Crunch and RCrunch basics

Crunch is a cloud-based data store and analytic engine. R is a programming language (technically the language is S, but the R dialect has eclipsed the commercial S-Plus implementation) and an interactive desktop environment for using the language.

Crunch also 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.

> Note: This document is a copy of the markdown output from the package vignette. A more recent version may be available in the built package. In R, simply `vignette('getting-started', package='rcrunch')`

```
library(rcrunch)
```

The Crunch API

Both *rcrunch* and the Crunch web application talk to the same Crunch Application Programming Interface. Within an R script or interactive session, the rcrunch 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 most rcrunch functions, they'll take care of using httr or RCurl, and in general API results 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 rcrunch package is to log in:

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

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

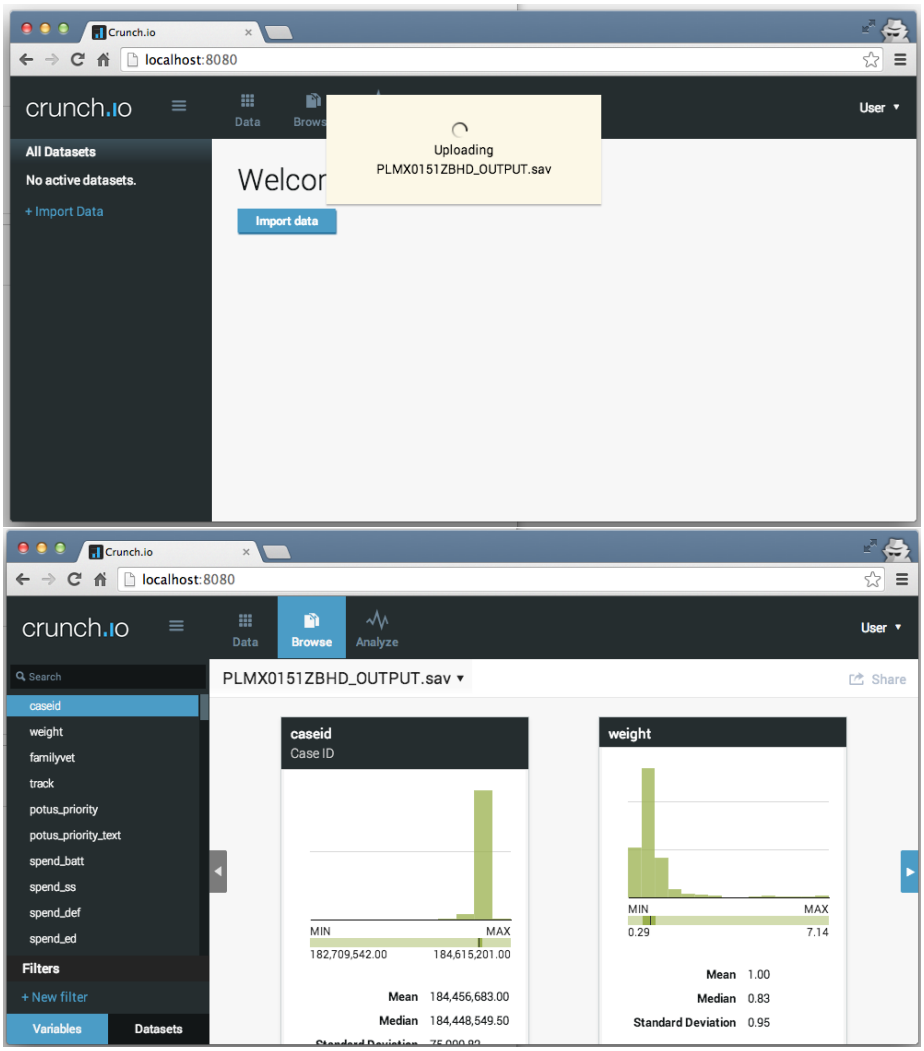
This will log you in to the crunch API. All of these parameters can be also be set as R `options` so you can simply `login()`.

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 rcrunch 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

The easiest way to add data is through the web client. Just 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.

Adding datasets with rcrunch

Of course, you can also upload datasets with rcrunch.

```
ds <- newDatasetFromFile("PLMX0151ZBHD_OUTPUT.sav")
```

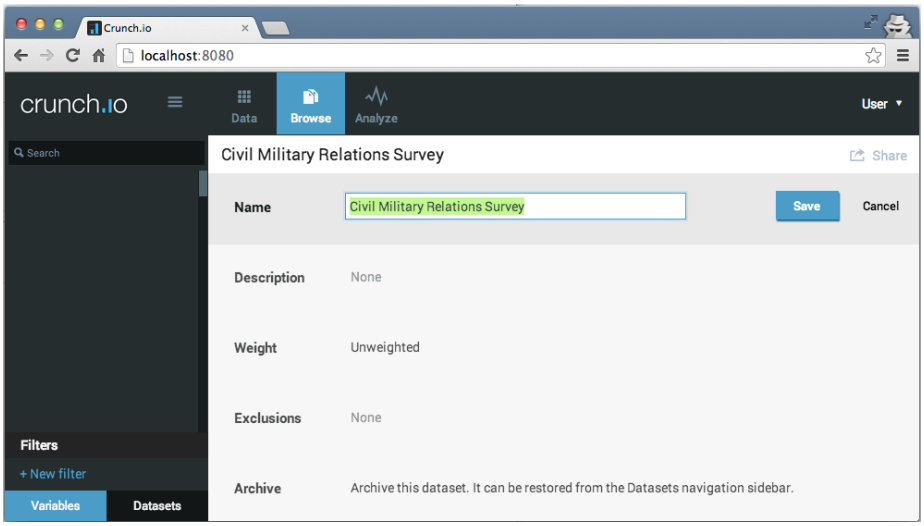


Figure 1: Editing the dataset name

Loading Crunch datasets into rcrunch

```
listDatasets()
```

```
## [1] "Civil Military Relations Survey" "Civil Military Relations Survey"
## [3] "Civil Military Relations Survey"
```

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

```
## Warning: Datasets with duplicate names found. Returning first match.
```

```
##
##  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)
```

```
##   caseid
## "caseid"
```

```
hiddenVariables(ds) <- 'spend_Batt'
hiddenVariables(ds)
```

```
##       caseid   spend_Batt
##     "caseid" "spend_batt"
```

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

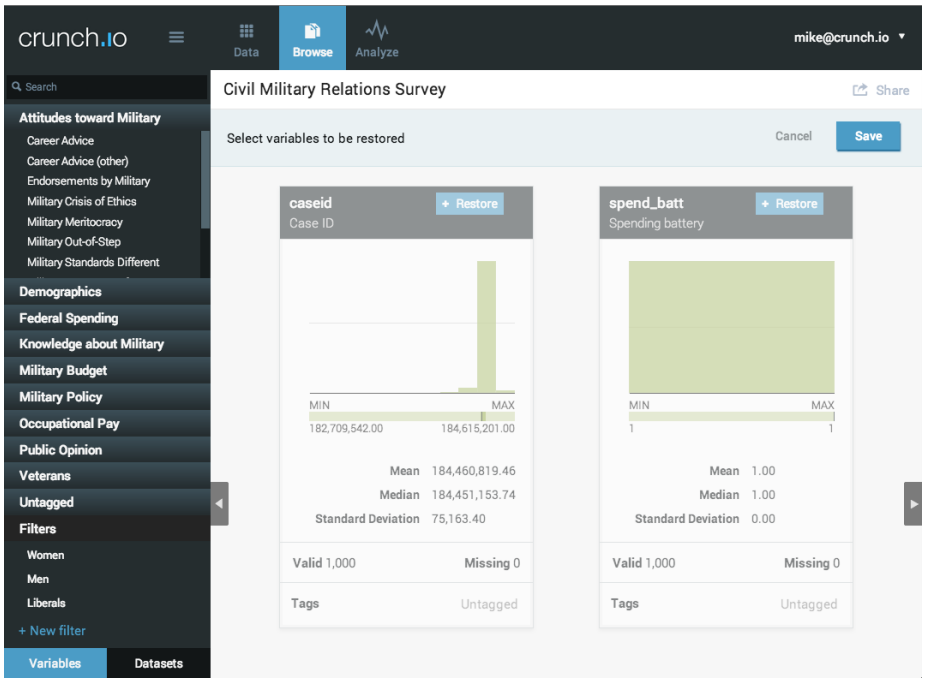


Figure 2: hidden variables

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

```
##   spend_Batt
## "spend_batt"
```

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 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 respones.

You can use rcrunch to "bind" Multiple Response and Categorical Arrays into a Crunch array variable (Numeric Array support is planned for the near future).

One of the resasons 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:

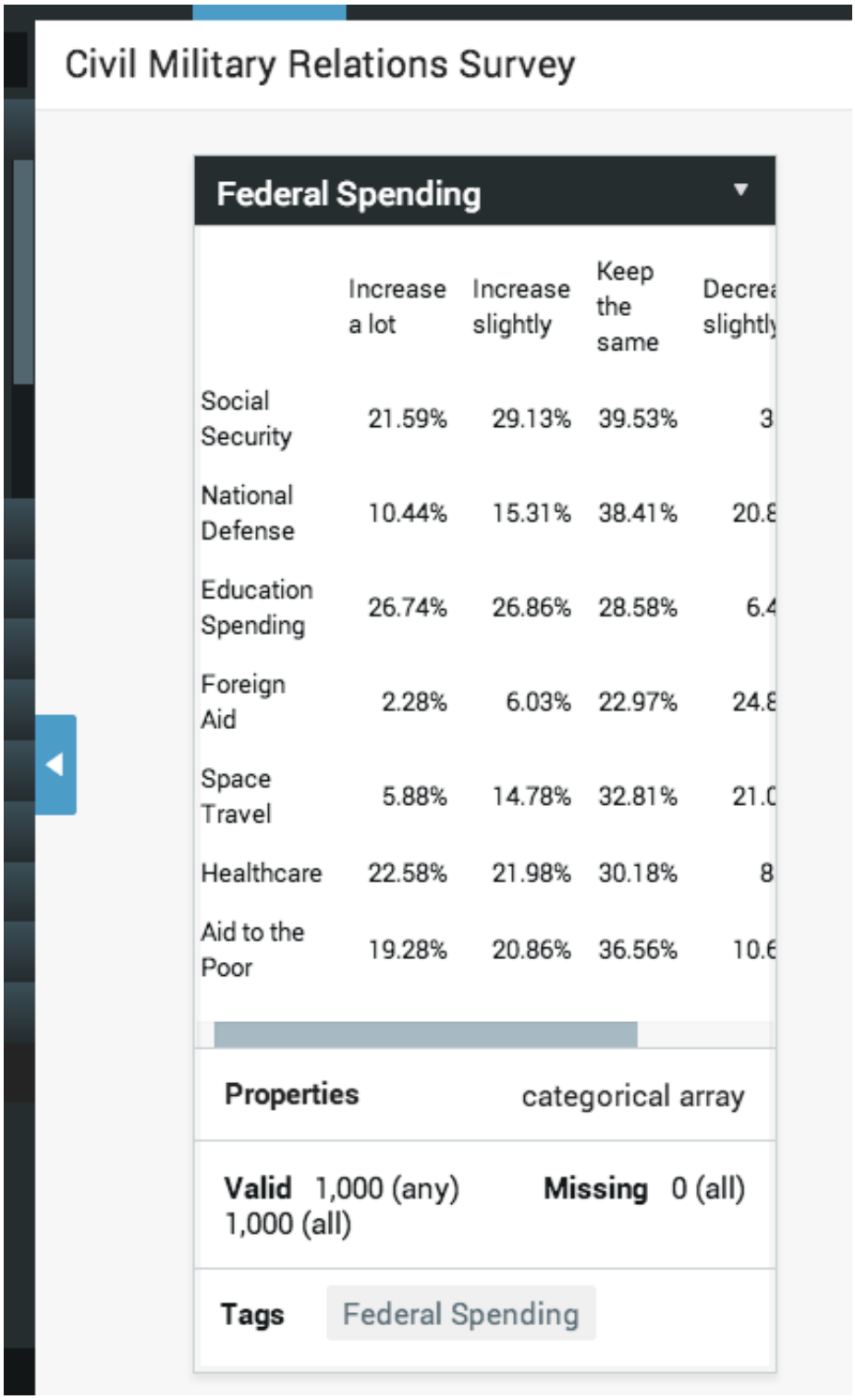


Figure 3: Categorical Array variable card

Figure 4: Array table in analysis view