Package 'fhircrackr'

November 24, 2020

Type Package

Title Handling HL7 FHIR Resources in R

Version 0.2.1

Date 2020-11-24

Description Useful tools for conveniently downloading FHIR resources in xml format and converting them to R data frames. The package uses FHIR-search to download bundles from a FHIR server, provides functions to save and read xml-files containing such bundles and allows flattening the bundles to data.frames using XPath expressions.

BugReports https://github.com/POLAR-fhiR/fhircrackr/issues

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports xml2, stringr, httr, utils, dplyr, plyr, data.table

Suggests knitr, rmarkdown

VignetteBuilder knitr

Depends R (>= 3.5.0)

NeedsCompilation no

Author Thomas Peschel [aut, cre],
Julia Gantner [aut] (<https://orcid.org/0000-0003-1568-5893>),
Jens Przybilla [aut],
Frank Meineke [aut] (<https://orcid.org/0000-0002-9256-7543>)

Maintainer Thomas Peschel <tpeschel@imise.uni-leipzig.de>

Repository CRAN

Date/Publication 2020-11-24 14:30:07 UTC

R topics documented:

`fhir_canonical_design` *Retrieve design of last call to fhir_crack*

Description

Returns the complete design of the last call to `fhir_crack` with automatically amended elements, i.e. the canonical form of the design with elements resource, cols, style and respective sub-elements.

Usage

```
fhir_canonical_design()
```

Examples

```
#load example bundles
bundles <- fhir_unserialize(patient_bundles)

#incomplete but valid design
design <- list(
  Pat = list(
    resource = "//Patient"
    )
)

result <- fhir_crack(bundles, design)

fhir_canonical_design()
```

`fhir_capability_statement` *Get capability statement*

Description

Get the capability statement of a FHIR server.

Usage

```
fhir_capability_statement(
  url = "https://hapi.fhir.org/baseR4",
  username = NULL,
  password = NULL,
  sep = " ",
  remove_empty_columns = TRUE,
  brackets = NULL,
  verbose = 2,
  add_indices
)
```

Arguments

`url`	The URL of the FHIR server endpoint.
`username`	A string containing the username for basic authentication. Defaults to NULL, meaning no authentication.
`password`	A string containing the password for basic authentication. Defaults to NULL, meaning no authentication.
`sep`	A string to separate pasted multiple entries
`remove_empty_columns`	Logical scalar. Remove empty columns?
`brackets`	A character vector of length two defining the brackets surrounding indices for multiple entries, e.g. `c( "<",">")`. If `NULL`, no indices will be added to multiple entries. `NULL` means `brackets` is looked up in design, if it is `NULL` there too, no indices are added.
`verbose`	An integer Scalar. If 0, nothings is printed, if 1, only finishing message is printed, if > 1, downloading/extraction progress will be printed. Defaults to 2.
`add_indices`	Deprecated. This argument was used to control adding of indices for multiple entries. This is now done via the brackets argument. If brackets is `NULL`, no indices are added, if brackets is not `NULL`, indices are added to multiple entries.

Value

A list of data frames containing the information from the statement

Examples

```
cap <- fhir_capability_statement("https://hapi.fhir.org/baseR4")
```

`fhir_common_columns` *Find common columns*

Description

This is a convenience function to find all column names in a data frame starting with the same string that can then be used for `fhir_melt`.

Usage

```
fhir_common_columns(data_frame, column_names_prefix)
```

Arguments

`data_frame`	A data frame with automatically named columns as produced by `fhir_crack`.
`column_names_prefix`	A string containing the common prefix of the desired columns.

Details

It is intended for use on data frames with column names that have been automatically produced by `fhir_crack` and follow the form `level1.level2.level3` such as `name.given.value` or `code.coding.system.value`. Note that this function will only work on column names following exactly this schema.

The resulting character vector can be used for melting all columns belonging to the same attribute in an indexed data frame, see `?fhir_melt`.

Value

A character vector with the names of all columns matching `column_names_prefix`.

Examples

```
#unserialize example bundles
bundles <- fhir_unserialize(medication_bundles)

#crack Patient Resources
design <- list(
  Patients = list(".//Patient")
)

dfs <- fhir_crack(bundles, design)

#look at automatically generated names
names(dfs$Patients)
```

```
#extract all column names beginning with the string "name"
fhir_common_columns(data_frame = dfs$Patients, column_names_prefix = "name")
```

`fhir_crack` *Flatten list of FHIR bundles*

Description

Converts all FHIR bundles (the result of `fhir_search`) to a list of data frames.

Usage

```
fhir_crack(
  bundles,
  design,
  sep = NULL,
  remove_empty_columns = NULL,
  brackets = NULL,
  verbose = 2,
  data.table = FALSE,
  add_indices
)
```

Arguments

`bundles`	A FHIR search result as returned by `fhir_search`.
`design`	A named list of data frame descriptions. Each data frame description will produce one data frame in the list of data frames returned by `fhir_crack`, where the data frame has the same name as the data frame description in `design`. Each data frame description is a list of 3 named elements: 1) `design$resource`: Mandatory. A string with an XPath expression locating the entries for this data frame in a FHIR bundle page. This is usually the path to a resource tpye such as `"//Patient"` or `"//Observation"`. 2) `design$cols`: Optional. Either a string containing an XPath expression referencing a certain level of attributes that should be extracted (`".//@value"` e.g. would extract all values on the root level) or a named list where the elements are XPath expressions indicating the specific position of attributes to extract and the names of the list elements are the column names of the resulting data frame. If `design$cols` is `NULL`, all available attributes will be extracted. 3) `design$style`: Optional. This can be used instead of the function arguments `sep`, `brackets` and `remove_empty_columns`, but will be overwritten if the corresponding function arguments are not `NULL`. A named list with the following optional elements: a) `design$style$sep` : A string to separate pasted multiple entries.

b) design$style$brackets: A character vector of length two defining the brackets surrounding indices for multiple entries, e.g. c("<",">"). If NULL, no indices will be added to multiple entries.

c) design$style$rm_empty_cols: Logical scalar. Remove empty columns?

For a more detailed explanation and comprehensive examples of design, please see the package vignette.

sep — A string to separate pasted multiple entries. NULL means sep is looked up in design, if it is NULL there too, sep will be set to " " as the default.

remove_empty_columns — Logical scalar. Remove empty columns? NULL means remove_empty_columns is looked up in design, if it is NULL there too, remove_empty_columns will be set to TRUE as the default.

brackets — A character vector of length two defining the brackets surrounding indices for multiple entries, e.g. c("<",">"). If NULL, no indices will be added to multiple entries. NULL means brackets is looked up in design, if it is NULL there too, no indices are added.

verbose — An Integer Scalar. If 0, nothing is printed, if 1, only finishing message is printed, if > 1, extraction progress will be printed. Defaults to 2.

data.table — Logical scalar. Should tables be returned in data.table format instead of data.frame? defaults to FALSE.

add_indices — Deprecated. This argument was used to control adding of indices for multiple entries. This is now done via the brackets argument. If brackets is NULL, no indices are added, if brackets is not NULL, indices are added to multiple entries.

Value

A list of data frames (if data.table = FALSE) or a list of data.tables if data.table = TRUE.

Examples

```
#unserialize example bundle
bundles <- fhir_unserialize(medication_bundles)

#define attributes to extract
design <- list(

 #define specifically which elements to extract
MedicationStatement = list(

resource = ".//MedicationStatement",

cols = list(
MS.ID              = "id",
STATUS.TEXT        = "text/status",
STATUS             = "status",
MEDICATION.SYSTEM  = "medicationCodeableConcept/coding/system",
MEDICATION.CODE    = "medicationCodeableConcept/coding/code",
MEDICATION.DISPLAY = "medicationCodeableConcept/coding/display",
```

```
DOSAGE              = "dosage/text",
PATIENT             = "subject/reference",
LAST.UPDATE         = "meta/lastUpdated"
),

style = list(
sep = " ",
brackets = c("[", "]"),
rm_empty_cols= FALSE
)
),

 #extract all values
Patients = list(

resource = ".//Patient"
)
)

#convert fhir to data frames
list_of_tables <- fhir_crack(bundles, design)

#check results
head(list_of_tables$MedicationStatement)
head(list_of_tables$Patients)
```

---

fhir_load *Load bundles from xml-files*

Description

Reads all bundles stored as xml files from a directory.

Usage

```
fhir_load(directory)
```

Arguments

directory	A string containing the path to the folder were the files are stored.

Value

A list of bundles in xml format.

Examples

```
#unserialize example bundle
bundles <- fhir_unserialize(medication_bundles)

#save to temporary directory
fhir_save(bundles, directory = tempdir())

#load from temporary directory
loaded_bundles <- fhir_load(tempdir())
```

`fhir_load_design` *Load design from xml*

Description

Loads a design for use with `fhir_crack` from an xml file into R

Usage

```
fhir_load_design(file)
```

Arguments

`file`	A string specifying the file from which to read

Value

A list representing a valid design for `fhir_crack`

Examples

```
#create and save design
design <- list(
   Pat = list(
      resource = "//Patient",
      cols = "./*"
   )

)
temp <- tempfile()

fhir_save_design(design, file = temp)
design <- fhir_load_design(temp)
```

`fhir_melt` *Melt multiple entries*

Description

This function divides multiple entries in an indexed data frame as produced by `fhir_crack` into separate observations.

Usage

```
fhir_melt(
  indexed_data_frame,
  columns,
  brackets = c("<", ">"),
  sep = " ",
  id_name = "resource_identifier",
  all_columns = FALSE
)
```

Arguments

`indexed_data_frame`	A data frame with indexed multiple entries.
`columns`	A character vector specifying the names of all columns that should be molten simultaneously. It is advisable to only melt columns simultaneously that belong to the same (repeating) attribute!
`brackets`	A character vector of length 2, defining the brackets used for the indices.
`sep`	A string defining the separator that was used when pasting together multiple entries in `fhir_crack`.
`id_name`	A string, the name of the column that will hold the identification of the origin of the new rows.
`all_columns`	A logical scalar. Return all columns or only the ones specified in `columns`?

Details

Every row containing values that consist of multiple entries on the variables specified by the argument `columns` will be turned into multiple rows, one for each entry. Values on other variables will be repeated in all the new rows.

The new data frame will contain only the molten variables (if `all_cloumns = FALSE`) or all variables (if `all_columns = TRUE`) as well as an additional variable `resource_identificator` that maps which rows came from the same origin. The name of this column can be changed in the argument `id_name`.

For a more detailed description on how to use this function please see the package vignette.

Value

A data frame where each entry from the variables in `columns` appears in a separate row.

Examples

```
#generate example
bundle <- xml2::read_xml(
"<Bundle>

    <Patient>
        <id value='id1'/>
        <address>
            <use value='home'/>
            <city value='Amsterdam'/>
            <type value='physical'/>
           <country value='Netherlands'/>
        </address>
        <birthDate value='1992-02-06'/>
    </Patient>

    <Patient>
        <id value='id2'/>
        <address>
            <use value='home'/>
            <city value='Rome'/>
            <type value='physical'/>
            <country value='Italy'/>
        </address>
        <address>
            <use value='work'/>
            <city value='Stockholm'/>
            <type value='postal'/>
            <country value='Sweden'/>
        </address>
        <birthDate value='1980-05-23'/>
    </Patient>
</Bundle>"
)

#crack fhir resources
dfs <- fhir_crack(bundles = list(bundle), design = list(Patients = list(resource = ".//Patient")),
                  brackets = c("[","]"))

#find all column names associated with attribute address
col_names <- fhir_common_columns(dfs$Patients, "address")

#original data frame
dfs$Patients

#only keep address columns
fhir_melt(indexed_data_frame = dfs$Patients, columns = col_names,
          brackets = c("[","]", sep = " "))
```

```
#keep all columns
fhir_melt(indexed_data_frame = dfs$Patients, columns = col_names,
          brackets = c("[","]"), sep = " ", all_columns = TRUE)
```

`fhir_next_bundle_url` *Next Bundle's URL*

Description

fhir_next_bundle_url() gives the url of the next available bundle. This is useful when you have not a lot of memory available or when a download of bundles was interrupted for some reason. In case of small memory, you can use `fhir_next_bundle_url` together with the `max_bundle` argument from `fhir_search` to download bundles in smaller batches in a loop. See details in the example.

Usage

```
fhir_next_bundle_url()
```

Value

A string containing an url to the next bundle available on the FHIR server of your last call to `fhir_search` or NULL if no further bundle is available.

Examples

```
# workflow for small memory environments, downloading small batches of bundles
# for really small memory environments consider also using the _count option in
# your FHIR search request.
# You can iteratively download, crack and save the bundles until all bundles are processed or the
# desired number of bundles is reached.
url <- "http://hapi.fhir.org/baseR4/Observation"
count <- 0
while(!is.null(url) && count < 5){
bundles <- fhir_search(url, verbose = 2, max_bundles = 2)
tables <- fhir_crack(bundles, list(Obs=list(resource = "//Observation")))
save(tables, file = paste0(tempdir(),"/table_", count, ".RData"))
count <- count + 1
url <- fhir_next_bundle_url()
}
```

`fhir_rm_indices` *Remove indices from data frame*

Description

Removes the indices produced by `fhir_crack` when `add_indices=TRUE`

Usage

```
fhir_rm_indices(
  indexed_data_frame,
  brackets = c("<", ">"),
  columns = names(indexed_data_frame)
)
```

Arguments

`indexed_data_frame`	A data frame with indices for multiple entries as produced by `fhir_crack`
`brackets`	A string vector of length two defining the brackets that were used in `fhir_crack`
`columns`	A string vector of column names, indicating from which columns indices should be removed. Defaults to all columns.

Value

A data frame without indices.

Examples

```
bundle <- xml2::read_xml(
"<Bundle>

        <Patient>
            <id value='id1'/>
            <address>
                <use value='home'/>
                <city value='Amsterdam'/>
                <type value='physical'/>
               <country value='Netherlands'/>
            </address>
            <birthDate value='1992-02-06'/>
        </Patient>

        <Patient>
            <id value='id2'/>
            <address>
                <use value='home'/>
```

```
                <city value='Rome'/>
                <type value='physical'/>
                <country value='Italy'/>
            </address>
            <address>
                <use value='work'/>
                <city value='Stockholm'/>
                <type value='postal'/>
                <country value='Sweden'/>
            </address>
            <birthDate value='1980-05-23'/>
        </Patient>
</Bundle>"
)


dfs <- fhir_crack(bundles = list(bundle),
                  design = list(Patients = list(resource = "/Bundle/Patient")),
                  brackets = c("[", "]"),verbose = 2)

df_indices_removed <- fhir_rm_indices(dfs[[1]], brackets=c("[", "]"))
```

`fhir_save` *Save FHIR bundles as xml-files*

Description

Writes a list of FHIR bundles as numbered xml files into a directory.

Usage

```
fhir_save(bundles, directory = "result")
```

Arguments

`bundles`	A list of xml objects representing the FHIR bundles.
`directory`	A string containing the path to the folder to store the data in.

Examples

```
#unserialize example bundle
bundles <- fhir_unserialize(medication_bundles)

#save to temporary directory
fhir_save(bundles, directory = tempdir())
```

fhir_save_design *Write design to xml*

Description

Writes a design for use with fhir_crack to an xml file

Usage

```
fhir_save_design(design, file = "design.xml")
```

Arguments

design	A list representing a valid design as used in fhir_crack
file	A string specifying the file to write to, defaults to writing "design.xml" into the current working directory

Examples

```
design <- list(
   Pat = list(
      resource = "//Patient",
      cols = "./*"
   )

)

fhir_save_design(design, file = tempfile())
```

fhir_search *Download Fhir search result*

Description

Downloads all FHIR bundles of a FHIR search request from a FHIR server.

Usage

```
fhir_search(
  request,
  username = NULL,
  password = NULL,
  max_bundles = Inf,
  verbose = 1,
  max_attempts = 5,
  delay_between_attempts = 10,
```

```
  log_errors = 0,
  save_to_disc = FALSE,
  directory = paste0("FHIR_bundles_", gsub("-| |:", "", Sys.time()))
)
```

Arguments

request	A string containing the full FHIR search request.
username	A string containing the username for basic authentication. Defaults to NULL, meaning no authentication.
password	A string containing the password for basic authentication. Defaults to NULL, meaning no authentication.
max_bundles	Maximal number of bundles to get. Defaults to Inf meaning all available bundles are downloaded.
verbose	An Integer Scalar. If 0, nothings is printed, if 1, only finishing message is printed, if > 1, downloading progress will be printed. Defaults to 2.
max_attempts	A numeric scalar. The maximal number of attempts to send a request, defaults to 5.
delay_between_attempts	A numeric scalar specifying the delay in seconds between two attempts. Defaults to 10.
log_errors	Takes values 0, 1 or 2. Controls the logging of errors. 1 and 2 will write a file to the current working directory. 0: no logging of errors, 1: tabulate http response and write to csv-file 2: write http response as to xml-file
save_to_disc	A logical scalar. If TRUE the bundles are saved as numerated xml-files into the directory specified in the argument `directory` and not returned as a bundle list in the R session. This is useful when a lot of bundles are to be downloaded and keeping them all in one R session might overburden working memory. When download is complete, the bundles can be loaded into R using `fhir_load`. Defaults to FALSE, i.e. bundles are returned as a list within the R session.
directory	The directory the bundles are saved to when `save_to_disc` is TRUE. Defaults to creating a time-stamped directory into the current working directory.

Value

A list of bundles in xml format when `save_to_disc = FALSE` (the default), else NULL.

Examples

```
bundles <- fhir_search("https://hapi.fhir.org/baseR4/Medication?", max_bundles=3)
```

fhir_serialize *Serialize a FHIR Bundle list*

Description

Serializes a list of FHIR bundles to allow for saving in .rda or .RData format without losing integrity of pointers

Usage

```
fhir_serialize(bundles)
```

Arguments

bundles	A list of xml objects representing FHIR bundles as returned by fhir_search

Value

A list of serialized xml objects

Examples

```
#example bundles are serialized, unserialize like this:
bundles <- fhir_unserialize(medication_bundles)

#Serialize like this:
bundles_for_saving <- fhir_serialize(bundles)
```

fhir_unserialize *Unserialize a FHIR Bundle list*

Description

Unserializes a list of FHIR bundles that have been serialized to allow for saving in .rda or .RData format.

Usage

```
fhir_unserialize(bundles)
```

Arguments

bundles	A list of serialized xml objects representing FHIR bundles as returned by fhir_search

Value

A list of unserialized xml objects

Examples

```
bundles <- fhir_unserialize(medication_bundles)
```

---

medication_bundles *Exemplary FHIR bundles*

Description

This data example can be used to explore some of the functions from the fhircrackr package when direct access to a FHIR server is not possible.

Usage

```
medication_bundles
```

Format

List of length 3 containing *serialized* "xml_document" objects, each representing one bundle from a FHIR search request. *They have to be unserialized before use, see Examples!*

Details

medication_bundles is a list of *serialized* xml objects representing FHIR bundles as returned by fhir_search().

It contains 3 bundles with MedicationStatement resources representing Medications with Snomed CT code 429374003 and the respective Patient resources that are linked to these MedicationStatements.

It corresponds to the example of downloading and flattening FHIR resources from the vignette of the package.

Source

Generated by

```
fhir_search("https://hapi.fhir.org/baseR4/MedicationStatement?code=http://snomed.info/ct|429374003
_include=MedicationStatement:subject",max.bundles = 3)
```

[Downloaded 06-22-20]

Examples

```
#unserialize xml objects before doing anything else with them!
medication_bundles <- fhir_unserialize(medication_bundles)
```

paste_paths *Concatenate paths*

Description

Concatenates two strings to path string correctly.

Usage

```
paste_paths(path1 = "w", path2 = "d", os = "LiNuX")
```

Arguments

path1	A string specifying the left hand part of the resulting path.
path2	A string specifying the right hand part of the resulting path.
os	A string specifying the operating system you're operating on: windows or linux.

Value

A string containing the concatenated path.

Examples

```
paste_paths("data", "patients")
paste_paths("/data", "patients")
paste_paths("/data/", "patients")
paste_paths("/data", "/patients")
paste_paths("/data/", "/patients/")
paste_paths("data", "patients", "windows")
```

patient_bundles *Exemplary FHIR bundles*

Description

This data example can be used to explore some of the functions from the fhircrackr package when direct access to a FHIR server is not possible.

Usage

```
patient_bundles
```

Format

List of length 2 containing *serialized* "xml_document" objects, each representing one bundle from a FHIR search request. *They have to be unserialized before use, see Examples!*

Details

patient_bundles is a list of *serialized* xml objects representing FHIR bundles as returned by fhir_search(). It contains 2 bundles with Patient resources.

Source

Generated by:

```
fhir_search(request="http://fhir.hl7.de:8080/baseDstu3/Patient?",max_bundles=2)
```

[Downloaded 07-07-20]

Examples

```
#unserialize xml objects before doing anything else with them!
patient_bundles <- fhir_unserialize(patient_bundles)
```

Index