ndexr - interface with the network data exchange (NDEx)

Florian J. Auer[1]

[1]IT-Infrastructure for Translational Medical Research, University of Augsburg

03/30/2022

Contents

1 Load libraries

```r
library(ndexr)
library(RCX)
```

2 Introduction

The Network Data Exchange, or NDEx, is an open-source software framework to manipulate, store and exchange networks of various types and formats (Pratt et al., 2015, Cell Systems 1, 302-305, October 28, 2015 ©2015 Elsevier Inc. ScienceDirect). NDEx can be used to upload, share and publicly distribute networks, while providing an output in formats, that can be used by plenty of other applications.

This package provides an interface to query the public NDEx server (https://www.ndexbio.org/), as well as private installations, in order to upload, download or modify biological networks.

This document aims to help the user to install and benefit from the wide range of funtionality of this implementation. The package also provides classes to implement the Cytoscape Cyberinfrastructure (CX) Format and to extend the [iGraph Package] (http://igraph.org/r/).

ndexr is compatible with both NDEx API versions 1.3 and 2.0.

For a complete documentation see the package vignette and reference manual at Bioconductor:

https://bioconductor.org/packages/ndexr

3 Quick Start

Some short overview of the most important functions:

```r
## login to the NDEx server
ndexcon <- ndex_connect("username", "password")
```

```r
## search the networks for 'EGFR'
networks <- ndex_find_networks(ndexcon, "EGFR")

## UUID of the first search result
networkId <- networks[1, "externalId"]

## get summary of the network
networkSummary <- ndex_network_get_summary(ndexcon, networkId)

## get the entire network as RCX object
rcx <- ndex_get_network(ndexcon, networkId)

## remove NDEx artefacts from network
rcx <- rcx_asNewNetwork(rcx)

## do some fancy stuff with the network, then update
## the meta-data
rcx <- rcx_updateMetaData(rcx)
```

```r
## upload network as a new network to the NDEx server
networkId <- ndex_create_network(ndexcon, rcx)

## do some other fancy stuff with the network, then
## update the network on the server
networkId <- ndex_update_network(ndexcon, rcx)

## realize, you did bad things to the poor network, so
## better delete it on the server
ndex_delete_network(ndexcon, networkId)
```

4 Connect to a server

First, establish an connection to the NDEx server. This object is required for most of the other ndexr functions, because it stores options and authentication details. It is possible to connect to the server anonymously or provide a username and password to enable further functionality.

```r
## connect anonymously
ndexcon <- ndex_connect()

## log in with user name and password
ndexconUser <- ndex_connect(username = "username", password = "password")
```

This package is developed following the structure of the documented api structure. For complete description of the NDEx server api see *http://www.home.ndexbio.org/using-the-ndex-server-api/*. The R functions are named by the category, context and function they fulfill. In the following, the usage is described in detail, and hopefully gives a better understanding of logic behind the naming convention of this package.

5 Find Networks

To explore or search the networks on an NDEx server, this package offers a function to retrieve a list of networks from the server. It is possible to restrict the networks to a specific search string (e.g. "EGFR"), an account name (only networks of this account will be shown), or limit the number of fetched networks.

```r
## list networks on server
networks <- ndex_find_networks(ndexcon)
## same as previous
networks <- ndex_find_networks(ndexcon, start = 0, size = 5)

## search for 'EGFR'
networksEgfr <- ndex_find_networks(ndexcon, searchString = "EGFR")
## same as previous
networksEgfr <- ndex_find_networks(ndexcon, "EGFR")
```

```
## same as previous
networksOfUser <- ndex_find_networks(ndexcon, accountName = "ndextutorials")
```

As result you get a data.frame containing information of the networks.

```
names(networks)
##  [1] "ownerUUID"        "isReadOnly"       "subnetworkIds"    "isValid"          "warnings"         "isSho
##  [8] "indexLevel"       "hasLayout"        "hasSample"        "cxFileSize"       "cx2FileSize"      "visil
## [15] "edgeCount"        "version"          "completed"        "owner"            "description"      "name"
## [22] "externalId"       "isDeleted"        "modificationTime" "creationTime"

print(networks[, c("name", "externalId")])
##                                                  name                           externalId
## 1 San Diego Research Connectivity Map (Top Journals) 9936b382-b06f-11ec-b3be-0ac135e8bacf
## 2                  San Diego Research Connectivity Map c2751109-b08a-11ec-b3be-0ac135e8bacf
## 3                                              covid19 a8c0decc-6bbb-11ea-bfdc-0ac135e8bacf
## 4                                         painmachine e1e5963a-eb0e-11e9-bb65-0ac135e8bacf
## 5                                          rasmachine cdba5bd5-5195-11e9-9f06-0ac135e8bacf
```

6 Simple network operations

To both, users and networks stored on an NDEx server, a universally unique identifier (UUID) is assigned. Although both have the same format (i.e. “xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx”, where x is one of `[a-z0-9]`), it has to be distinguished between the user UUID and the network UUID, but the difference is obvious by the context. Within RCX objects and search results, the network UUID is also referred to as “externalId” (see previous section). This UUID can be used to access a network on the server and retrieve just a summary of the network (similar to the results of a network search) or even the entire network as RCX object (see next section).

Since networks can contain many nodes and edges, and a huge amount of other attributes, it is typically advisable to first get a network summary, to check the node and edge counts for a network before retrieving the entire network. Thereby the structure of the network summary is similar the structure of the network list

```
## UUID of the first search result
networkId <- networksOfUser[1, "externalId"]

## get network summary
networkSummary <- ndex_network_get_summary(ndexcon, networkId)

names(networkSummary)
##  [1] "ownerUUID"        "isReadOnly"       "subnetworkIds"    "isValid"          "warnings"         "isSho
##  [8] "indexLevel"       "hasLayout"        "hasSample"        "cxFileSize"       "cx2FileSize"      "visil
## [15] "edgeCount"        "version"          "completed"        "owner"            "description"      "name"
## [22] "externalId"       "isDeleted"        "modificationTime" "creationTime"

networkSummary[c("name", "externalId")]
## $name
## [1] "Metabolism"
##
```

```
## $externalId
## [1] "3bd76cf4-c4a1-11e4-bcc4-000c29cb28fb"

## get the entire network as RCX object
rcx <- ndex_get_network(ndexcon, networkId)
```

To send a network to an server, there are two possibilities. Either one wants to update an existing network on the server or create a new one. In both cases, a UUID is returned, either of the updated network or a newly generated one for the created network. For updating a network, the UUID is extracted from the "externalId" property of the "ndexStatus" aspect, or can be set manually.

```
## create a new network on server
networkId <- ndex_create_network(ndexcon, rcx)

## update a network on server
networkId <- ndex_update_network(ndexcon, rcx)

## same as previous
networkId <- ndex_update_network(ndexcon, rcx, networkId)
```

Besides creating, reading and updating, it is also possible to delete networks on the server. This operation cannot be undone, so be careful!

```
## deletes the network from the server
ndex_delete_network(ndexcon, networkId)
```

7 Example Workflow

This example workflow shows how to connect to the public NDEx server, browse and retrieve the pathways of the Pathway Interaction Database of the NCI which are hosted there.

```
## load the library!
library(ndexr)

## login to the NDEx server
ndexcon <- ndex_connect()

## retrieve pathways of user 'nci-pid'
networks_pid <- ndex_find_networks(ndexcon, accountName = "nci-pid")

## list retrieved network information
networks_pid[, "name"]

## show information on the first pathways listed
networks_pid[1, ]

## retrieve network data
mynetwork <- ndex_get_network(ndexcon, networks_pid[1, "externalId"])
```

This code snippet starts with loading the ndexr library and connecting to the server anonymously. Afterwards `ndex_find_networks` retrieves a list of networks of user `nci-pid`, which contains the data of the Pathway Interaction Database. The function `ndex_get_network` downloads the network data and stores in the `RCX` format.

8 Aspects and Metadata

In general it is not advisable to retrieve a complete RCX object from a server without knowing the number of aspects and its corresponding size, because this may cause unwanted or unnecessary network traffic and decline in performance. To avoid these problems, a possible workflow is to download the meta-data of a network at first to check the available aspects.

```
## get meta-data for a network
metadata <- ndex_network_get_metadata(ndexcon, networkId)

names(metadata)
## [1] "name"         "elementCount" "version"      "idCounter"

print(metadata[c("name", "elementCount")])
##                    name elementCount
## 1      nodeAttributes         6259
## 2  cyHiddenAttributes            3
## 3                 nodes         6376
## 4   networkAttributes            7
## 5       cyTableColumn           19
## 6     cartesianLayout         6376
## 7      edgeAttributes        56924
## 8                 edges        56924
## 9  cyVisualProperties            3
## 10            citations         2916
## 11       edgeCitations        44129
```

Afterwards, only the favored aspects can be downloaded individually.

```
## get aspect 'nodeCitations' for the network
networkAttibutes <- ndex_network_get_aspect(ndexcon, networkId, "networkAttributes")

print(networkAttibutes)
##                  n
## 1      description
## 2          version
## 3              URI
## 4         @context
## 5           Source
## 6             name
## 7 ndex:sourceFormat
##
## 1
## 2
## 3
## 4 {"KEGG GLYCAN":"http://identifiers.org/kegg.glycan/","RAT GENOME DATABASE":"http://identifiers.org/rgd/"
## 5
## 6
## 7
```

9 NDEx Network properties

Even after creation, it is possible to change the name, the description or version of a network.

```r
ndex_network_update_profile(
    ndexcon, networkId, name = "My network", version = "1.3"
)
ndex_network_update_profile(
    ndexcon, networkId, description = "Nothing to see here"
)
```

For collaborative work, it is necessary to share networks between several users and groups. Therefore there are specialized functions to grant access to a network, change the permissions and withdraw access permissions. It is possible to use those functions on single users or groups. Possible permissions are "READ" to have reading access to private networks, "WRITE" to be able modify, and "ADMIN" for the owner of the network.

```r
## show all user who have permission to a network
permissions <- ndex_network_get_permission(ndexcon, networkId, "user")

## show all groups who have permission to a network
permissions <- ndex_network_get_permission(ndexcon, networkId, "group")

## show all users with write access to a network
permissions <- ndex_network_get_permission(ndexcon, networkId, "user", "WRITE")

## grant an user permission to a network
ndex_network_update_permission(ndexcon, networkId, user = someUserUuid, "READ")

## change the permission of an user to the network
ndex_network_update_permission(ndexcon, networkId, user = someUserUuid, "WRITE")

## withdraw the permission from an user
ndex_network_delete_permission(ndexcon, networkId, user = someUserUuid)
```

Besides permission management on user and group level, it is also possible to set some system properties on a network that influence the accessibility further. By default a network is private, which means that it is only visible to the owner and invited users and groups. If at some point one decides to make the network readable by anyone, it is possible to change the visibility of a network to "PUBLIC".

```r
ndex_network_set_systemProperties(ndexcon, networkId, visibility = "PUBLIC")
ndex_network_set_systemProperties(ndexcon, networkId, visibility = "PRIVATE")
```

When a network has reached the point to be published, further edits should be prevented. While it would be possible to set the access permissions of all users and groups to “READ”, this approach is very inconvenient. Therefore, a simpler way is to just set the network to read-only using the network system properties.

```
ndex_network_set_systemProperties(ndexcon, networkId, readOnly = TRUE)
```

One also has the option at the NDEx server to choose a selection of their favorite networks for display in his or her home page.

```
ndex_network_set_systemProperties(ndexcon, networkId, showcase = TRUE)
ndex_network_set_systemProperties(ndexcon, networkId, showcase = FALSE)

# change more than one property simultaneously
ndex_network_set_systemProperties(
    ndexcon, networkId, readOnly = TRUE, visibility = "PUBLIC",
    showcase = TRUE
)
```

10 Session info

```
sessionInfo()
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.3 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=de_DE.UTF-8        LC_COLLATE=en_US.UT
##  [6] LC_MESSAGES=en_US.UTF-8    LC_PAPER=de_DE.UTF-8       LC_NAME=C                  LC_ADDRESS=C
## [11] LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] XML_3.99-0.8        httr_1.4.2          RJSONIO_1.3-1.6     pacman_0.5.1        devtools_2.4.2      u
##  [8] timeDate_3043.102   pander_0.6.4        xtable_1.8-4        stringr_1.4.0       BiocStyle_2.18.1    F
## [15] igraph_1.2.7        gplots_3.1.1        dplyr_1.0.7         RColorBrewer_1.1-2  survival_3.2-7
##
## loaded via a namespace (and not attached):
##  [1] pkgload_1.2.3       jsonlite_1.7.2      splines_4.0.3       gtools_3.9.2        assertthat_0.2.1    B
##  [8] stats4_4.0.3        remotes_2.4.1       yaml_2.2.1          sessioninfo_1.1.1   pillar_1.6.4        l
## [15] digest_0.6.28       htmltools_0.5.2     Matrix_1.2-18       plyr_1.8.6          pkgconfig_2.0.3     b
## [22] webshot_0.5.2       processx_3.5.2      tibble_3.1.5        generics_0.1.1      ellipsis_0.3.2      L
## [29] cachem_1.0.6        BiocGenerics_0.36.1 cli_3.0.1           mime_0.12           magrittr_2.0.1      c
## [36] memoise_2.0.0       evaluate_0.14       fs_1.5.0            fansi_0.5.0         pkgbuild_1.2.0      g
## [43] tools_4.0.3        formatR_1.11       lifecycle_1.0.1    callr_3.7.0        compiler_4.0.3     j
## [50] tinytex_0.34       rlang_0.4.12       grid_4.0.3         htmlwidgets_1.5.4  crosstalk_1.1.1    b
## [57] testthat_3.1.0     DBI_1.1.1          curl_4.3.2         markdown_1.1       R6_2.5.1           h
## [64] utf8_1.2.2         rprojroot_2.0.2    desc_1.4.0         KernSmooth_2.23-17 stringi_1.7.5      p
## [71] vctrs_0.3.8        tidyselect_1.1.1   xfun_0.27
```