.test, 44
stats::formula, 110
stats::formula(), 110
stats::lm(), 47
stats::median(), 37
stats::sd(), 37
stats::xtabs(), 33
subset, CrunchDataset-method  
(dataset-extract), 45
Subtotal (Subtotal-class), 142

Subtotal(), 80, 158
Subtotal-class, 142
subtotalArray  
(subtotalArray, CrunchCube-method), 145
subtotalArray, CrunchCube-method, 145
Subtotals, 145
subtotals, 79
subtotals (Subtotal-class), 142
subtotals, CrunchVariable-method  
(Subtotal-class), 142
subtotals, VariableTuple-method  
(Subtotal-class), 142
subtotals<= (Subtotal-class), 142
subtotals<=, CrunchVariable, ANY-method  
(Subtotal-class), 142
subtotals<=, CrunchVariable, NULL-method  
(Subtotal-class), 142
subtotals<=, Insertion, ANY-method  
(Insertions-class), 77
SubtotalsHeadings (Subtotal-class), 142
Subvariables, 106, 108
Subvariables (Subvariables-class), 146
subvariables, 8
subvariables (Subvariables-class), 146
subvariables, CategoricalArrayVariable-method  
(Subvariables-class), 146
subvariables, VariableTuple-method  
(Subvariables-class), 146
Subvariables-class, 146
subvariables<= (Subvariables-class), 146
subvariables<=, CategoricalArrayVariable, ANY-method  
(Subvariables-class), 146
subvariables<=, CategoricalArrayVariable, Subvariables-method  
(Subvariables-class), 146
subvars-extract, 147
SummaryStat (SummaryStat-class), 149
SummaryStat-class, 149
tabBook, 150
tabBook(), 120
tabbook-methods, 151
table, 152
teams, 72
teams  
(members<=, CrunchTeam, MemberCatalog-method), 97
temp.option (temp.options), 153
temp.options, 153
TextVariable (CrunchVariable-class), 41
TextVariable-class (CrunchVariable-class), 41
timestamps (names, BatchCatalog-method), 106
timestamps, VersionCatalog-method (names, BatchCatalog-method), 106
toJSON (tojson-crunch), 154
tojson-crunch, 154
toVariable, 155, 166
toVariable, AsIs-method (toVariable), 155
toVariable, character-method (toVariable), 155
toVariable, CrunchExpr-method (toVariable), 155
toVariable, Date-method (toVariable), 155
toVariable, factor-method (toVariable), 155
toVariable, haven_labelled-method (toVariable), 155
toVariable, haven_labelled_spss-method (toVariable), 155
toVariable, labelled-method (toVariable), 155
toVariable, labelled_spss-method (toVariable), 155
toVariable, logical-method (toVariable), 155
toVariable, numeric-method (toVariable), 155
toVariable, POSIXt-method (toVariable), 155
toVariable, VariableDefinition-method (toVariable), 155
transformations, 78
Transforms (Transforms-class), 157
transforms (Transforms-class), 157
transforms, CrunchCube-method (Transforms-class), 157
transforms, CrunchVariable-method (Transforms-class), 157
transforms, VariableCatalog-method (Transforms-class), 157
transforms, VariableTuple-method (Transforms-class), 157
Transforms-class, 157
transforms<- (Transforms-class), 157
transforms<- (CrunchCube, list-method (Transforms-class), 157
transforms<- (CrunchCube, NULL-method (Transforms-class), 157
transforms<- (CrunchVariable, NULL-method (Transforms-class), 157
transforms<- (CrunchVariable, Transforms-method (Transforms-class), 157
type, 158
type, CrunchVariable-method (type), 158
type, VariableEntity-method (type), 158
type<- (type), 158
type<- (CrunchVariable-method (type), 158
types (names, BatchCatalog-method), 106
types, CrunchCube-method (cube-methods), 41
types, ShojiFolder-method (names, BatchCatalog-method), 106
types, VariableCatalog-method (names, BatchCatalog-method), 106
unbind, 159
undichotomize (dichotomize), 59
undichotomize, CategoricalArrayVariable-method (dichotomize), 59
undichotomize, CategoricalVariable-method (dichotomize), 59
undichotomize, Categories-method (dichotomize), 59
ungrouped (grouped), 73
unhide (hide), 74
unhide, CrunchVariable-method (hide), 74
unhide, VariableCatalog-method (hide), 74
unhideVariables (hideVariables), 74
uniformBasis (describe), 57
uniformBasis, MultipleResponseVariable-method (describe), 57
uniformBasis<- (describe), 57
uniformBasis<-, MultipleResponseVariable-method (describe), 57
unlock (lock), 90
unshare, 131, 159
users, 160
users, CrunchDataset-method (users), 160
users, CrunchProject-method (users), 160
users, DatasetTuple-method (users), 160
utils::write.csv, 173
utils::write.table(), 65

value (name, AbstractCategory-method), 105
value, Category-method
 (name, AbstractCategory-method), 105
value<- (name, AbstractCategory-method), 105
value<-, Category-method
 (name, AbstractCategory-method), 105
values (Categories-class), 21
values, Categories-method
 (Categories-class), 21
values<- (Categories-class), 21
values<-, Categories-method
 (Categories-class), 21
var-categories, 160
VarDef (VariableDefinition), 166
variable-as-methods, 162
variable-extract
 ([,CrunchExpr, CrunchLogicalExpr, ANY-method), 173
variable-to-R
 (as.vector, CrunchExpr-method), 13
variable-update, 164
VariableCatalog, 113, 167
VariableCatalog
 (VariableCatalog-class), 165
VariableCatalog-class, 165
VariableDefinition, 10, 27, 35, 80, 94, 166
VariableDefinition()
 (95, 156
VariableGroup (VariableOrder-class), 167
VariableGroup-class
 (VariableOrder-class), 167
variableMetadata, 167
VariableOrder, 31, 73, 113, 137, 165
VariableOrder (VariableOrder-class), 167
VariableOrder-class, 167
variables, 168
variables, CrunchCube-method
 (variables), 168
variables, CrunchDataset-method
 (variables), 168
variables, CubeDims-method (variables), 168
variables, SearchResults-method
 (variables), 168
variables, VariableFolder-method
 (variables), 168
variables<- (variables), 168
variables<-, CrunchDataset, VariableCatalog-method
 (variables), 168
versions, 123, 124, 169
versions(), 122, 124

webApp, 169
weight, 150, 170
weight(), 34, 37, 95, 171
weight<- (weight), 170
weightVariables, 171
weightVariables(), 170
weightVariables, CrunchDataset-method
 (weightVariables), 171
weightVariables, VariableCatalog-method
 (weightVariables), 171
weightVariables<-
 (modifyWeightVariables), 101
weightVariables<-, CrunchDataset-method
 (modifyWeightVariables), 101
which, 171
which, CrunchLogicalExpr-method (which), 171
which, context-manager
 (with-context-manager), 172
with-context-manager
 (with-context-manager), 172
with_consent (consent), 30
with_consent(), 52, 54, 55, 68
write.csv (exportDataset), 65
write.csv, CrunchDataset-method
 (exportDataset), 65
write.csv.gz, 172