

Package ‘networkscaleup’

June 26, 2026

Title Network Scale-Up Models for Aggregated Relational Data

Version 0.2-2

Description Provides a variety of Network Scale-up Models for researchers to analyze Aggregated Relational Data, through the use of Stan and 'glmmTMB'. Also provides tools for model checking

In this version, the package implements models from

Laga, I., Bao, L., and Niu, X (2023) <[doi:10.1080/01621459.2023.2165929](https://doi.org/10.1080/01621459.2023.2165929)>, Zheng, T., Salganik, M. J., and Gelman, A. (2006) <[doi:10.1198/016214505000001168](https://doi.org/10.1198/016214505000001168)>, Killworth, P. D., Johnsen, E. C., McCarty, C., Shelley, G. A., and Bernard, H. R. (1998) <[doi:10.1016/S0378-8733\(96\)00305-X](https://doi.org/10.1016/S0378-8733(96)00305-X)>, and Killworth, P. D., McCarty, C., Bernard, H. R., Shelley, G. A., and Johnsen, E. C. (1998) <[doi:10.1177/0193841X9802200205](https://doi.org/10.1177/0193841X9802200205)>.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Biarch true

Depends R (>= 4.1.0)

Imports methods, Rcpp (>= 0.12.0), rstan (>= 2.26.0), LaplacesDemon (>= 16.1.6), dplyr, ggplot2, scales, stats, readr, tibble, tidyr, rlang, glmmTMB, gridExtra, purrr, stringr, trialr, tidyselect, RMTstat

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), rstan (>= 2.26.0), StanHeaders (>= 2.26.0)

SystemRequirements GNU make

Suggests rmarkdown, knitr

VignetteBuilder knitr

LazyData true

NeedsCompilation yes

Author Ian Laga [aut] (ORCID: <<https://orcid.org/0000-0002-5164-4856>>), Owen G. Ward [aut, cre], Anna L. Smith [aut],

Benjamin Vogel [aut],
Jieyun Wang [aut],
Le Bao [aut],
Xiaoyue Niu [aut]

Maintainer Owen G. Ward <oward@sfu.ca>

URL <https://github.com/ilaga/networkscaleup>

Repository CRAN

Date/Publication 2026-06-26 08:00:02 UTC

Contents

construct_pearson	2
construct_rqr	3
correlatedStan	3
cov_plots	8
dispersion_metric	8
example_data	9
fit_mle	9
get_surrogate	10
hang_rootogram_ard	10
killworth	11
log_mix_uniform	12
make_ard	13
make_ard_tidy	14
networkscaleup	15
overdispersed	15
overdispersedStan	18
plot_fitted	21
plot_tw_statistic	21
residual_correlation	22
residual_heatmap	23
rqr_nbinom_logs	23
rqr_pois_logs	24
scaling	24
tw_group_corr_test	26

Index	27
--------------	-----------

construct_pearson	<i>Compute Pearson Residuals for ARD matrix and fitted model</i>
-------------------	--

Description

Compute Pearson Residuals for ARD matrix and fitted model

Usage

```
construct_pearson(ard, model_fit)
```

Arguments

ard	ARD matrix y
model_fit	estimated model

Value

a vector (column by column) of corresponding residuals from ARD matrix

construct_rqr	<i>Compute Randomized Quantile Residuals for ARD Models</i>
---------------	---

Description

Compute Randomized Quantile Residuals for ARD Models

Usage

```
construct_rqr(ard, model_fit)
```

Arguments

ard	ard matrix
model_fit	fitted model, along with required details

Value

a vector of residuals (column by column)

correlatedStan	<i>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.</i>
----------------	--

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.

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

<code>ard</code>	The ‘ $n_i \times n_k$ ’ 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’.
<code>known_sizes</code>	The known subpopulation sizes corresponding to a subset of the columns of <code>ard</code> .
<code>known_ind</code>	The indices that correspond to the columns of <code>ard</code> with <code>known_sizes</code> . By default, the function assumes the first <code>n_known</code> columns, where <code>n_known</code> corresponds to the number of <code>known_sizes</code> .
<code>N</code>	The known total population size.
<code>model</code>	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.
<code>scaling</code>	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.
<code>x</code>	A matrix with dimensions ‘ $n_i \times n_{\text{unknown}}$ ’, 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.

<code>z_global</code>	A matrix with dimensions ‘ <code>n_i x p_global</code> ’, where ‘ <code>p_global</code> ’ is the number of demographic covariates used. This matrix represents the demographic information about the respondents in order to capture the barrier effects.
<code>z_subpop</code>	A matrix with dimensions ‘ <code>n_i x p_subpop</code> ’, where ‘ <code>p_subpop</code> ’ is the number of demographic covariates used. This matrix represents the demographic information about the respondents in order to capture the barrier effects.
<code>G1_ind</code>	A vector of indices denoting the columns of ‘ <code>ard</code> ’ 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 <code>G2_ind</code> and <code>B2_ind</code> are not provided, ‘ <code>C = C_1</code> ’, so only <code>G1_ind</code> are used. If <code>G1_ind</code> is not provided, no scaling is performed.
<code>G2_ind</code>	A vector of indices denoting the columns of ‘ <code>ard</code> ’ that correspond to the subpopulations that belong to the first secondary scaling groups, i.e. the collection of somewhat popular girls’ names.
<code>B2_ind</code>	A vector of indices denoting the columns of ‘ <code>ard</code> ’ that correspond to the subpopulations that belong to the second secondary scaling groups, i.e. the collection of somewhat popular boys’ names.
<code>chains</code>	A positive integer specifying the number of Markov chains.
<code>cores</code>	A positive integer specifying the number of cores to use to run the Markov chains in parallel.
<code>warmup</code>	A positive integer specifying the total number of samples for each chain (including warmup). Matches the usage in stan .
<code>iter</code>	A positive integer specifying the number of warmup samples for each chain. Matches the usage in stan .
<code>thin</code>	A positive integer specifying the interval for saving posterior samples. Default value is 1 (i.e. no thinning).
<code>return_fit</code>	A logical indicating whether the fitted ‘ <code>stanfit</code> ’ object should be return. Defaults to ‘ <code>FALSE</code> ’.
<code>...</code>	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 an 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

Laga, I., Bao, L., and Niu, X (2021). A Correlated Network Scaleup Model: Finding the Connection Between Subpopulations

Examples

```
## 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",
```

```

    chains = 1,
    cores = 1,
    warmup = 50,
    iter = 100
  )

cov_uncorr_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 = "uncorrelated",
  scaling = "all",
  x = x,
  z_global = z_global,
  z_subpop = z_subpop,
  chains = 1,
  cores = 1,
  warmup = 50,
  iter = 100
)

cov_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 = "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)

```

cov_plots	<i>Covariance plots</i>
-----------	-------------------------

Description

Plots of the estimated covariance structure from a given fitted model

Usage

```
cov_plots(  
  ard,  
  model_fit,  
  x_cov,  
  resid_type = c("rqr", "pearson_residuals"),  
  method = "lm",  
  se = F  
)
```

Arguments

- ard ard matrix
- model_fit a fitted object from [fit_mle()] or [fit_map()]
- x_cov covariate matrix
- resid_type the type of residuals to use
- method the method to use
- se whether to compute standard errors of estimates

Value

a list of ggplots, corresponding to covariance structure

dispersion_metric	<i>Dispersion Metric for Fitted ARD Model</i>
-------------------	---

Description

Dispersion Metric for Fitted ARD Model

Usage

```
dispersion_metric(ard, model_fit)
```


Arguments

<code>ard</code>	ard matrix
<code>model_fit</code>	list of fitted model and details

Value

a ggplot of the hanging rootogram

<code>example_data</code>	<i>Simulated ARD data set with z and x.</i>
---------------------------	---

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

<code>fit_mle</code>	<i>Fit basic Poisson and Negative Binomial models using glmmTMB</i>
----------------------	---

Description

Fit basic Poisson and Negative Binomial models using glmmTMB

Usage

```
fit_mle(
  ard,
  x_cov_global = NULL,
  x_cov_local = NULL,
  family = c("poisson", "nbinomial")
)
```

Arguments

- ard n_i by n_k ARD matrix
- x_cov_global n_i by p_global covariate matrix of global covariates
- x_cov_local n_i by p_local covariate matrix of local covariates
- family distribution to fit, either "poisson" or "nbinomial"

Value

list containing fitted model and extracted parameters

get_surrogate	<i>Compute Surrogate Residuals for ARD Models</i>
---------------	---

Description

Compute Surrogate Residuals for ARD Models

Usage

get_surrogate(ard, model_fit = NULL)

Arguments

- ard the ARD matrix
- model_fit list containing fitted model, details

Value

a vector of residuals (column by column)

hang_rootogram_ard	<i>Hanging Rootogram for Fitted ARD Model</i>
--------------------	---

Description

Hanging Rootogram for Fitted ARD Model

Usage

hang_rootogram_ard(ard, model_fit, width = 0.9, x_max = NULL, by_group = FALSE)

Arguments

<code>ard</code>	ard matrix
<code>model_fit</code>	fitted model object
<code>width</code>	width of bars
<code>x_max</code>	the maximum x value to display
<code>by_group</code>	logical; if TRUE, create separate rootograms for each column (group)

Value

a ggplot of the hanging rootogram (single plot if `by_group=FALSE`, combined plot if `by_group=TRUE`)

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

Description

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

<code>ard</code>	The ‘ $n_i \times n_k$ ’ 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’.
<code>known_sizes</code>	The known subpopulation sizes corresponding to a subset of the columns of <code>ard</code> .
<code>known_ind</code>	The indices that correspond to the columns of <code>ard</code> with <code>known_sizes</code> . By default, the function assumes the first <code>n_known</code> columns, where <code>n_known</code> corresponds to the number of <code>known_sizes</code> .
<code>N</code>	The known total population size.
<code>model</code>	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

Killworth, P. D., Johnsen, E. C., McCarty, C., Shelley, G. A., and Bernard, H. R. (1998). A Social Network Approach to Estimating Seroprevalence in the United States, *Social Networks*, **20**, 23–50

Killworth, P. D., McCarty, C., Bernard, H. R., Shelley, G. A., and Johnsen, E. C. (1998). Estimation of Seroprevalence, Rape and Homelessness in the United States Using a Social Network Approach, *Evaluation Review*, **22**, 289–308

Laga, I., Bao, L., and Niu, X. (2021). Thirty Years of the Network Scale-up Method, *Journal of the American Statistical Association*, **116:535**, 1548–1559

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
)
```

log_mix_uniform

log computed uniform quantile

Description

log computed uniform quantile

Usage

```
log_mix_uniform(logFl, logFu)
```

Arguments

```
logFl      log of lower value
logFu      log of upper value
```

Value

log value of uniform between Flower and Fupper

make_ard	<i>Generate simulated ARD</i>
----------	-------------------------------

Description

Generate simulated ARD

Usage

```
make_ard(
  n_i = 500,
  n_k = 20,
  N = 1e+06,
  p = 0,
  p_global_nonzero = 0,
  p_local_nonzero = 0,
  group_corr = FALSE,
  degree_corr = FALSE,
  family = c("poisson", "nbinomial"),
  omega_range = c(1, 5),
  alpha_mean = 5,
  alpha_sd = 0.15,
  eta = 3,
  seed = NULL
)
```

Arguments

```
n_i      number of respondents (rows)
n_k      number of groups (columns)
N        total population size
p        number of collected covariates
p_global_nonzero
          number of non-zero global covariates
```

p_local_nonzero	number of non-zero local covariates
group_corr	group correlation
degree_corr	degree correlation
family	sampling distribution
omega_range	minimum and maximum omega for negative binomial overdispersion
alpha_mean	mean of alphas
alpha_sd	variance of alphas
eta	correlation hyperparameter for LKJ prior
seed	random seed

Value

simulated ARD along with all true parameters. Parameters which are not used in a specific setting are set to NULL.

Examples

```
make_ard(N = 10000, family = "poisson")
```

make_ard_tidy	<i>Construct tibble from ARD matrix</i>
---------------	---

Description

Construct tibble from ARD matrix

Usage

```
make_ard_tidy(ard)
```

Arguments

ard the ARD matrix

Value

a tibble of ARD, with columns for row/col index

networkscaleup	<i>The 'networkscaleup' package.</i>
----------------	--------------------------------------

Description

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

Author(s)

Maintainer: Owen G. Ward <oward@sfu.ca>

Authors:

- Ian Laga <ilaga25@gmail.com> ([ORCID](#))
- Anna L. Smith
- Benjamin Vogel
- Jieyun Wang
- Le Bao <lebao@psu.edu>
- Xiaoyue Niu <Xiaoyue@psu.edu>

See Also

Useful links:

- <https://github.com/ilaga/networkscaleup>

overdispersed	<i>Fit Overdispersed model to ARD (Gibbs-Metropolis)</i>
---------------	--

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

<code>ard</code>	The ' $n_i \times n_k$ ' 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 '.
<code>known_sizes</code>	The known subpopulation sizes corresponding to a subset of the columns of <code>ard</code> .
<code>known_ind</code>	The indices that correspond to the columns of <code>ard</code> with <code>known_sizes</code> . By default, the function assumes the first <code>n_known</code> columns, where <code>n_known</code> corresponds to the number of <code>known_sizes</code> .
<code>G1_ind</code>	A vector of indices denoting the columns of ' <code>ard</code> ' that correspond to the primary scaling groups, i.e. the collection of rare girls' names in Zheng, Salganik, and Gelman (2006). By default, all <code>known_sizes</code> are used. If <code>G2_ind</code> and <code>B2_ind</code> are not provided, ' $C = C_1$ ', so only <code>G1_ind</code> are used. If <code>G1_ind</code> is not provided, no scaling is performed.
<code>G2_ind</code>	A vector of indices denoting the columns of ' <code>ard</code> ' that correspond to the subpopulations that belong to the first secondary scaling groups, i.e. the collection of somewhat popular girls' names.
<code>B2_ind</code>	A vector of indices denoting the columns of ' <code>ard</code> ' that correspond to the subpopulations that belong to the second secondary scaling groups, i.e. the collection of somewhat popular boys' names.
<code>N</code>	The known total population size.
<code>warmup</code>	A positive integer specifying the number of warmup samples.
<code>iter</code>	A positive integer specifying the total number of samples (including warmup).
<code>refresh</code>	An integer specifying how often the progress of the sampling should be reported. By default, resorts to every 10 <code>verbose = FALSE</code> .
<code>thin</code>	A positive integer specifying the interval for saving posterior samples. Default value is 1 (i.e. no thinning).
<code>verbose</code>	Logical value, specifying whether sampling progress should be reported.
<code>alpha_tune</code>	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.
<code>beta_tune</code>	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.

<code>omega_tune</code>	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.
<code>init</code>	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, <code>init = 'random'</code> simulates 'alpha' and 'beta' from a normal random variable with mean 0 and standard deviation 1. By default, <code>init = 'MLE'</code> 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

Zheng, T., Salganik, M. J., and Gelman, A. (2006). How many people do you know in prison, *Journal of the American Statistical Association*, **101:474**, 409–423

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
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

<code>ard</code>	The ‘ $n_i \times n_k$ ’ 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’.
<code>known_sizes</code>	The known subpopulation sizes corresponding to a subset of the columns of <code>ard</code> .
<code>known_ind</code>	The indices that correspond to the columns of <code>ard</code> with <code>known_sizes</code> . By default, the function assumes the first <code>n_known</code> columns, where <code>n_known</code> corresponds to the number of <code>known_sizes</code> .
<code>G1_ind</code>	A vector of indices denoting the columns of ‘ <code>ard</code> ’ that correspond to the primary scaling groups, i.e. the collection of rare girls’ names in Zheng, Salganik, and Gelman (2006). By default, all <code>known_sizes</code> are used. If <code>G2_ind</code> and <code>B2_ind</code> are not provided, ‘ <code>C = C_1</code> ’, so only <code>G1_ind</code> are used. If <code>G1_ind</code> is not provided, no scaling is performed.
<code>G2_ind</code>	A vector of indices denoting the columns of ‘ <code>ard</code> ’ that correspond to the subpopulations that belong to the first secondary scaling groups, i.e. the collection of somewhat popular girls’ names.
<code>B2_ind</code>	A vector of indices denoting the columns of ‘ <code>ard</code> ’ that correspond to the subpopulations that belong to the second secondary scaling groups, i.e. the collection of somewhat popular boys’ names.
<code>N</code>	The known total population size.
<code>chains</code>	A positive integer specifying the number of Markov chains.
<code>cores</code>	A positive integer specifying the number of cores to use to run the Markov chains in parallel.
<code>warmup</code>	A positive integer specifying the total number of samples for each chain (including warmup). Matches the usage in stan .
<code>iter</code>	A positive integer specifying the number of warmup samples for each chain. Matches the usage in stan .
<code>thin</code>	A positive integer specifying the interval for saving posterior samples. Default value is 1 (i.e. no thinning).
<code>return_fit</code>	A logical indicating whether the fitted Stan model should be returned instead of the <code>rstan::extracted</code> and scaled parameters. This is <code>FALSE</code> by default.
<code>...</code>	Additional arguments to be passed to stan .

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

Zheng, T., Salganik, M. J., and Gelman, A. (2006). How many people do you know in prison, *Journal of the American Statistical Association*, **101:474**, 409–423

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)

```

plot_fitted	<i>Plot residuals against fitted values</i>
-------------	---

Description

Plot residuals against fitted values

Usage

```
plot_fitted(ard, model_fit = NULL, resid = c("rqr", "pearson", "surrogate"))
```

Arguments

ard	ARD matrix (may be needed)
model_fit	fitted model
resid	the type of residuals to be used

Value

a ggplot showing fitted values against residuals

plot_tw_statistic	<i>Plot the TW Test Statistic</i>
-------------------	-----------------------------------

Description

Plot the TW Test Statistic

Usage

```
plot_tw_statistic(  
  tw_stat,  
  beta = 1,  
  base_x_limits = c(-5, 10),  
  base_size = 9,  
  title_size = 10,  
  axis_title_size = 8,  
  axis_text_size = 7,  
  label_size = 2.8  
)
```

Arguments

tw_stat	Tracey-Widom test statistic
beta	order param of TW distribution
base_x_limits	limits for axis
base_size	plot sizes
title_size	title size
axis_title_size	axis title size
axis_text_size	axis text size
label_size	label size

Value

a ggplot

residual_correlation	<i>Construction Residual (row/column) correlation matrix</i>
----------------------	--

Description

Construction Residual (row/column) correlation matrix

Usage

```
residual_correlation(ard_residuals, ard, type = "column")
```

Arguments

ard_residuals	vector of residuals
ard	ard matrix
type	type of correlation to use (row or column)

Value

a ggplot of the specified correlation matrix

residual_heatmap	<i>Construct heatmap of residuals</i>
------------------	---------------------------------------

Description

Construct heatmap of residuals

Usage

```
residual_heatmap(ard_residuals, ard)
```

Arguments

ard_residuals	a vector (column wise) of estimated residuals
ard	an ard matrix

Value

A ggplot of residual heatmap

rqr_nbinom_logs	<i>compute numerically stable negative binomial rqr</i>
-----------------	---

Description

compute numerically stable negative binomial rqr

Usage

```
rqr_nbinom_logs(y, size, prob, eps = 1e-12)
```

Arguments

y	observed value
size	size parameter
prob	prob parameter
eps	precision parameter

Value

appropriate randomized quantile residual

rqr_pois_logs	<i>compute numerically stable Poisson rqr</i>
---------------	---

Description

compute numerically stable Poisson rqr

Usage

```
rqr_pois_logs(y, mu, eps = 1e-12)
```

Arguments

y	observed value
mu	mean value of poisson
eps	precision parameter

Value

appropriate randomized quantile residual

scaling	<i>Scale raw log degree and log prevalence estimates</i>
---------	--

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 <code>ard</code> .
known_ind	The indices that correspond to the columns of <code>ard</code> with <code>known_sizes</code> . By default, the function assumes the first <code>n_known</code> columns, where <code>n_known</code> corresponds to the number of <code>known_sizes</code> .
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 <code>known_sizes</code> are used. If <code>G2_ind</code> and <code>B2_ind</code> are not provided, 'C = C_1', so only <code>G1_ind</code> are used. If <code>G1_ind</code> 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 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

Zheng, T., Salganik, M. J., and Gelman, A. (2006). How many people do you know in prison, *Journal of the American Statistical Association*, **101:474**, 409–423

Laga, I., Bao, L., and Niu, X (2021). A Correlated Network Scaleup Model: Finding the Connection Between Subpopulations

tw_group_corr_test	<i>Tracy-Widom test for residual group correlation</i>
--------------------	--

Description

Tracy-Widom test for residual group correlation

Usage

```
tw_group_corr_test(model_fit, correction = c("none", "half"), plot = TRUE)
```

Arguments

- model_fit fitted model object
- correction correction constant, either "none", "half"
- plot a logical, whether to return a ggplot density plot of TW with observed statistic

Value

a list containing test statistic, p-value, and diagnostic plots

Index

- * **datasets**
 - example_data, 9
- construct_pearson, 2
- construct_rqr, 3
- correlatedStan, 3
- cov_plots, 8
- dispersion_metric, 8
- example_data, 9
- extract, 6, 20
- fit_mle, 9
- get_surrogate, 10
- hang_rootogram_ard, 10
- killworth, 11
- log_mix_uniform, 12
- make_ard, 13
- make_ard_tidy, 14
- networkscaleup, 15
- networkscaleup-package
 - (networkscaleup), 15
- overdispersed, 15, 18
- overdispersedStan, 5, 15, 18, 25
- plot_fitted, 21
- plot_tw_statistic, 21
- residual_correlation, 22
- residual_heatmap, 23
- rqr_nbinom_logs, 23
- rqr_pois_logs, 24
- scaling, 4, 24
- stan, 5, 19
- tw_group_corr_test, 26