

## Crunch and RCrunch basics

Crunch is a cloud-based data store and analytic engine. It has a web client for doing the most common analytic tasks, allowing analysts and clients to see their data and interact with it. Most of the time, clients will use the web client to explore data, quickly making tables and graphs, filters and subsets. The *rrunch* package for R allows analysts to interact with and manipulate Crunch datasets from within R.

```
library(rrunch)
```

### The Crunch API

Both *rrunch* and the Crunch web application talk to the same Crunch Application Programming Interface (API). Within an R script or interactive session, the *rrunch* package is designed to make interacting with your data in Crunch into idiomatic R.

The Crunch API is served over secure http, and uses [Shoji](#), a way to structure APIs with JSON. When you call *rrunch* functions, they'll take care of using [httr](#) (which wraps [RCurl](#)), and API responses are turned into R objects. Some functions act more like commands, such as `login()`, but its side effects are documented in the regular package documentation.

### Accounts

*Important: authentication to crunch.io with a username and password will change*

The first step after loading the *rrunch* package is to log in:

```
login(email = "xkcd@crunch.io",  
      password = "correct horse battery staple")
```

```
## Logged into crunch.io as xkcd@crunch.io
```

This will log you in to the crunch API. All of these parameters can be also be set as R `options` so you can simply `login()`.

### Datasets and Variables

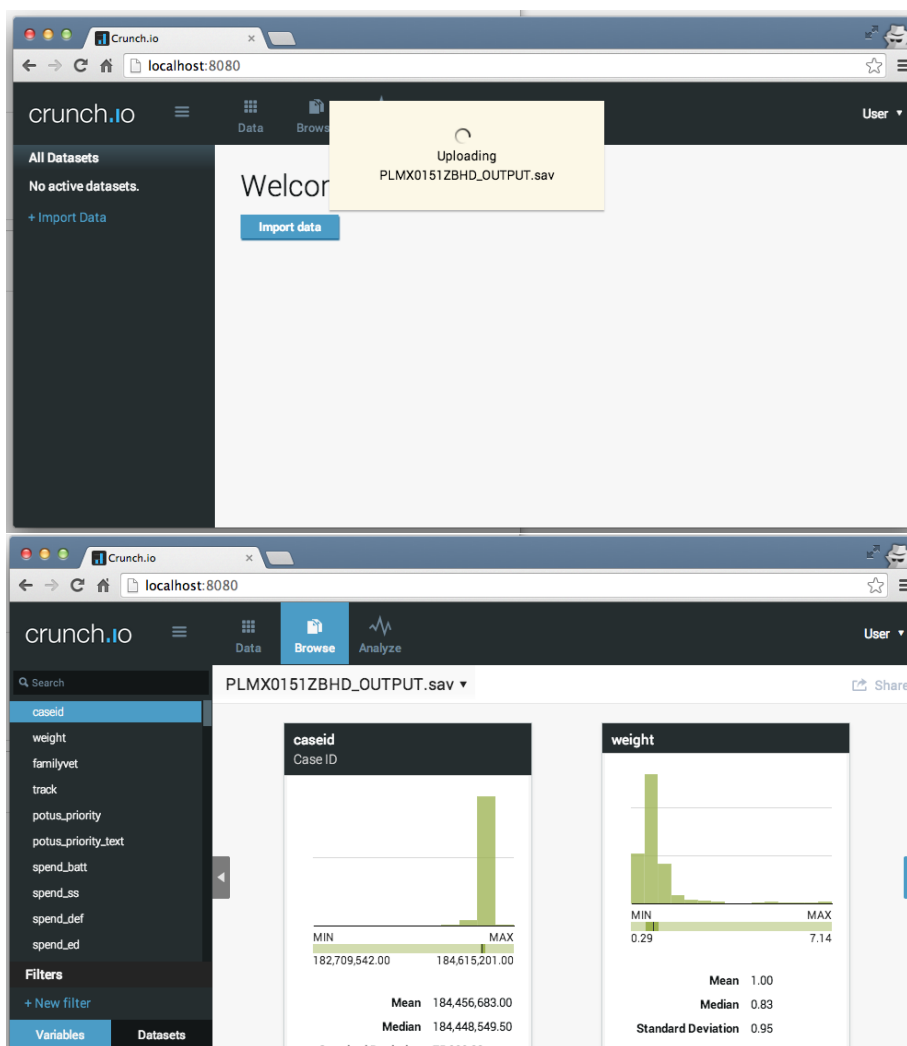
The Crunch data store is built around datasets, which contain variables. Unlike R `data.frames` and atomic vectors, Crunch datasets and variables have a lot of metadata. This vignette is going to use *rrunch* to manipulate a dataset,

alongside an instance of the same dataset in the web client. Many of the operations can be done with either client, but might be faster or easier to automate with R.

## Adding datasets to Crunch

### Via the web client

In the web application, click *Import Data* and upload either a .csv or an SPSS .sav file.



Since the dataset is loaded as soon as it's uploaded, we can go ahead and change its name on the web client before connecting with `rcrunch`.

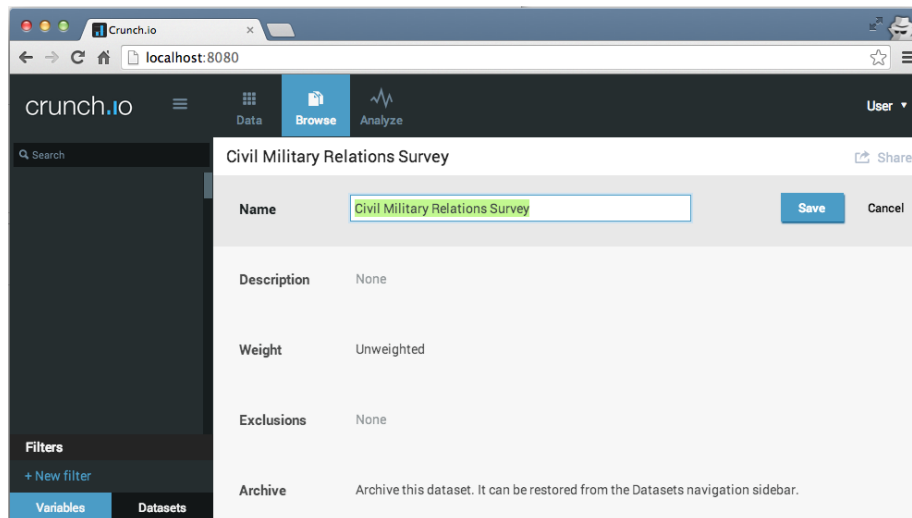


Figure 1: Editing the dataset name

## Adding datasets with `rcrunch`

Of course, you can also create datasets with `rcrunch`. There are two methods:

### 1. `newDatasetFromFile`

`newDatasetFromFile` essentially does what you would do in the web application: uploads your file and creates a dataset from it.

```
ds <- newDatasetFromFile("PLMX0151ZBHD_OUTPUT.sav",  
  name = "Civil Military Relations Survey")
```

### 2. `newDataset`

Alternatively, you can create a dataset from any `data.frame` you have in your R session:

```
library(foreign)  
df <- read.spss("PLMX0151ZBHD_OUTPUT.sav")  
ds <- newDataset(df, name="Civil Military Relations Survey")
```

## Loading Crunch datasets into rcrunch

Datasets already existing on the Crunch server can be loaded by name or by position within the dataset list using `loadDataset`:

```
listDatasets()

## [1] "Civil Military Relations Survey"

(ds <- loadDataset("Civil Military Relations Survey"))

##
## Dataset 'Civil Military Relations Survey'
##
##
## Contains 1000 rows of 101 variables:
##
## $caseid: caseid (numeric)
## $weight: weight (numeric)
## $familyvet: familyvet (numeric)
## $track: track (categorical)
## $potus_priority: potus_priority (categorical)
## $potus_priority_text: potus_priority_text (text)
## $spend_batt: spend_batt (numeric)
## $spend_ss: spend_ss (categorical)
## $spend_def: spend_def (categorical)
## $spend_ed: spend_ed (categorical)
## $spend_fa: spend_fa (categorical)
## $spend_space: spend_space (categorical)
## $spend_hc: spend_hc (categorical)
## $spend_wel: spend_wel (categorical)
## $mil_spending: mil_spending (numeric)
## $ms_pay: ms_pay (categorical)
## $ms_training: ms_training (categorical)
## $ms_milcon: ms_milcon (categorical)
## $ms_weapons: ms_weapons (categorical)
## $ms_war: ms_war (categorical)
## $ms_fa: ms_fa (categorical)
## $fed_tax: fed_tax (categorical)
## $occ_pay: occ_pay (numeric)
## $pay_hst: pay_hst (categorical)
## $pay_up: pay_up (categorical)
## $pay_pd: pay_pd (categorical)
```

```

## $pay_fd: pay_fd (categorical)
## $pay_dr: pay_dr (categorical)
## $pay_rn: pay_rn (categorical)
## $pay_ce: pay_ce (categorical)
## $pay_mo: pay_mo (categorical)
## $pay_em: pay_em (categorical)
## $pay_cp: pay_cp (categorical)
## $pay_pe: pay_pe (categorical)
## $pay_eo: pay_eo (categorical)
## $gayserve: gayserve (categorical)
## $gaymar: gaymar (categorical)
## $pros_grid5: pros_grid5 (numeric)
## $pros_terr5: pros_terr5 (categorical)
## $pros_rec5: pros_rec5 (categorical)
## $pros_war5: pros_war5 (categorical)
## $CAR0000: CAR0000 (multiple_response)
## $career_advice_txt: career_advice_txt (text)
## $pew_draft: pew_draft (categorical)
## $pew_afgh: pew_afgh (categorical)
## $retmil_endor: retmil_endor (categorical)
## $fam_mil: fam_mil (categorical)
## $mil_partis: mil_partis (categorical)
## $believe: believe (categorical)
## $pew_merit: pew_merit (categorical)
## $statements1: statements1 (numeric)
## $state_old: state_old (categorical)
## $state_diff: state_diff (categorical)
## $state_ethic: state_ethic (categorical)
## $state_opp: state_opp (categorical)
## $statements2: statements2 (categorical)
## $statements3: statements3 (categorical)
## $veteran: veteran (numeric)
## $vet_pref: vet_pref (categorical)
## $vet_work: vet_work (categorical)
## $vet_stress: vet_stress (categorical)
## $vet_resp: vet_resp (categorical)
## $mil_comp2: mil_comp2 (categorical)
## $mil_comp3: mil_comp3 (categorical)
## $mil_comp1: mil_comp1 (categorical)
## $milsvc: milsvc (categorical)
## $MIL0000: MIL0000 (multiple_response)
## $vetsvc1: vetsvc1 (numeric)
## $vetsvc2: vetsvc2 (numeric)
## $count_usa: count_usa (numeric)
## $count_usmc: count_usmc (numeric)
## $count_ad: count_ad (numeric)

```

```
## $milsvcfam2: milsvcfam2 (numeric)
## $mil2_sp: mil2_sp (categorical)
## $mil2_pa: mil2_pa (categorical)
## $mil2_ma: mil2_ma (categorical)
## $mil2_bro: mil2_bro (categorical)
## $mil2_sis: mil2_sis (categorical)
## $mil2_son: mil2_son (categorical)
## $mil2_dau: mil2_dau (categorical)
## $newsint2: newsint2 (categorical)
## $birthyr: birthyr (numeric)
## $gender: gender (categorical)
## $race: race (categorical)
## $race_other: race_other (text)
## $educ: educ (categorical)
## $marstat: marstat (categorical)
## $child18: child18 (categorical)
## $religpew: religpew (categorical)
## $txt: txt (text)
## $bornagain: bornagain (categorical)
## $pid3: pid3 (categorical)
## $pid3_t: pid3_t (text)
## $pid7: pid7 (categorical)
## $pid7text: pid7text (text)
## $ideo5: ideo5 (categorical)
## $votereg: votereg (categorical)
## $sourcenews: sourcenews (categorical)
## $othsource: othsource (text)
## $income_v2: income_v2 (categorical)
## $region: region (categorical)
```

## Manipulating datasets

From here, we can set a number of dataset properties, such as toggling a weight variable, changing the name or adding a description, from within R and our changes are carried out on the Crunch dataset.

Note: sometimes changes from `rcrunch` are applied to the remote dataset but not immediately reflected in the `rcrunch` environment. As the package develops, these ‘sync hiccups’ should be smoothed out. For example, because `rcrunch` attaches to datasets by their name, changing their name requires a call to `updateDatasetList()` before `loadDataset` on the new name.

## Hide Variables

Datasets often contain variables that you may want to use – perhaps through a derived variable, a transformation (or recode) – or simply not be relevant for the analytic tasks at hand. In short, you want to hide them. Of course they are never deleted, but they no longer clutter the dataset “workspace”

As with a typical R `data.frame`, you typically assign the return of a dataset-level function back to the variable representing the dataset in your R script or session. As with the `is.na` function, you can update a variable by assigning it to the hidden variables list.

```
ds <- hideVariables(ds, "caseid")
hiddenVariables(ds)

## [1] "caseid"

hiddenVariables(ds) <- "spend_Batt"
hiddenVariables(ds)

## [1] "caseid"
```

These variables are now hidden in the application. And, just as you could restore them there, you can also restore them from R:

```
ds <- unhideVariables(ds, "caseid")
hiddenVariables(ds)

## NULL
```

## Create Array Variables

Sometimes variables’ source datasets don’t contain all of the metadata needed to create the more complex representations available in Crunch datasets. Plain-text .csv files, for example, can’t express that some columns are actually indicator matrices of multiple selections (Multiple Response Variables). Most SPSS .sav files do not indicate Categorical Arrays (otherwise known as “grids”) as being part of a group—they are simply several categorical variables—and the format does not support non-categorical numeric-valued arrays that are sometimes used in surveys for ratings or ‘thermometer’ type responses.

You can use `rcrunch` to “bind” Multiple Response and Categorical Arrays into a Crunch array variable.

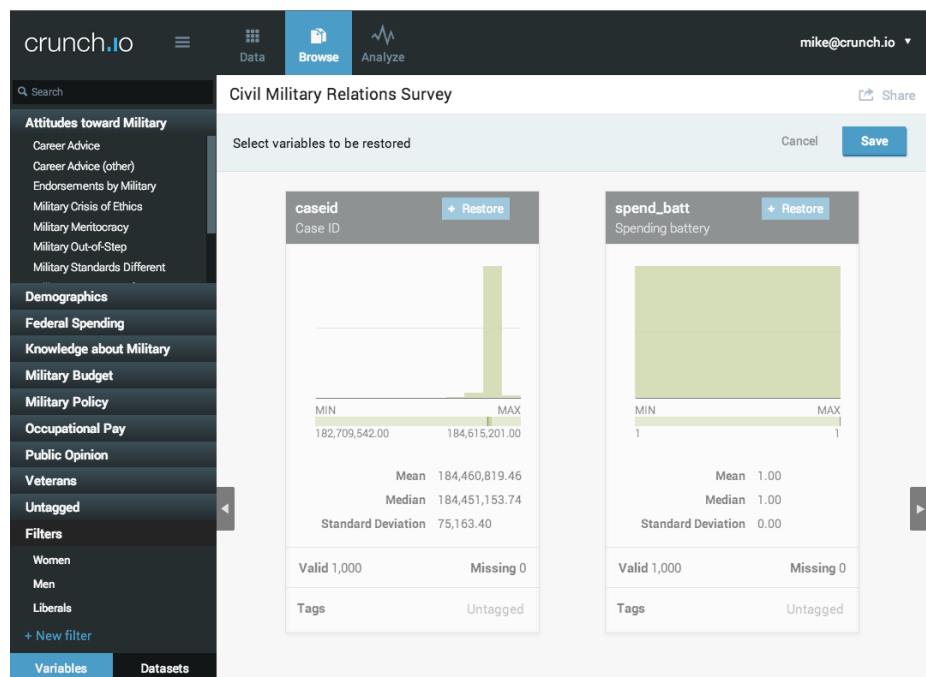


Figure 2: hidden variables



One of the reasons to use R with Crunch is to borrow the power of scripting for repetitive tasks. Hiding variables by regular expression is an obvious use case for datasets with a number of variables that are important to the data but not used for most analytic tasks. Most R functions that work on Crunch datasets have an optional `pattern` argument that lets you use regular expressions for these “bulk” operations. Having hidden the constant `spend_Batt`, we can now group the other `spend_` variables into a Categorical Array.

```
spending <- makeArray(pattern = "^spend_", dataset = ds,
  name = "Federal Spending")

##
## Federal Spending (categorical_array)
##
##
##
##
##
##          Length          Class          Mode
##          1 CategoricalArrayVariable      S4
```

In the Crunch web application, the Federal Spending array has gone from seven separate categorical variable cards to just one, where the subvariables are shown as rows, and the common categories across all of them are shown as columns:

Federal Spending					
	Increase a lot	Increase slightly	Keep the same	Decrease slightly	Decrease a lot
Social Security	216	291	395	39	54
National Defense	104	153	384	209	150
Education Spending	267	269	286	65	110
Foreign Aid	23	60	230	249	434
Space Travel	59	148	328	210	251
Healthcare	226	220	302	87	155
Aid to the Poor	193	209	366	106	124
Properties			Categorical Array		
Valid 1,000 (any) 983 (all)			Missing 0 (all)		

Figure 3: Categorical Array variable card

crunch.io

Data

Browse

Analyze

mike@crunch.io

Search

Attitudes toward Military

Career Advice

Career Advice (other)

Endorsements by Military

Military Crisis of Ethics

Military Meritocracy

Military Out-of-Step

Military Standards Different

Demographics

Federal Spending

Knowledge about Military

Military Budget

Military Policy

Occupational Pay

Public Opinion

Veterans

Untagged

Filters

Women

Men

Liberals

+ New filter

Variables

Datasets

Civil Military Relations Survey

Share

Federal Spending

Valid 1,000

Missing 0

Percent

	Increase a lot	Increase slightly	Keep the same	Decrease slightly	Decrease a lot
Social Security	21.59	29.13	39.53	3.90	5.41
National Defense	10.44	15.31	38.41	20.85	14.99
Education Spending	26.74	26.86	28.58	6.45	11.00
Foreign Aid	2.28	6.03	22.97	24.86	43.45
Space Travel	5.88	14.78	32.81	21.04	25.07
Healthcare	22.58	21.98	30.18	8.70	15.48
Aid to the Poor	19.28	20.86	36.56	10.62	12.40

Federal Spending

Valid 1,000

Missing 0

N %

0.00

WEIGHTED

Figure 4: Array table in analysis view