ABSTRACT

We introduce ICON, an R package that contains 1075 complex network datasets in a standard edgelist format. All provided datasets have associated citations and have been indexed by the Colorado Index of Complex Networks - also referred to as ICON. In addition to supplying a large and diverse corpus of useful real-world networks, ICON also implements an S3 generic to work with the network and ggnetwork R packages for network analysis and visualization, respectively. Sample code in this report also demonstrates how ICON can be used in conjunction with the igraph package. Currently, the Comprehensive R Archive Network hosts ICON v0.4.0. We hope that ICON will serve as a standard corpus for complex network research and prevent redundant work that would be otherwise necessary by individual research groups. The open source code for ICON and for this reproducible report can be found at https://github.com/rrrlw/ICON.

Keywords complex networks · R programming language
2.1 Installation

The stable version of ICON (currently v0.4.0) can be downloaded from CRAN, while the development version can be downloaded from the package’s GitHub repository using the remotes package. Both options are demonstrated in the following code chunk.

```r
# install stable version from CRAN
install.packages("ICON")

# install development version from GitHub
remotes::install_github("rrrlw/ICON", build_vignettes = TRUE)
```

2.2 Complex network datasets

Currently, ICON provides 1075 complex network edge lists. The largest network, named `amazon_copurchase`, consists of 3,387,388 edges. Due to the large volume of data and CRAN package size limits, all of ICON’s networks cannot be downloaded to a local machine upon installation and loaded with `utils::data`. Instead, the package’s GitHub repository contains a directory named `data-host`, which `ICON::get_data` accesses to download networks named by the user. After successful download, `ICON::get_data` loads these networks into the user’s environment of choice (default: `.GlobalEnv`) and cleans any intermediate artifacts. To avoid dependence on an internet connection, users can save and access individual networks in RDS format (binary; `.rds` extension; via `base::saveRDS` and `base::readRDS`) or do the same for a set of networks in RData/RDa format (binary; `.RData` or `.rda` extension; via `base::save` and `base::load`). An obvious deficiency of this system is the inability to take advantage of automated data documentation and checking tools, such as roxygen2. However, the `ICON::ICON_data` dataset provides the necessary documentation for ICON users and implements a sufficient and slightly soporific checking system for the package authors.

Although providing standardized data format avoids redundant work, an important processing step being completed by a single party (package authors) opens the door to inaccuracies. It befits us to simply counter this limitation with ICON’s status as free, open-source software (FOSS), which offers every user the opportunity to inspect, question, and correct all aspects of ICON. The `data-raw` directory in ICON’s GitHub repository follows Wickham’s (2015) advice and contains: (1) the original raw data acquired directly from the source indexed by the ICON website; (2) the R code that converts each raw dataset into a data frame comprised of an edge list and potential edge attributes; and (3) the R code saving the resulting data frame as an RDA file in the aforementioned `data-host` directory. We hope that this not only offers, but indeed encourages, ICON users to confirm dataset accuracy.

Note that to minimize unnecessary package elements, ICON’s `.Rbuildignore` contains `data-host` and `data-raw`. However, for reproducibility and documentation, ICON’s GitHub repository provides public access to both directories.

We will now look at sample code to acquire complex network datasets using ICON. To do so, we must load the library in the R session and load the `ICON_data` dataset, which contains relevant complex network metadata. The metadata can be explored in the package documentation with ?ICON_data; in this report, we will focus only on the essentials, starting with the following code chunk.

```r
# load library
library("ICON")

# load metadata
# explore this data frame to figure out which networks suit your needs
data("ICON_data")

# peek at the first few and last few packages available to download
head(ICON_data$Var_name, n = 3)
## [1] "aishihik_intensity" "aishihik_prevalence" "amazon_copurchase"

tail(ICON_data$Var_name, n = 3)
## [1] "wordadj_french" "wordadj_japanese" "wordadj_spanish"
```

We first try downloading a single dataset with `ICON::get_data` and peeking at its contents. Once this succeeds, we confidently download multiple datasets.
# download single dataset named in previous code chunk output
# could also use `get_data(ICON_data$Var_name[1])` to same effect
get_data("aishihik_intensity")

## DATASET(S) aishihik_intensity LOADED

# look at the structure of the complex network
str(aishihik_intensity)

## Classes 'ICON' and 'data.frame': 78 obs. of 3 variables:
## $ Fish : chr "1" "1" "1" "1" ... 
## $ Parasite : chr "V1" "V9" "V16" "V22" ... 
## $ Intensity: num 5.8 7 3 1 7.2 ... 

# confirm that metadata reflects the correct number of edges
(ICON_data$Edges[1] == nrow(aishihik_intensity))

## [1] TRUE

# look at the first few rows; for all ICON datasets:
# columns 1 and 2 = nodes that define the edge
# columns 3 and beyond = edge attributes (e.g. weight)
head(aishihik_intensity)

## Fish Parasite Intensity
## 1 1 V1 5.8
## 2 1 V9 7.0
## 3 1 V16 3.0
## 4 1 V22 1.0
## 5 2 V3 7.2
## 6 2 V8 65.8

# download multiple datasets
get_data(c("wordadj_japanese", "wordadj_french"))

## DATASET(S) wordadj_japanese wordadj_french LOADED

# confirm downloads by looking at internal structure
str(wordadj_japanese)

## Classes 'ICON' and 'data.frame': 8300 obs. of 2 variables:
## $ From: chr "2" "3" "3" "3" ... 
## $ To : chr "3" "4" "5" "9" ...

str(wordadj_french)

## Classes 'ICON' and 'data.frame': 24295 obs. of 2 variables:
## $ From: chr "1" "1" "1" "1" ... 
## $ To : chr "2" "5" "7" "30" ...

A keen reader might observe that all of ICON's datasets could be downloaded with `get_data(ICON_data$Var_name)`; due to the potential runtime and memory commitment, we strongly recommend that users exercise caution if attempting this.

2.3 The ICON class and network S3 generic

Looking at the structure of the complex networks with `utils::str` shows that ICON complex networks all have two classes: `ICON` and `data.frame`. The latter provides a suitable container for edge list objects with potential edge
attributes in rectangular format. The former, an S3 class, benefits users in two ways. First, it provides certain guarantees about object format, i.e., an unmodified complex network object acquired via ICON will have the ICON S3 class and is guaranteed to be a data frame containing an edge list in which each row represents a single edge, the first two columns specify nodes that define the corresponding edge, and additional columns define edge attributes. This standard format guarantee allows users, among other things, to generate code for one ICON dataset with assurances that it will function effectively for other ICON datasets. Second, the S3 class allows users to take advantage of relevant S3 generics. ICON specifically defines the as.network.ICON method for compatibility with the as.network generic in the network R package. This allows users to easily analyze and visualize ICON datasets with the reliable network and ggnetwork R packages, respectively. Section 3.1 will provide sample code that highlights the benefit of this feature.

3 Use cases

Before starting with the use cases, the following code chunk will load the appropriate libraries and download the sample dataset.

```r
# load necessary libraries
library("ICON")
library("network")
library("ggnetwork")
library("ggplot2")
library("igraph")

# for reproducibility
set.seed(42)

# download sample dataset
get_data("seed_disperse_beehler")
```

## DATASET(S) seed_disperse_beehler LOADED

A quick exploration of seed_disperse_beehler will grant a deeper understanding of the use cases. Primarily, we would like to explore the third column - named Frequency. Due to the heavy skew, we will use two consecutive logarithmic transformations to more easily see the effects of coloring edges by the Frequency edge attribute. The following code chunk produces histograms of seed_disperse_beehler$Frequency before and after this transformation for comparison.

```r
# plot a histogram w/o transformation (skewed, tough to see differences)
ggplot(seed_disperse_beehler, aes(x = Frequency)) +
  geom_histogram(bins = 10, fill = "white", color = "black") +
  theme_bw()

# plot a histogram w/ transformation (more spread out, differences easily seen)
 ggplot(seed_disperse_beehler, aes(x = log(log(Frequency)))) +
  geom_histogram(bins = 10, fill = "white", color = "black") +
  theme_bw()
```

![](image1.png)
3.1 With the network R package

Using the `seed_disperse_beehler` sample dataset, we first coerce it to have class `network` with the `as.network` method under the `as.network` S3 generic. This allows us to take advantage of the large set of tools already built in the [Statnet suite] of R packages. Although we first use `ggnetwork` to rapidly visualize the nodes and edges, we also show how to visualize edge attributes toward the end of the code chunk.

```r
# setup using as.network generic
coerced <- as.network(seed_disperse_beehler)

# plot with ggnetwork
ggplot(coerced, aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_edges(alpha = 0.25) +
  geom_nodes() +
  theme_blank()
```

# are there any edge attributes in 'seed_disperse_beehler'?
# YES, we have the "Frequency" edge attribute (see third column name)
str(seed_disperse_beehler)

```r
## Classes 'ICON' and 'data.frame': 119 obs. of 3 variables:
## $ Bird : chr "1" "1" "1" "1" ... 
## $ Plant : chr "V4" "V6" "V8" "V11" ... 
## $ Frequency: num 1 1 1 25 19 1 1 8 1 1 ... 
```

# is this edge attribute also present in the coerced network?
# YES, let's plot it in the next network visualization (see end of output)
print(coerced)

```r
## Network attributes:
## vertices = 40
## directed = FALSE
```
## hyper = FALSE
## loops = FALSE
## multiple = FALSE
## bipartite = FALSE
## total edges= 119
## missing edges= 0
## non-missing edges= 119
##
## Vertex attribute names:
## vertex.names
##
## Edge attribute names:
## Frequency

```r
# plot with \text{log}(\log(\text{Frequency})) as an edge attribute (edge color)

ggplot(coerced, aes(x = x, y = y, xend = xend, yend = yend)) +
geom_edges(aes(color = \text{log}(\log(\text{Frequency})))) +
geom_nodes() +
theme_blank()
```

Of course, even with the edge attribute, we are only scraping the surface of the visualization capability provided. More details can be found in the documentation of the appropriate packages.

### 3.2 With the \texttt{igraph} R package

The \texttt{igraph} package provides a rich set of network analysis and visualization tools. As a consequence, the installed package is large. Making \texttt{ICON} strongly dependent on \texttt{igraph} by building an \texttt{as.igraph.ICON} method for the \texttt{as.igraph} \texttt{S3} generic would require all \texttt{ICON} users to also install \texttt{igraph}. To avoid this, \texttt{ICON} does not strongly depend on \texttt{igraph}, however, the following code chunk and the \texttt{GitHub README} demonstrate how to use \texttt{ICON} in conjunction with \texttt{igraph} to emphasize that, despite the size limitation that excluded it from being an \texttt{ICON} dependency, it is a powerful tool. Instead of coercing \texttt{seed\_disperse\_beehler} into an object with class \texttt{igraph}, we transfer the
information in `seed_disperse_beehler` to a new igraph object with `igraph::graph_from_edgelist` and then visualize the network using the `igraph::plot.igraph` method for the `base::plot` S3 generic.

```r
# plot network using igraph
my_graph <- graph_from_edgelist(as.matrix(seed_disperse_beehler[, 1:2]),
                                 directed = FALSE)
plot(my_graph,
     vertex.label = NA, vertex.size = 5)
```

As was the case with the previous use case, we have only scratched the surface of visualization possibilities. More details can be found in the igraph package documentation.

4 Discussion/Conclusions

We have introduced the ICON R package, explained its potential use as a network corpus, and demonstrated its compatibility with existing complex network software. With time, we hope that ICON’s corpus will grow and encourage users to contribute complex network datasets by following steps in the package’s contributing guidelines and adhering to the code of conduct,

5 Acknowledgments

The authors thank Aaron Clauset, PhD and the members of his research group at the University of Colorado Boulder for advice and for their tireless efforts in creating the Colorado Index of Complex Networks.

References

1. Ghasemian A, Hosseinmardi H, Clauset A. Evaluating overfit and underfit in models of network community structure. IEEE Transactions on Knowledge and Data Engineering 2020;32:1722-1735.
2. Broido AD, Clauset A. Scale-free networks are rare. Nature Communications 2019;10:1017.

3. Clauset A, Tucker E, Sainz M. The Colorado Index of Complex Networks., 2016: Available at https://icon.colorado.edu.

4. R Core Team. R: A Language and Environment for Statistical Computing. Vienna, Austria: R Foundation for Statistical Computing, 2020: Available at https://www.R-project.org.

5. Butts C. network: A package for managing relational data in R. Journal of Statistical Software 2008;24:1-36.

6. Hester J. covr: Test Coverage for Packages., 2020: Available at https://CRAN.R-project.org/package=covr.

7. Briatte F. ggnetwork: Geometries to Plot Networks with ’ggplot2’. 2020: Available at https://CRAN.R-project.org/package=ggnetwork.

8. Wickham H. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag: New York, NY, 2016.

9. Xie Y. Dynamic Documents with R and Knitr. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC, 2015.

10. Wickham H. testthat: get started with testing. The R Journal 2011;3:5-10.

11. Xie Y, Allaire JJ, Grolemund G. R Markdown: The Definitive Guide. Boca Raton, Florida: Chapman; Hall/CRC, 2018. https://bookdown.org/yihui/rmarkdown.

12. Wickham H, Hester J, Chang W. devtools: Tools to Make Developing R Packages Easier., 2020: Available at https://CRAN.R-project.org/package=devtools.

13. Hester J, Csárdi G, Wickham H, et al. remotes: R Package Installation from Remote Repositories, Including 'Github’. 2020: Available at https://CRAN.R-project.org/package=remotes.

14. Wickham H, Danenberg P, Csárdi G, et al. roxygen2: in-Line Documentation for R., 2020: Available at https://CRAN.R-project.org/package=roxygen2.

15. Contributor Covenant: A Code of Conduct for Open Source Projects, 2014: Available at https://www.contributor-covenant.org.

16. Wickham H. R Packages. O’Reilly Media: Sebastopol, CA, 2015.