Package 'latChanNet'

November 18, 2019

Type Package

Title Latent Channel Networks

Version 1.0

Date 2019-05-09

Author Clifford Anderson-Bergman

Maintainer Clifford Anderson-Bergman <pistacliffcho@gmail.com>

Description Analysis of undirected networks via Latent Channel Networks.

License GPL (>= 2)

biocViews

Imports Rcpp (>= 1.0.1), RcppParallel (>= 4.4.3), methods, ComplexHeatmap, circlize, mltools

LinkingTo Rcpp, RcppParallel

SystemRequirements GNU make

RoxygenNote 6.1.99.9001

Suggests testthat (>= 2.1.0)

NeedsCompilation yes

Depends R (>= 3.5.0)

R topics documented:

latChanNet-package *Latent Network Models for edge and metadata prediction.*

Description

Fits Latent Channel Networks and the Poisson model of Ball, Karrer and Newman (2011). Allows for unknown edges statuses. Augments network with metadata to allow for metadata predictions.

Details

Models are built (but not fitted) with `makeLatentModel`. Models are fit with the `$fit()` method. Fitted parameters can be extracting via the `$get_pars()` method. Predictions of both edges and metadata can be done with `predict`. Heatmaps of parameters can be plotted with `plot`.

Author(s)

Clifford Anderson-Bergman.

Maintainer: Clifford Anderson-Bergman <pistacliffcho@gmail.com>

References

Clifford Anderson-Bergman, Phan Nguyen, and Jose Cadena Pico. "Latent Channel Networks", submitted 2019

BKN model:

Brian Ball, Brian Karrer, and Mark EJ Newman. "Efficient and principled method for detecting communities in networks." Physical Review E 84.3 (2011): 036103.

Examples

```
  ## Not run:
     ## Optional simple examples of the most important functions
     ## These can be in \dontrun{} and \donttest{} blocks.

## End(Not run)
```

channel_sizes *Compute sizes of channels*

Description

Returns the size of each channel

Usage

```
channel_sizes(mod, type = "nodes_using")
```

Arguments

`mod`	LatClass model
`type`	How size is defined. Either 'nodes_using' or 'exp_connects'

Details

The size of each channel can be defined in two ways: number of nodes that have non-zero attachment to a channel ('nodes_using') *or* the expected number of connections through a channel a new node would have if it connected through that channel with probability 1 ('exp_connects') We note that the 'exp_connects' metric is a better description of size, but 'nodes_using' is more intuitive.

Examples

```
data(email_data)
mod = makeLatentModel(email_data$edgeList, 10,
                      metadata = email_data$meta)
mod$fit(fast_em = TRUE)

channel_sizes(mod, "exp_connects")
```

---

chan_connect *Estimate Channels Nodes Connect Through*

Description

Estimates probability two nodes are connected through a channel, *conditional on them having an edge.*

Usage

```
chan_connect(i, j = NULL, model)
```

Arguments

i	Node ids
j	Node ids. If left blank, will select *all* edges with i
model	LatClass model

Examples

```
data("email_data")
mod = makeLatentModel(email_data$edgeList, 10,
                      meta = email_data$meta)
mod$fit(fast_em = TRUE)

# Checking channel usage for
# first few edges
nodes_1 = email_data$edgeList[1:5, 1]
nodes_2 = email_data$edgeList[1:5, 2]
chan_connect(nodes_1, nodes_2, mod)

# Checking channel usage for all edges
# for first two nodes
chan_connect(i = c(1000, 1001), model = mod)
```

email_data *Email data for EU Univeristy*

Description

An email network with metadata for an EU university. Nodes are professors, edges indicate an email was sent between the two nodes. Metadata is the department that each node belong to.

Usage

```
email_data
```

Format

List with two objects:

edgeList A 16706 x 2 matrix of edges

meta A 1005 x 1 data frame indicating department of each node

est_auc *Estimate Out-of-Sample AUC*

Description

Estimate Out-of-Sample AUC

Usage

```
est_auc(
  edgeList,
  models = c("LCN", "BKN"),
  nChan = 10,
  nEdgesMasked = 400,
  nNonEdgesMasked = 400
)
```

Arguments

edgeList	nx2 matrix of edges
models	Character vector of models to use
nChan	Number of channels to use
nEdgesMasked	Number of edges to mask
nNonEdgesMasked	Number of non-edges to mask

makeLatentModel *Make Latent Structure model*

Description

Make a latent class model. Can be used for predicting unknown edge status and unknown metadata.

Usage

```
makeLatentModel(
  edgeList,
  nChans,
  model = "LCN",
  missingList = NULL,
  metadata = NULL
)
```

Arguments

edgeList	An matrix edgelist. Can be nx2 (both) or nx3 (BKN only)
nChans	Number of latent dimensions to use
model	Type of model to fit. Options are "LCN" or "BKN"
missingList	A nx2 matrix edgelist of edges for which the value is unknown
metadata	A data.frame with all factors representing metadata

Details

Fits either a Latent Channels Network (LCN), or the symmetric low-rank Poisson model of Ball, Karrer and Newman (BKN). The model assumes an undirected graph.

If edges are counts, use the BKN model. The data format for each row is (i,j, count), with i,j as integer IDs starting at 1.

If edges are binary, either a BKN or LCN model may be used, although an LCN model is somewhat more appropriate.

LCN model:

Clifford Anderson-Bergman, Phan Nguyen, and Jose Cadena Pico. "Latent Channel Networks", submitted 2019

BKN model:

Brian Ball, Brian Karrer, and Mark EJ Newman. "Efficient and principled method for detecting communities in networks." Physical Review E 84.3 (2011): 036103.

Examples

```
data(email_data)
# Building model with metadata
model = makeLatentModel(email_data$edgeList,
                        10,
                        metadata = email_data$meta)
# Fitting model
model$fit()
```

```
# Predicting two edge probabilities
predict(model, i = c(2,3), j = c(4,5))
```

`plot_net` *Build heatmap from model*

Description

Build heatmap from model

Usage

```
plot_net(
  mod,
  grp = NULL,
  metanames = NULL,
  minGrpSize = NULL,
  row_subset = NULL,
  col_subset = NULL,
  name = "",
  plotratio = 2,
  ...
)
```

Arguments

`mod`	LatClass object
`grp`	Vector of group categories for each node
`metanames`	Names of metavariables to plot
`minGrpSize`	Minimum size of group in both. Smaller groups put in "other"
`row_subset`	Subset of nodes to plot
`col_subset`	Subset of channels to plot
`name`	Legend names for plot
`plotratio`	If node parameters + meta parameters plotted, ratio between plots
`...`	Additional arguments passed to ComplexHeatmap::Heatmap

predict.LatClass *Predictions from LatClass objects*

Description

Predict edge probabilities and categorical metadata

Usage

```
## S3 method for class 'LatClass'
predict(object, i, j, type = "pairs", ...)
```

Arguments

object	LatClass model
i	node index
j	Either an node index or metadata colname name
type	Should node pairs ('pairs') or cross ('cross') of all combinations be predicted
...	Additional arguments. Ignored.

Examples

```
data(email_data)

# Building model and fitting
mod = makeLatentModel(email_data$edgeList,
                      nChans = 10,
                      metadata = email_data$meta)
mod$fit(fast_em = TRUE)

# Predicting edge pairs
predict(mod, i = 1:3, j = 4:2)

# Predicting all combinations of i and j
predict(mod, i = 1:3, j = 1:3, type = "cross")

# Predicting metadata
# Subsetting for brevity
predict(mod, i = 1:3, "dpt")[,1:5]
```

predicts_meta *Subsets channels that are predictive of metadata*

Description

Returns both a vector of channels that are predictive of at least one of the metadata values of interest and a parameter matrix of channels by metadata.

Usage

```
predicts_meta(
  model,
  metanames = NULL,
  metavars = NULL,
  threshold = 0.5,
  sumFun = max
)
```

Arguments

model	LatClass model
metanames	Vector of names of metadata values to predict
metavars	Vector of column names of metadata to predict
threshold	Minimal parameter value to be considered predictive
sumFun	Summary function: suggest either max or sum

Details

metanames refers to the individual values we might want to predict, while metavars is the column names.

Examples

```
data(email_data)
mod = makeLatentModel(email_data$edgeList, 20,
                      meta = email_data$meta)
mod$fit(fast_em = TRUE)

# Returns channels that are predictive
# of dpt == 1 or 2
predicts_meta(mod, metanames = c("dpt1", "dpt2") )
# Returns channels that are predictive
# of *any* dpt
predicts_meta(mod, metanames = NULL, metavars = "dpt")
```

---

simLCN *Simulate Latent Channel Network*

Description

Simulate Latent Channel Network

Usage

```
simLCN(p_mat)
```

Arguments

p_mat	Matrix of channel usage probabilities

Index