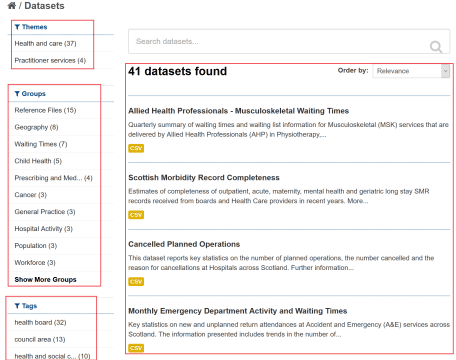
**How resources are grouped in CKAN**

Before we start testing out some code from ckanr, it is important to consider how resources (*I am going to call the individual data items such as specific csv files hosted on CKAN as ‘resources’ but I am not sure if this is necessarily correct*) on CKAN are grouped up as this helps to understand the design of some functions within ckanr. Resources can be grouped within ‘Datasets’, ‘Groups’, ‘Tags’ and ‘Themes’ (and possibly more that I don’t yet know about). Out of these, it is clear to me that ckanr offers functions for exploring resources by all of these groupings except themes (although I could also be mistaken about this). With this out of the way, let’s delve into some code.



**Figure**: How resources are grouped in CKAN.

**Initialising ckanr and exploring groups of resources**

There are, of course, many different data portals that are powered by CKAN so the first thing we need to do with the ckanr package is to tell it which URL to use by default with ckanr\_setup(). Note that if you are working in a place where you need to use a proxy to connect R to the internet, this can also be set within ckanr\_setup() using the proxy argument.

library(tidyverse)

library(ckanr)

ckanr\_setup(url = "<https://www.opendata.nhs.scot>")

Now we can explore the groupings available in the NHS Scotland CKAN website with group\_list(), package\_list() and tag\_list(); from the Figure above, these correspond to ‘Groups’, ‘datasets’ and ‘Tags’ respectively. Note that I only show 10 records in each case to keep things concise.

# View available groups and packages/datasets

group\_list(as = "table")[1:10]

# [1] "acute-hospital-activity" "cancer"

# [3] "child-health" "dental-care"

# [5] "deprivation" "emergency-care"

# [7] "general-practice" "geography"

# [9] "hospital-activity" "infection-disease-and-virus-surveillance"

package\_list(as = "table")[1:10]

# [1] "18-weeks-referral-to-treatment"

# [2] "27-30-month-review-statistics"

# [3] "alcohol-related-hospital-statistics-scotland"

# [4] "allied-health-professionals-musculoskeletal-waiting-times"

# [5] "allied-health-professional-vacancies"

# [6] "annual-cancer-incidence"

# [7] "births-in-scottish-hospitals"

# [8] "cancelled-planned-operations"

# [9] "cancer-mortality"

# [10] "cancer-waiting-times"

tag\_list(as = "table")$name[1:10]

# [1] "31 day" "62 day"

# [3] "address" "adolescent"

# [5] "adult" "age"

# [7] "agenda for change" "agenda for change band"

# [9] "ahp" "ailment"

Being able to see the names related to different groupings is useful if you want to download data from all resources in a particular group. I’ll give an [example of doing this later](https://alan-y.netlify.com/post/trying-ckanr/#downloading-all-resources-from-a-dataset) but first I want to mimic some of the things I done in previous blog posts but using ckanr.

**Connect to CKAN with dplyr and download from one resource**

Let’s demonstrate downloading from one resource using the fairly small [Patients Referred](https://www.opendata.nhs.scot/dataset/allied-health-professionals-musculoskeletal-waiting-times/resource/3988df43-3516-4190-93da-16189db7329a) dataset within [Allied Health Professionals – Musculoskeletal Waiting Times](https://www.opendata.nhs.scot/dataset/allied-health-professionals-musculoskeletal-waiting-times) which has resource ID 3988df43-3516-4190-93da-16189db7329a. We start by using src\_ckan() to create a connection to CKAN (similar to how you would do so for other non-CKAN databases). From there, you can download data in a similar way to when using [dbplyr](https://dbplyr.tidyverse.org/) but using a CKAN resource ID instead of a database table name.

ckan <- src\_ckan("<https://www.opendata.nhs.scot>")

res\_id <- "3988df43-3516-4190-93da-16189db7329a"

dplyr::tbl(src = ckan$con, from = res\_id) %>%

as\_tibble()

# A tibble: 1,115 x 9

# `\_id` HBT2014 ReferralsPerOne~ `\_full\_text` Specialty NumberOfReferra~ NumberOfReferra~

#

# 1 1 S08000~ 2890. '2015q3':9 ~ All AHP ~ 8904 d

# 2 2 S08000~ 411. '1267':5 '2~ Chiropod~ 1267 ""

# 3 3 S08000~ 94.4 '2015q3':3 ~ Occupati~ 291 ""

# 4 4 S08000~ 178. '178.17':5 ~ Orthotics 549 ""

# 5 5 S08000~ 2206. '2015q3':2 ~ Physioth~ 6797 ""

# 6 6 S08000~ 1530. '1472':7 '1~ All AHP ~ 1472 d

# 7 7 S08000~ 165. '159':1 '16~ Orthotics 159 ""

# 8 8 S08000~ 1365. '1313':2 '1~ Physioth~ 1313 ""

# 9 9 S08000~ 2562. '2015q3':7 ~ All AHP ~ 3212 d

# 10 10 S08000~ 197. '197.02':1 ~ Chiropod~ 247 ""

# # ... with 1,105 more rows, and 2 more variables: Quarter ,

# # ReferralsPerOneHundredThousandPopulationQF

The variables look like they’ve been downloaded in a bit of a random order and that \_full\_text variable seems to have appeared so this makes me think that ckanr is using [SQL to download the data](https://alan-y.netlify.com/post/ckan-api/#querying-with-sql). This is easy enough to confirm.

getAnywhere(tbl.src\_ckan)

function (src, from, ..., name = NULL)

{

if (is.null(name)) {

tbl\_sql("ckan", src = src, from = sql(from), ...)

}

else {

tbl\_sql(subclass = "ckan", src = src, from = sql(sprintf("SELECT \* FROM \"%s\"",

name)))

}

}

Now let’s try combining this with some basic dplyr functions like select() and filter().

dplyr::tbl(src = ckan$con, from = res\_id) %>%

select(Quarter, HBT2014) %>%

filter(HBT2014 == "S08000015") %>%

as\_tibble()

# A tibble: 89 x 2

# Quarter HBT2014

#

# 1 2015Q3 S08000015

# 2 2015Q3 S08000015

# 3 2015Q3 S08000015

# 4 2015Q3 S08000015

# 5 2015Q3 S08000015

# 6 2015Q4 S08000015

# 7 2015Q4 S08000015

# 8 2015Q4 S08000015

# 9 2015Q4 S08000015

# 10 2015Q4 S08000015

# # ... with 79 more rows

# Warning messages:

# 1: Translator is missing window variants of the following aggregate functions:

# \* all

# \* any

# \* cor

# \* cov

# \* paste

# \* sd

#

# 2: Translator is missing window variants of the following aggregate functions:

# \* all

# \* any

# \* cor

# \* cov

# \* paste

# \* sd

#

# 3: Translator is missing window variants of the following aggregate functions:

# \* all

# \* any

# \* cor

# \* cov

# \* paste

# \* sd

We get a long list of warnings explaining what you cannot do with the SQL translation available in ckanr but otherwise, works great!

**Downloading all resources from a dataset**

Often, a dataset on CKAN contains many resources related to the same thing. For example, the [Consultant Vacancies](https://www.opendata.nhs.scot/dataset/consultant-vacancies) dataset (remember you can see all available ‘Datasets’ using package\_list()) contains different csv files for vacancies at different time points.

cons\_vac <- package\_show("consultant-vacancies", as = "table")$resources

cons\_vac %>%

select(name, id)

# name id

# 1 Vacancies June 2019 16e27935-325c-471b-89dc-d41c84b3a744

# 2 Vacancies March 2019 ca67b2a4-b2f3-4420-8b77-3771c53b01f4

# 3 Vacancies December 2018 5da80103-4da8-4694-a8b5-2332dfc43e25

# 4 Vacancies September 2018 91d7b780-f2cb-47fb-919f-1c165ed7d301

# 5 Vacancies June 2018 e874f6f4-6cf5-402c-af1d-2d4f26cc669f

# 6 Vacancies March 2018 415c2f86-db7c-4c12-9a64-0cd9cf0d9118

Now if you extract the required resource IDs, you can download all of the datasets with some help from the fantastic [purrr](https://purrr.tidyverse.org/) package.

id\_list <- cons\_vac$id

# Download each resource into a list item

cons\_vac\_list <- map(id\_list, ~as\_tibble(dplyr::tbl(src = ckan$con, from = .x)))

# Check how many variables in each resource

map\_dbl(cons\_vac\_list, length)

# [1] 12 12 12 14 15 12

# Not all resources have the same structure

# Check variable names for a couple that differ

map(cons\_vac\_list[3:4], names)

# [[1]]

# [1] "WorkforceRegionGrouping" "HB2014"

# [3] "HB2014QF" "TotalVacancies"

# [5] "\_full\_text" "Specialty"

# [7] "VacanciesGreater6Months" "Date"

# [9] "SpecialtyQF" "\_id"

# [11] "Establishment" "StaffInPost"

#

# [[2]]

# [1] "WorkforceRegionGrouping" "HB2014"

# [3] "HB2014QF" "TotalVacancies"

# [5] "TotalVacanciesQF" "\_full\_text"

# [7] "Specialty" "EstablishmentQF"

# [9] "VacanciesGreater6Months" "Date"

# [11] "SpecialtyQF" "\_id"

# [13] "Establishment" "StaffInPost"

# TotalVacanciesQF and EstablishmentQF not in resource 3 but are in resource 4

# Combine just the first 3 resources which look like they all have the same structure

bind\_rows(cons\_vac\_list[1:3])

# A tibble: 1,822 x 12

# WorkforceRegion~ HB2014 HB2014QF TotalVacancies `\_full\_text`

#

# 1 National Bodies~ SB0806 "" 0 '0':9 '1.4'~

# 2 Scotland S9200~ d 0 '0':5 '2019~

# 3 Scotland S9200~ d 0 '0':5 '2.1'~

# 4 National Bodies~ SB0807 "" 0 '0':9 '0.3'~

# 5 North S0800~ "" 0 '0':5 '0.1'~

# 6 National Bodies~ SB0808 "" 0 '0':9 '1.2'~

# 7 East S0800~ "" 0 '0':3 '12.1~

# 8 East S0800~ "" 0 '0':3 '1.7'~

# 9 National Bodies~ SB0804 "" 0 '0':1 '1':9~

# 10 National Bodies~ SB0807 "" 0 '0':9 '0.4'~

# ... with 1,812 more rows, and 7 more variables: Specialty ,

# VacanciesGreater6Months , Date , SpecialtyQF ,

# `\_id` , Establishment , StaffInPost

So that is a basic workflow using ckanr alongside functions from purrr for combining related resources into one dataset. I’ve also presented some ways of checking consistency in the structure of those datasets (an essential step when trying to do something like this) and in this case, not all of the datasets were the same so I just combined the most recent 3 datasets for consultant vacancies at the end for simplicity here; in reality you might want to look at ways to make all of the data consistent first and then combine them up but I’ll leave that data wrangling exercise up to the interested reader.