Extensions to langnetr: Degree distributions

CSQ Siew

2017-10-30

For some reason the package works fine in R but not in the .Rmd file, so I can't render the doc fully... -CS

Vignette Info

The `langnetr` package was built to help the user easily convert lists of words into network objects based on an edit distance of 1. A set of functions (starting with "dd.") were added to this package to make it easy for the user to analyze the degree distributions of the language network. This vignette will demonstrate how to use the "degree distribution" functions in this package using made up data.

Note that the process for fitting a power law (and other distributions) to the data was directly replicated from the `poweRlaw` R package - https://github.com/csgillespie/poweRlaw. These are a merely a set of helper functions to automate the processes specific to the needs of a language researcher (me!).

Set up and create a fake distribution for demonstration

```
library(langnetr)
library(igraph)

# make a random small world network
network <- watts.strogatz.game(1, 5000, 1, 0.35, loops = FALSE, multiple =
FALSE)
net.degree <- as.data.frame(table(as.data.frame(degree(network))))
distribution <- net.degree$Freq # tabulated degrees of p.net, the "raw" data
```

Fit various distributions to the data

A power law, log-normal, exponential, and Poisson distributions were fitted to the raw data. The `estimates` argument is used to decide what outputs should be returned. If `estimates = T` the optimal xmin and estimated parameters are returned. If `estimates = F` the fitted model is returned and should be saved as an object for subsequent analyses.

```
fitpl(distribution, estimates = T) # get estimates

## [1] 1.000 1.197
```

```
model.pl <- fitpl(distribution, estimates = F) # get fitted model

fitln(distribution, estimates = T)

## [1] 71.000  6.073  1.266

model.ln <- fitln(distribution, estimates = F)

fitex(distribution, estimates = T)

## [1] 11.000  0.002

model.ex <- fitex(distribution, estimates = F)

fitpo(distribution, estimates = T)

## [1] 334 983

model.po <- fitpo(distribution, estimates = F)
```

Visualize the fits

Plots a nice-ish figure of the raw data with the fits of various distributions.

```
visualize(distribution)
```

Get uncertainty estimates for power law fit

The `getplsd()` function characterizes the uncertainty of the parameter estimates through bootstrapping and the results of the bootstrap can be visualized. If `getraw = F` summary statistics from the bootstrap are returned. If `getraw = T` the raw output of the bootstrap is returned.

```
getplsd(model.pl, getraw = F) # get estimates
bsresults <- getplsd(model.pl, getraw = T) # returns the raw outputs of the
bootstrap

# visualize the output of the bootstrap using the raw bootstrap output
hist(bsresults$bootstraps[,2], xlab = 'xmin estimates',
     main = 'xmin bootstrap estimates') # histogram of uncertainty of xmin
hist(bsresults$bootstraps[,3], xlab = 'alpha estimates',
     main = 'alpha bootstrap estimates') # histogram of uncertainty of alpha

plot(bsresults, trim=0.1) # trim=0.1 only displays the final 90% of
iterations
# The top row shows the mean estimate of parameters xmin, alpha and ntail.
# The bottom row shows the estimate of standard deviation for each parameter.
# The dashed-lines give approximate 95% confidence intervals.
```

Test the power law fit to the data

Because it is possible to fit a power law to any data, it is important to establish that the power fit **IS** in fact a good fit to the data. The `testpl()` function tests the goodness of fit of the power law to the data through bootstrapping and the results of the bootstrap can be visualized. If `getraw = F` summary statistics from the bootstrap are returned. If `getraw = T` the raw output of the bootstrap is returned.

```
testpl(model.pl, getraw = F) # get estimates
bspresults <- testpl(model.pl, getraw = T) # returns the raw outputs of the bootstrap

# visualize the results
plot(bspresults, trim=0.1) # includes the p-value of the test of pl distribution
```

Comparing distributions

"A second approach to test the power law hypothesis is a direct comparison of two models. A standard technique is to use Vuong's test, which a likelihood ratio test for model selection using the Kullback-Leibler criteria. The test statistic, R, is the ratio of the log-likelihoods of the data between the two competing models. The sign of R indicates which model is better. Since the value of R is obviously subject to error, we use the method proposed by Vuong (1989)." *[from Gillespie's Vignette on Comparing Distributions]*

```
cpdist(model.pl, model.ln, output = F) # returns 2-sided and 1-sided p-values

## [1] "The two-sided p-value is 0.28"
## [1] "The one-sided p-value is 0.86"

cpdist(model.pl, model.ex, output = F)

## [1] "The two-sided p-value is 0.838"
## [1] "The one-sided p-value is 0.419"

cpdist(model.pl, model.po, output = F)

## [1] "The two-sided p-value is 0"
## [1] "The one-sided p-value is 0"
```

Here are the two competing hypotheses:

- H0: Both distributions are equally far from the true distribution
- H1: One of the test distributions is closer to the true distribution.

The results of the Vuong's test are very confusing. Here is what it means...

2-sided p-value:

- if significant, then H0 is rejected

- one of the test distributions is closer to the true distribution
- order of m1 and m2 does not matter for two-sided p
- no need to look at one-sided p if two-sided p is not sig (stick with H0)

1-sided p-value:

- order of m1 and m2 matters for 1-sided p # m1 = left, m2 = right # if sig, m1 is better # if not sig, m1 isn't better than m2 (but does that mean that m2 is better?)