A brief introduction to the data structures

The Network Data Exchange (NEx), 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). NEx 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.

The ndexr package provides an interface to query the public NEx server, 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 functionality 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/).

**NEx Network Data Model – The CX Format**

CX is a network exchange format, designed as a flexible structure for transmission of networks. It is designed for flexibility, modularity, and extensibility, and as a message payload in common REST protocols. It is not intended as an optimized format for use in applications or for storage.

CX is "aspect-oriented", meaning that different types of information about network elements are separated into types of modules ("aspects"). Each aspect type specifies a schema for the information that it contains, typically a set of elements where the elements for that aspect type also have a defined schema. There are design guidelines for dependencies between aspects; the most basic guideline is that dependencies should be simple and minimal.

The flexibility of CX enables straightforward strategies for lossless encoding of potentially any network. At the most basic level, this means that CX imposes very few restrictions: graphs can be cyclic or acyclic and edges are implicitly directed, but formats can choose annotations schemes to override this. CX does not, itself, make any commitment to a single “correct” model of biology or graphic markup scheme.
CX is designed to facilitate streaming, potentially reducing memory footprint burden on applications processing large CX networks. In particular, in a CX stream, the elements of each aspect are broken into fragments and the fragments can be transmitted in any order.

Further information can be found at http://www.home.ndexbio.org/data-model/

**R representation of the CX format**

For the exchange of network data, NDEEx uses the Cytoscape Cyberinfrastructure Network Interchange Format (CX). CX is an Aspect-Oriented Network Interchange Format encoded in JSON, which is used as basis for the R implementation of the CX format, namely RCX.

The RCX object is currently implemented within this package as a list of data.frames, containing meta-data and all aspects of the network. The structure of an RCX object, as shown via str(rcx) could be a list like this:

```r
str(rcx, max.level = 2)
#> List of 13
#>$ metaData : 'data.frame': 11 obs. of 7 variables:
#> ..$ elementCount : int [1:11] 1 1 395 667 328 299 3434 160 99 3 ...
#> ..$ lastUpdate : num [1:11] 1.49e+12 1.49e+12 NA NA NA ...}
#> ..$ name : chr [1:11] "ndexStatus" "provenanceHistory" "nodes" "edges" ...
#> ..$ properties :List of 11
#> ..$ version : chr [1:11] "1.0" "1.0" NA NA ...
#> ..$ consistencyGroup: int [1:11] NA NA NA NA NA NA NA ...
#> ..$ idCounter : int [1:11] NA NA 1689 1689 1689 1689 NA NA NA ...
#> ..$ numberVerification: 'data.frame': 1 obs. of 1 variable:
#> ..$ longNumber: num 2.81e+14
#> ..$ ndexStatus : 'data.frame': 1 obs. of 10 variables:
#> ..$ externalId : chr "7aedd4dd-14e4-11e6-af8-06603eb7f303"
#> ..$ creationTime : num 1.46e+12
#> ..$ modificationTime: num 1.49e+12
#> ..$ visibility : chr "PUBLIC"
#> ..$ published : logi FALSE
#> ..$ nodeCount : int 395
#> ..$ edgeCount : int 667
#> ..$ owner : chr "rasmachine"
#> ..$ ndexServerURI : chr "http://public.ndexbio.org"
#> ..$ readOnly : logi FALSE
#> ..$ @id: int [1:395] 0 1 5 10 11 20 27 28 32 33 ...
#> ..$ n : chr [1:395] "ABCC8" "ATP" "CHI" "ADCY1" ...
#> ..$ @id: int [1:667] 2 6 12 21 29 34 38 43 53 58 ...
#> ..$ s : int [1:667] 0 10 20 27 32 1 11 52 52 ...
#> ..$ t : int [1:667] 1 5 11 28 33 10 42 10 57 ...
#> ..$ i : chr [1:667] "Activation" "Activation" "Activation" "Activation" ...
#> ..$ supports : 'data.frame': 328 obs. of 4 variables:
```
The data.frames representing nodes and edges could look like this:

rcx[['nodes']][1:5,]

# @id n
# 1 0 ABCC8
# 2 1 ATP
# 3 5 CHI
# 4 10 ADCY1
# 5 11 CAMP
Usually, an RCX object is automatically created by using the functions of this package for downloading network data from a NDEx server. But it might be useful to convert an RCX object from/to JSON manually, for example for down-/uploading a CX file from/to a NDEx server via the web interface.

### iGraph conversion of RCX object

For handling the network information within R, besides RCX objects, one can use RCXgraph objects. A lossless conversion between the two files can be done using the following functions:

```r
## convert RCX to JSON
json <- rcx_toJSON(rcx)

## ...and back
rcx <- rcx_fromJSON(json)

## convert RCX to RCXgraph
rcxgraph <- rcx_toRCXgraph(rcx)

## ...and back
rcx <- rcxgraph_toRCX(rcxgraph)
```

These functions convert RCX to igraph. This means aspects to graph, node and edge attributes. Map aspects to the respective nodes, edges or create custom attribute tables for further graph data. [iGraph Package](http://igraph.org/r/)

### Demo

Some short overview of the most important functionality and to plot a NDEx network.

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

## login to the NDEx server
ndexcon = ndex_connect("username", "password")

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

## 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)

## show structure of the RCX object (see section on RCX in the document)
str(rcx)

## convert RCX to RCXgraph
rcxgraph <- rcx_toRCXgraph(rcx)

## show graph information (see section on iGraph object in the document)
rcxgraph

## use readable node names instead of IDs and plot the graph
V(rcxgraph)[as.character(rcxgraph$nodes[,"@id"])$name = rcxgraph$nodes[,"n"]]
plot(rcxgraph)