Package ‘networkscaleup’

April 6, 2022

Title  Network Scale-Up Models for Aggregated Relational Data

Version  0.1-1


License  GPL (>= 3)

Encoding  UTF-8

RoxygenNote  7.1.2

Biarch  true

Depends  R (>= 3.4.0)

Imports  methods, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), rstantools (>= 2.1.1), LaplacesDemon (>= 16.1.6)

LinkingTo  BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.18.1), StanHeaders (>= 2.18.0)

SystemRequirements  GNU make

Suggests  rmarkdown, knitr

VignetteBuilder  knitr

LazyData  true

NeedsCompilation  yes

Author  Ian Laga [aut, cre] (<https://orcid.org/0000-0002-5164-4856>), Le Bao [aut], Xiaoyue Niu [aut]

Maintainer  Ian Laga <ilaga25@gmail.com>

Repository  CRAN

Date/Publication  2022-04-06 07:10:02 UTC
**R topics documented:**

- networkscaleup-package ........................................ 2
- correlatedStan .................................................. 2
- example_data .................................................... 6
- killworth ............................................................ 7
- overdispersed ..................................................... 9
- overdispersedStan ............................................... 12
- scaling ............................................................. 14

**Index**

networkscaleup-package

*The 'networkscaleup' package.*

---

**Description**

Provides a variety of Network Scale-up Models for researchers to analyze Aggregated Relational Data, mostly through the use of Stan.

**References**


**correlatedStan**

*Fit ARD using the uncorrelated or correlated model in Stan* This function fits the ARD using either the uncorrelated or correlated model in Laga et al. (2021) in Stan. The population size estimates and degrees are scaled using a post-hoc procedure.

---

**Description**

Fit ARD using the uncorrelated or correlated model in Stan This function fits the ARD using either the uncorrelated or correlated model in Laga et al. (2021) in Stan. The population size estimates and degrees are scaled using a post-hoc procedure.
correlatedStan

Usage

correlatedStan(
   ard,
   known_sizes = NULL,
   known_ind = NULL,
   N = NULL,
   model = c("correlated", "uncorrelated"),
   scaling = c("all", "overdispersed", "weighted", "weighted_sq"),
   x = NULL,
   z_global = NULL,
   z_subpop = NULL,
   G1_ind = NULL,
   G2_ind = NULL,
   B2_ind = NULL,
   chains = 3,
   cores = 1,
   warmup = 1000,
   iter = 1500,
   thin = 1,
   return_fit = FALSE,
   ...
)

Arguments

ard The n_i x n_k matrix of non-negative ARD integer responses, where the (i,k)th element corresponds to the number of people that respondent i knows in subpopulation k.

known_sizes The known subpopulation sizes corresponding to a subset of the columns of ard.

known_ind The indices that correspond to the columns of ard with known_sizes. By default, the function assumes the first n_known columns, where n_known corresponds to the number of known_sizes.

N The known total population size.

model A character vector denoting which of the two models should be fit, either 'uncorrelated' or 'correlated'. More details of these models are provided below. The function decides which covariate model is needed based on the covariates provided below.

scaling An optional character vector providing the name of scaling procedure should be performed in order to transform estimates to degrees and subpopulation sizes. If NULL, the parameters will be returned unscaled. Alternatively, scaling may be performed independently using the scaling function. Scaling options are NULL, overdispersed, all, weighted, or weighted_sq (weighted and weighted_sq are only available if model = "correlated". Further details are provided in the Details section.

x A matrix with dimensions n_i x n_unknown, where n_unknown refers to the number of unknown subpopulation sizes. In the language of Teo et al. (2019), these represent the individual’s perception of each hidden population.
correlatedStan

z_global A matrix with dimensions n_i x p_global, where p_global is the number of demographic covariates used. This matrix represents the demographic information about the respondents in order to capture the barrier effects.

z_subpop A matrix with dimensions n_i x p_subpop, where p_subpop is the number of demographic covariates used. This matrix represents the demographic information about the respondents in order to capture the barrier effects.

G1_ind A vector of indices denoting the columns of ard that correspond to the primary scaling groups, i.e. the collection of rare girls’ names in Zheng, Salganik, and Gelman (2006). By default, all known_sizes are used. If G2_ind and B2_ind are not provided, C = C_1, so only G1_ind are used. If G1_ind is not provided, no scaling is performed.

G2_ind A vector of indices denoting the columns of ard that correspond to the subpopulations that belong to the first secondary scaling groups, i.e. the collection of somewhat popular girls’ names.

B2_ind A vector of indices denoting the columns of ard that correspond to the subpopulations that belong to the second secondary scaling groups, i.e. the collection of somewhat popular boys’ names.

chains A positive integer specifying the number of Markov chains.

cores A positive integer specifying the number of cores to use to run the Markov chains in parallel.

warmup A positive integer specifying the total number of samples for each chain (including warmup). Matches the usage in stan.

iter A positive integer specifying the number of warmup samples for each chain. Matches the usage in stan.

thin A positive integer specifying the interval for saving posterior samples. Default value is 1 (i.e. no thinning).

return_fit A logical indicating whether the fitted stanfit object should be return. Defaults to FALSE.

... Additional arguments to be passed to stan.

Details

This function currently fits a variety of models proposed in Laga et al. (2022+). The user may provide any combination of x, z_global, and z_subpop. Additionally, the user may choose to fit a uncorrelated version of the model, where the correlation matrix is equal to the identity matrix.

The scaling options are described below:

NULL No scaling is performed

overdispersed The scaling procedure outlined in Zheng et al. (2006) is performed. In this case, at least Pg1_ind must be provided. See overdispersedStan for more details.

all All subpopulations with known sizes are used to scale the parameters, using a modified scaling procedure that standardizes the sizes so each population is weighted equally. Additional details are provided in Laga et al. (2022+).

weighted All subpopulations with known sizes are weighted according their correlation with the unknown subpopulation size. Additional details are provided in Laga et al. (2022+).
**weighted_sq**  Same as weighted, except the weights are squared, providing more relative weight to subpopulations with higher correlation.

**Value**

Either the full fitted Stan model if `return_fit = TRUE`, else a named list with the estimated parameters extracted using `extract` (the default). The estimated parameters are named as follows (if estimated in the corresponding model), with additional descriptions as needed:

- **delta**  Raw delta parameters
- **sigma_delta**  Standard deviation of delta
- **rho**  Log prevalence, if scaled, else raw rho parameters
- **mu_rho**  Mean of rho
- **sigma_rho**  Standard deviation of rho
- **alpha**  Slope parameters corresponding to `z`
- **beta_global**  Slope parameters corresponding to `x_global`
- **beta_subpop**  Slope parameters corresponding to `x_subpop`
- **tau_N**  Standard deviation of random effects `b`
- **Corr**  Correlation matrix, if `Correlation = TRUE`

If scaled, the following additional parameters are included:

- **log_degrees**  Scaled log degrees
- **degree**  Scaled degrees
- **log_prevalences**  Scaled log prevalences
- **sizes**  Subpopulation size estimates

**References**


**Examples**

```r
## Not run:
data(example_data)

x = example_data$x
z_global = example_data$z[,1:2]
z_subpop = example_data$z[,3:4]

basic_corr_est = correlatedStan(example_data$ard,
    known_sizes = example_data$subpop_sizes[c(1, 2, 4)],
    known_ind = c(1, 2, 4),
    N = example_data$N,
    model = "correlated",
    scaling = "weighted",
```
example_data = Simulated ARD data set with z and x.

```r
ch = 1,
c = 1,
w = 50,
iter = 100

c = correlateStan(example_data$ard,
known_sizes = example_data$subpop_sizes[c(1, 2, 4)],
known_ind = c(1, 2, 4),
N = example_data$N,
model = "uncorrelated",
scaling = "all",
x = x,
z_global = z_global,
z_subpop = z_subpop,
chains = 1,
cores = 1,
warmup = 50,
iter = 100)

c = correlateStan(example_data$ard,
known_sizes = example_data$subpop_sizes[c(1, 2, 4)],
known_ind = c(1, 2, 4),
N = example_data$N,
model = "correlated",
scaling = "all",
x = x,
z_subpop = z_subpop,
chains = 1,
cores = 1,
warmup = 50,
iter = 100)

# Compare size estimates
round(data.frame(true = example_data$subpop_sizes,
corr_basic = colMeans(basic_corr_est$sizes),
uncorr_x_zsubpop_zglobal = colMeans(cov_uncorr_est$sizes),
corr_x_zsubpop = colMeans(cov_corr_est$sizes)))

# Look at z slope parameters
colMeans(cov_uncorr_est$beta_global)
colMeans(cov_corr_est$beta_subpop)
colMeans(cov_uncorr_est$beta_subpop)

# Look at x slope parameters
colMeans(cov_uncorr_est$alpha)
colMeans(cov_corr_est$alpha)

## End(Not run)
```
Description

A simulated data set to demonstrate and test the NSUM methods. The data was simulated from the basic Killworth Binomial model.

Usage

example_data

Format

A named list for an ARD survey from 100 respondents about 5 subpopulations.

ard  A 100 x 5 matrix with integer valued respondents
x    A 100 x 5 matrix with simulated answers from a 1-5 Likert scale
z    A 100 x 4 matrix with answers for each respondents about 4 demographic questions
N    An integer specifying the total population size
subpop_size A vector with the 5 true subpopulation sizes
degrees A vector with the 100 true respondent degrees

killworth

Fit Killworth models to ARD. This function estimates the degrees and population sizes using the plug-in MLE and MLE estimator.

Usage

killworth(
  ard,
  known_sizes = NULL,
  known_ind = 1:length(known_sizes),
  N = NULL,
  model = c("MLE", "PIMLE")
)

Arguments

ard  The n_i x n_k matrix of non-negative ARD integer responses, where the (i,k)th element corresponds to the number of people that respondent i knows in subpopulation k.
known_sizes  The known subpopulation sizes corresponding to a subset of the columns of ard.
known_ind: The indices that correspond to the columns of ard with known_sizes. By default, the function assumes the first n_known columns, where n_known corresponds to the number of known_sizes.

N: The known total population size.

model: A character string corresponding to either the plug-in MLE (PIMLE) or the MLE (MLE). The function assumes MLE by default.

Value

A named list with the estimated degrees and sizes.

References


Examples

# Analyze an example ard data set using the killworth function
data(example_data)

ard = example_data$ard
subpop_sizes = example_data$subpop_sizes
N = example_data$N

mle.est = killworth(ard,
known_sizes = subpop_sizes[c(1, 2, 4)],
known_ind = c(1, 2, 4),
N = N, model = "MLE")

pimle.est = killworth(ard,
known_sizes = subpop_sizes[c(1, 2, 4)],
known_ind = c(1, 2, 4),
N = N, model = "PIMLE")

## Compare estimates with the truth
plot(mle.est$degrees, example_data$degrees)

data.frame(true = subpop_sizes[c(3, 5)],
mle = mle.est$sizes,
pimle = pimle.est$sizes)
overdispersed

Fit Overdispersed model to ARD (Gibbs-Metropolis)

Description

This function fits the ARD using the Overdispersed model using the original Gibbs-Metropolis Algorithm provided in Zheng, Salganik, and Gelman (2006). The population size estimates and degrees are scaled using a post-hoc procedure. For the Stan implementation, see overdispersedStan.

Usage

overdispersed(
  ard,
  known_sizes = NULL,
  known_ind = NULL,
  G1_ind = NULL,
  G2_ind = NULL,
  B2_ind = NULL,
  N = NULL,
  warmup = 1000,
  iter = 1500,
  refresh = NULL,
  thin = 1,
  verbose = FALSE,
  alpha_tune = 0.4,
  beta_tune = 0.2,
  omega_tune = 0.2,
  init = "MLE"
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ard</td>
<td>The n_i x n_k matrix of non-negative ARD integer responses, where the (i,k)th element corresponds to the number of people that respondent i knows in sub-population k.</td>
</tr>
<tr>
<td>known_sizes</td>
<td>The known subpopulation sizes corresponding to a subset of the columns of ard.</td>
</tr>
<tr>
<td>known_ind</td>
<td>The indices that correspond to the columns of ard with known_sizes. By default, the function assumes the first n_known columns, where n_known corresponds to the number of known_sizes.</td>
</tr>
<tr>
<td>G1_ind</td>
<td>A vector of indices denoting the columns of ard that correspond to the primary scaling groups, i.e. the collection of rare girls' names in Zheng, Salganik, and Gelman (2006). By default, all known_sizes are used. If G2_ind and B2_ind are not provided, C = C_1, so only G1_ind are used. If G1_ind is not provided, no scaling is performed.</td>
</tr>
</tbody>
</table>
G2_ind  A vector of indices denoting the columns of ard that correspond to the subpopulations that belong to the first secondary scaling groups, i.e. the collection of somewhat popular girls’ names.

B2_ind  A vector of indices denoting the columns of ard that correspond to the subpopulations that belong to the second secondary scaling groups, i.e. the collection of somewhat popular boys’ names.

N  The known total population size.

warmup  A positive integer specifying the number of warmup samples.

iter  A positive integer specifying the total number of samples (including warmup).

refresh  An integer specifying how often the progress of the sampling should be reported. By default, resorts to every 10%. Suppressed if verbose = FALSE.

thin  A positive integer specifying the interval for saving posterior samples. Default value is 1 (i.e. no thinning).

verbose  Logical value, specifying whether sampling progress should be reported.

alpha_tune  A positive numeric indicating the standard deviation used as the jumping scale in the Metropolis step for alpha. Defaults to 0.4, which has worked well for other ARD datasets.

beta_tune  A positive numeric indicating the standard deviation used as the jumping scale in the Metropolis step for beta. Defaults to 0.2, which has worked well for other ARD datasets.

omega_tune  A positive numeric indicating the standard deviation used as the jumping scale in the Metropolis step for omega. Defaults to 0.2, which has worked well for other ARD datasets.

init  A named list with names corresponding to the first-level model parameters, name ‘alpha’, ‘beta’, and ‘omega’. By default the ‘alpha’ and ‘beta’ parameters are initialized at the values corresponding to the Killworth MLE estimates (for the missing ‘beta’), with all ‘omega’ set to 20. Alternatively, init = ‘random’ simulates ‘alpha’ and ‘beta’ from a normal random variable with mean 0 and standard deviation 1. By default, init = ‘MLE’ initializes values at the Killworth et al. (1998b) MLE estimates for the degrees and sizes and simulates the other parameters.

Details

This function fits the overdispersed NSUM model using the Metropolis-Gibbs sampler provided in Zheng et al. (2006).

Value

A named list with the estimated posterior samples. The estimated parameters are named as follows, with additional descriptions as needed:

alphas  Log degree, if scaled, else raw alpha parameters

betas  Log prevalence, if scaled, else raw beta parameters

inv_omegas  Inverse of overdispersion parameters
**sigma_alpha** Standard deviation of alphas

**mu_beta** Mean of betas

**sigma_beta** Standard deviation of betas

**omegas** Overdispersion parameters

If scaled, the following additional parameters are included:

**mu_alpha** Mean of log degrees

**degrees** Degree estimates

**sizes** Subpopulation size estimates

**References**


**Examples**

```r
# Analyze an example ard data set using Zheng et al. (2006) models
# Note that in practice, both warmup and iter should be much higher
data(example_data)

ard = example_data$ard
subpop_sizes = example_data$subpop_sizes
known_ind = c(1, 2, 4)
N = example_data$N

overdisp.est = overdispersed(ard,
known_sizes = subpop_sizes[known_ind],
known_ind = known_ind,
G1_ind = 1,
G2_ind = 2,
B2_ind = 4,
N = N,
warmup = 50,
iter = 100)

# Compare size estimates
data.frame(true = subpop_sizes,
basic = colMeans(overdisp.est$sizes))

# Compare degree estimates
plot(example_data$degrees, colMeans(overdisp.est$degrees))

# Look at overdispersion parameter
colMeans(overdisp.est$omegas)
```
overdispersedStan  

Fit ARD using the Overdispersed model in Stan

Description

This function fits the ARD using the Overdispersed model in Stan. The population size estimates and degrees are scaled using a post-hoc procedure. For the Gibbs-Metropolis algorithm implementation, see overdispersed.

Usage

overdispersedStan(
  ard,
  known_sizes = NULL,
  known_ind = NULL,
  G1_ind = NULL,
  G2_ind = NULL,
  B2_ind = NULL,
  N = NULL,
  chains = 3,
  cores = 1,
  warmup = 1000,
  iter = 1500,
  thin = 1,
  return_fit = FALSE,
  ...
)

Arguments

- **ard**: The n_i x n_k matrix of non-negative ARD integer responses, where the (i,k)th element corresponds to the number of people that respondent i knows in subpopulation k.
- **known_sizes**: The known subpopulation sizes corresponding to a subset of the columns of ard.
- **known_ind**: The indices that correspond to the columns of ard with known_sizes. By default, the function assumes the first n_known columns, where n_known corresponds to the number of known_sizes.
- **G1_ind**: A vector of indices denoting the columns of ard that correspond to the primary scaling groups, i.e. the collection of rare girls’ names in Zheng, Salganik, and Gelman (2006). By default, all known_sizes are used. If G2_ind and B2_ind are not provided, C = C_1, so only G1_ind are used. If G1_ind is not provided, no scaling is performed.
- **G2_ind**: A vector of indices denoting the columns of ard that correspond to the subpopulations that belong to the first secondary scaling groups, i.e. the collection of somewhat popular girls’ names.
B2_ind
A vector of indices denoting the columns of ard that correspond to the subpopulations that belong to the second secondary scaling groups, i.e. the collection of somewhat popular boys’ names.

N
The known total population size.

chains
A positive integer specifying the number of Markov chains.

cores
A positive integer specifying the number of cores to use to run the Markov chains in parallel.

warmup
A positive integer specifying the total number of samples for each chain (including warmup). Matches the usage in stan.

iter
A positive integer specifying the number of warmup samples for each chain. Matches the usage in stan.

thin
A positive integer specifying the interval for saving posterior samples. Default value is 1 (i.e. no thinning).

return_fit
A logical indicating whether the fitted Stan model should be returned instead of the rstan::extracted and scaled parameters. This is FALSE by default.

Details
This function fits the overdispersed NSUM model using the Gibbs-Metropolis algorithm provided in Zheng et al. (2006).

Value
Either the full fitted Stan model if return_fit = TRUE, else a named list with the estimated parameters extracted using extract (the default). The estimated parameters are named as follows, with additional descriptions as needed:

alphas
Log degree, if scaling = TRUE, else raw alpha parameters

betas
Log prevalence, if scaling = TRUE, else raw beta parameters

inv_omegas
Inverse of overdispersion parameters

sigma_alpha
Standard deviation of alphas

mu_beta
Mean of betas

sigma_beta
Standard deviation of betas

omegas
Overdispersion parameters

If scaling = TRUE, the following additional parameters are included:

mu_alpha
Mean of log degrees

degrees
Degree estimates

sizes
Subpopulation size estimates

References
Examples

# Analyze an example ard data set using Zheng et al. (2006) models
# Note that in practice, both warmup and iter should be much higher
## Not run:
data(example_data)

ard = example_data$ard
subpop_sizes = example_data$subpop_sizes
known_ind = c(1, 2, 4)
N = example_data$N

overdisp.est = overdispersedStan(ard,
known_sizes = subpop_sizes[known_ind],
known_ind = known_ind,
G1_ind = 1,
G2_ind = 2,
B2_ind = 4,
N = N,
chains = 1,
cores = 1,
warmup = 250,
iter = 500)

# Compare size estimates
round(data.frame(true = subpop_sizes,
basic = colMeans(overdisp.est$sizes)))

# Compare degree estimates
plot(example_data$degrees, colMeans(overdisp.est$degrees))

# Look at overdispersion parameter
colMeans(overdisp.est$omegas)

## End(Not run)

---

scaling

Scale raw log degree and log prevalence estimates

Description

This function scales estimates from either the overdispersed model or from the correlated models. Several scaling options are available.

Usage

scaling(
log_degrees,
log_prevalences,
scaling = c("all", "overdispersed", "weighted", "weighted_sq"),
known_sizes = NULL,
known_ind = NULL,
Correlation = NULL,
G1_ind = NULL,
G2_ind = NULL,
B2_ind = NULL,
N = NULL
)

Arguments

log_degrees The matrix of estimated raw log degrees from either the overdispersed or correlated models.

log_prevalences The matrix of estimates raw log prevalences from either the overdispersed or correlated models.

scaling An character vector providing the name of scaling procedure should be performed in order to transform estimates to degrees and subpopulation sizes. Scaling options are overdispersed, all (the default), weighted, or weighted_sq (weighted and weighted_sq are only available if Correlation is provided. Further details are provided in the Details section.

known_sizes The known subpopulation sizes corresponding to a subset of the columns of ard.

known_ind The indices that correspond to the columns of ard with known_sizes. By default, the function assumes the first n_known columns, where n_known corresponds to the number of known_sizes.

Correlation The estimated correlation matrix used to calculate scaling weights. Required if scaling = weighted or scaling = weighted_sq.

G1_ind If scaling = overdispersed, a vector of indices corresponding to the subpopulations that belong to the primary scaling groups, i.e. the collection of rare girls’ names in Zheng, Salganik, and Gelman (2006). By default, all known_sizes are used. If G2_ind and B2_ind are not provided, C = C_1, so only G1_ind are used. If G1_ind is not provided, no scaling is performed.

G2_ind If scaling = overdispersed, a vector of indices corresponding to the subpopulations that belong to the first secondary scaling groups, i.e. the collection of somewhat popular girls’ names.

B2_ind If scaling = overdispersed, a vector of indices corresponding to the subpopulations that belong to the second secondary scaling groups, i.e. the collection of somewhat popular boys’ names.

N The known total population size.

Details

The scaling options are described below:

NULL No scaling is performed

overdispersed The scaling procedure outlined in Zheng et al. (2006) is performed. In this case, at least Pg1_ind must be provided. See overdispersedStan for more details.
All subpopulations with known sizes are used to scale the parameters, using a modified scaling procedure that standardizes the sizes so each population is weighted equally. Additional details are provided in Laga et al. (2021).

**weighted** All subpopulations with known sizes are weighted according their correlation with the unknown subpopulation size. Additional details are provided in Laga et al. (2021)

**weighted_sq** Same as weighted, except the weights are squared, providing more relative weight to subpopulations with higher correlation.

**Value**

The named list containing the scaled log degree, degree, log prevalence, and size estimates

**References**


Index

* datasets
  example_data, 6

  correlatedStan, 2

  example_data, 6
  extract, 5, 13

  killworth, 7

  networkscaleup
    (networkscaleup-package), 2

  networkscaleup-package, 2

  overdispersed, 9, 12

  overdispersedStan, 4, 9, 12, 15

  scaling, 3, 14

  stan, 4, 13