Package ‘GGMncv’

December 14, 2021

Type Package

Title Gaussian Graphical Models with Nonconvex Regularization

Version 2.1.1

Date 2021-12-13

Description
Estimate Gaussian graphical models with nonconvex penalties <doi:10.31234/osf.io/ad57p>,
including the atan Wang and Zhu (2016) <doi:10.1155/2016/6495417>,
seamless L0 Dicker, Huang, and Lin (2013) <doi:10.5705/ss.2011.074>,
exponential Wang, Fan, and Zhu <doi:10.1007/s10463-016-0588-3>,
smooth integration of counting and absolute deviation Lv and Fan (2009) <doi:10.1214/09-AOS683>,
logarithm Mazumder, Friedman, and Hastie (2011) <doi:10.1198/jasa.2011.tm09738>,
Lq, smoothly clipped absolute deviation Fan and Li (2001) <doi:10.1198/016214501753382273>,
and minimax concave penalty Zhang (2010) <doi:10.1214/09-AOS729>. There are also extensions
for computing variable inclusion probabilities, multiple regression coefficients, and
statistical inference <doi:10.1214/15-EJS1031>.

License GPL-2

Depends R (>= 4.0.0)

Imports Rcpp (>= 1.0.4.6),
Rdpack (>= 0.11-1),
reshape,
GGally (>= 1.4.0),
ggplot2 (>= 3.3.0),
glassoFast (>= 1.0),
network (>= 1.15),
numDeriv (>= 2016.8-1.1),
mathjaxr (>= 1.0-1),
MASS (>= 7.3-51.5),
methods,
parallel,
pbapply,
sna (>= 2.5),
stats,
utils

Suggests car,
corpcor,
corrplot,

dplyr,
NetworkToolbox,
NetworkComparisonTest,
nlshrink,
rmarkdown,
knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.2

LinkingTo Rcpp,
RcppArmadillo

RdMacros Rdpack,
mathjaxr

BugReports https://github.com/donaldRwilliams/GGMncv/issues

VignetteBuilder knitr

R topics documented:

`GGMncv-package` *GGMncv: Gaussian Graphical Models with Nonconvex Regularization*

Description

The primary goal of GGMncv is to provide non-convex penalties for estimating Gaussian graphical models. These are known to overcome the various limitations of lasso (least absolute shrinkage "screening" operator), including inconsistent model selection (Zhao and Yu 2006), biased estimates (Zhang 2010), and a high false positive rate (see for example Williams and Rast 2020;Williams et al. 2019)

Several of the penalties are (continuous) approximations to the ℓ_0 penalty, that is, best subset selection. However, the solution does not require enumerating all possible models which results in a computationally efficient solution.

L0 Approximations

- Atan: `penalty = "atan"` (Wang and Zhu 2016). This is currently the default.
- Seamless ℓ_0: `penalty = "selo"` (Dicker et al. 2013).
- Exponential: `penalty = "exp"` (Wang et al. 2018)
- Log: `penalty = "log"` (Mazumder et al. 2011).
- Sica: `penalty = "sica"` (Lv and Fan 2009)

Additional penalties:

- SCAD: `penalty = "scad"` (Fan and Li 2001).
- MCP: `penalty = "mcp"` (Zhang 2010).
- Adaptive lasso: `penalty = "adapt"` (Zou 2006).
- Lasso: `penalty = "lasso"` (Tibshirani 1996).

Citing GGMncv

It is important to note that GGMncv merely provides a software implementation of other researchers work. There are no methodological innovations, although this is the most comprehensive R package for estimating GGMs with non-convex penalties. Hence, in addition to citing the package `citation("GGMncv")`, it is important to give credit to the primary sources. The references are provided above and in `ggmncv`.

Further, a survey (or review) of these penalties can be found in Williams (2020).

References

Dicker L, Huang B, Lin X (2013). "Variable selection and estimation with the seamless-L 0 penalty." *Statistica Sinica*, 929–962.

Fan J, Li R (2001). "Variable selection via nonconcave penalized likelihood and its oracle properties." *Journal of the American statistical Association*, **96**(456), 1348–1360.

Lv J, Fan Y (2009). "A unified approach to model selection and sparse recovery using regularized least squares." *The Annals of Statistics*, **37**(6A), 3498–3528.

Mazumder R, Friedman JH, Hastie T (2011). "Sparsenet: Coordinate descent with nonconvex penalties." *Journal of the American Statistical Association*, **106**(495), 1125–1138.

Tibshirani R (1996). "Regression shrinkage and selection via the lasso." *Journal of the Royal Statistical Society: Series B (Methodological)*, **58**(1), 267–288.

Wang Y, Fan Q, Zhu L (2018). "Variable selection and estimation using a continuous approximation to the L0 penalty." *Annals of the Institute of Statistical Mathematics*, **70**(1), 191–214.

Wang Y, Zhu L (2016). "Variable selection and parameter estimation with the Atan regularization method." *Journal of Probability and Statistics*.

Williams DR (2020). "Beyond Lasso: A Survey of Nonconvex Regularization in Gaussian Graphical Models." *PsyArXiv*.

Williams DR, Rast P (2020). "Back to the basics: Rethinking partial correlation network methodology." *British Journal of Mathematical and Statistical Psychology*, **73**(2), 187–212.

Williams DR, Rhemtulla M, Wysocki AC, Rast P (2019). "On nonregularized estimation of psychological networks." *Multivariate behavioral research*, **54**(5), 719–750.

Zhang C (2010). "Nearly unbiased variable selection under minimax concave penalty." *The Annals of statistics*, **38**(2), 894–942.

Zhao P, Yu B (2006). "On model selection consistency of Lasso." *Journal of Machine learning research*, **7**(Nov), 2541–2563.

Zou H (2006). "The adaptive lasso and its oracle properties." *Journal of the American statistical association*, **101**(476), 1418–1429.

`bfi` *Data: 25 Personality items representing 5 factors*

Description

This dataset and the corresponding documentation was taken from the **psych** package. We refer users to that package for further details (Revelle 2019).

Usage

```
data("bfi")
```

Format

A data frame with 25 variables and 2800 observations (including missing values)

Details

- `A1` Am indifferent to the feelings of others. (q_146)
- `A2` Inquire about others' well-being. (q_1162)
- `A3` Know how to comfort others. (q_1206)

- `A4` Love children. (q_1364)
- `A5` Make people feel at ease. (q_1419)
- `C1` Am exacting in my work. (q_124)
- `C2` Continue until everything is perfect. (q_530)
- `C3` Do things according to a plan. (q_619)
- `C4` Do things in a half-way manner. (q_626)
- `C5` Waste my time. (q_1949)
- `E1` Don't talk a lot. (q_712)
- `E2` Find it difficult to approach others. (q_901)
- `E3` Know how to captivate people. (q_1205)
- `E4` Make friends easily. (q_1410)
- `E5` Take charge. (q_1768)
- `N1` Get angry easily. (q_952)
- `N2` Get irritated easily. (q_974)
- `N3` Have frequent mood swings. (q_1099)
- `N4` Often feel blue. (q_1479)
- `N5` Panic easily. (q_1505)
- `o1` Am full of ideas. (q_128)
- `o2` Avoid difficult reading material.(q_316)
- `o3` Carry the conversation to a higher level. (q_492)
- `o4` Spend time reflecting on things. (q_1738)
- `o5` Will not probe deeply into a subject. (q_1964)
- `gender` Males = 1, Females =2
- `education` 1 = HS, 2 = finished HS, 3 = some college, 4 = college graduate 5 = graduate degree

References

Revelle W (2019). *psych: Procedures for Psychological, Psychometric, and Personality Research*. Northwestern University, Evanston, Illinois. R package version 1.9.12, https://CRAN.R-project.org/package=psych.

`boot_eip` *Bootstrapped Edge Inclusion 'Probabilities'*

Description

Compute the number of times each edge was selected when performing a non-parametric bootstrap (see Figure 6.7, Hastie et al. 2009).

Usage

```
boot_eip(Y, method = "pearson", samples = 500, progress = TRUE, ...)
```

Arguments

Y	A matrix of dimensions *n* by *p*.
method	Character string. Which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman".
samples	Numeric. How many bootstrap samples (defaults to `500`)?
progress	Logical. Should a progress bar be included (defaults to `TRUE`)?
...	Additional arguments passed to `ggmncv`.

Value

An object of class `eip` that includes the "probabilities" in a data frame.

Note

Although Hastie et al. (2009) suggests this approach provides probabilities, to avoid confusion with Bayesian inference, these are better thought of as "probabilities" (or better yet proportions).

References

Hastie T, Tibshirani R, Friedman J (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science \& Business Media.

Examples

```
# data (ptsd symptoms)
Y <- GGMncv::ptsd[,1:10]

# compute eip's
boot_samps <- boot_eip(Y, samples  = 100, progress = FALSE)

boot_samps
```

`coef.ggmncv` *Regression Coefficients from* `ggmncv` *Objects*

Description

There is a direct correspondence between the inverse covariance matrix and multiple regression (Stephens 1998; Kwan 2014). This readily allows for converting the off diagonal elements to regression coefficients, resulting in noncovex penalization for multiple regression modeling.

Usage

```
## S3 method for class 'ggmncv'
coef(object, ...)
```

Arguments

object	An Object of class `ggmncv`.
...	Currently ignored.

Value

A matrix of regression coefficients.

Note

The coefficients can be accessed via `coefs[1,]`, which provides the estimates for predicting the first node.

Further, the estimates are essentially computed with both the outcome and predictors scaled to have mean 0 and standard deviation 1.

References

Kwan CC (2014). "A regression-based interpretation of the inverse of the sample covariance matrix." *Spreadsheets in Education*, **7**(1), 4613.

Stephens G (1998). "On the Inverse of the Covariance Matrix in Portfolio Analysis." *The Journal of Finance*, **53**(5), 1821–1827.

Examples

```
# data
Y <- GGMncv::ptsd[,1:5]

# correlations
S <- cor(Y)

# fit model
fit <- ggmncv(R = S, n = nrow(Y), progress = FALSE)

# regression
coefs <- coef(fit)

coefs


# no regularization, resulting in OLS

# data
# note: scaled for lm()
Y <- scale(GGMncv::ptsd[,1:5])

# correlations
S <- cor(Y)

# fit model
# note: non reg
fit <- ggmncv(R = S, n = nrow(Y), progress = FALSE, lambda = 0)

# regression
coefs <- coef(fit)

# fit lm
fit_lm <- lm(Y[,1] ~ 0 + Y[,-1])
```

```
# ggmncv
coefs[1,]

# lm
as.numeric(coef(fit_lm))
```

---

compare_edges *Compare Edges Between Gaussian Graphical Models*

Description

Establish whether each of the corresponding edges are significantly different in two groups, with the de-sparsified estimator of (Jankova and Van De Geer 2015).

Usage

```
compare_edges(object_1, object_2, method = "fdr", alpha = 0.05, ...)
```

Arguments

`object_1`	object of class `ggmncv` .
`object_2`	An object of class `ggmncv`.
`method`	Character string. A correction method for multiple comparisons (defaults to `fdr`), which can be abbreviated. See p.adjust.
`alpha`	Numeric. Significance level (defaults to `0.05`).
`...`	Currently ignored.

Value

- `P_diff` De-sparsified partial correlation differences
- `adj` Adjacency matrix based on the p-values.
- `pval_uncorrected` Uncorrected p-values
- `pval_corrected` Corrected p-values
- `method` The approach used for multiple comparisons
- `alpha` Significance level

Note

For low-dimensional settings, i.e., when the number of observations far exceeds the number of nodes, this function likely has limited utility and a non regularized approach should be used for comparing edges (see for example **GGMnonreg**).

Further, whether the de-sparsified estimator provides nominal error rates remains to be seen, at least across a range of conditions. For example, the simulation results in Williams (2021) demonstrated that the confidence intervals can have (severely) compromised coverage properties (whereas non-regularized methods had coverage at the nominal level).

References

Jankova J, Van De Geer S (2015). "Confidence intervals for high-dimensional inverse covariance estimation." *Electronic Journal of Statistics*, **9**(1), 1205–1229.

Williams DR (2021). "The Confidence Interval that Wasn't: Bootstrapped "Confidence Intervals" in L1-Regularized Partial Correlation Networks." *PsyArXiv*. doi: 10.31234/osf.io/kjh2f.

Examples

```
# data
# note: all edges equal
Y1 <- MASS::mvrnorm(250, rep(0, 10), Sigma = diag(10))
Y2 <- MASS::mvrnorm(250, rep(0, 10), Sigma = diag(10))

# fit models
# note: atan penalty by default

# group 1
fit1 <- ggmncv(cor(Y1), n = nrow(Y1),
               progress = FALSE)

# group 2
fit2 <- ggmncv(cor(Y2), n = nrow(Y2),
               progress = FALSE)

# compare
compare_ggms <- compare_edges(fit1, fit2)

compare_ggms
```

`confirm_edges` *Confirm Edges*

Description

Confirmatory hypothesis testing of edges that were initially detected with data-driven model selection.

Usage

```
confirm_edges(object, Rnew, method, alpha)
```

Arguments

`object`	An object of class ggmncv
`Rnew`	Matrix. A correlation matrix of dimensions p by p.
`method`	Character string. A correction method for multiple comparison (defaults to `fdr`). Can be abbreviated. See p.adjust.
`alpha`	Numeric. Significance level (defaults to `0.05`).

Details

The basic idea is to merge exploration with confirmation (see for example, Rodriguez et al. 2020). This is accomplished by testing those edges (in an independent dataset) that were initially detected via data driven model selection.

Confirmatory hypothesis testing follows the approach described in Jankova and Van De Geer (2015): (1) graphical lasso is computed with lambda fixed to $\lambda = \sqrt{log(p)/n}$, (2) the de-sparsified estimator is computed, and then (3) p-values are obtained for the de-sparsified estimator.

Value

An object of class ggmncv, including:

- **P**: Matrix of confirmed edges (partial correlations)
- **adj**: Matrix of confirmed edges (adjacency)

References

Jankova J, Van De Geer S (2015). "Confidence intervals for high-dimensional inverse covariance estimation." *Electronic Journal of Statistics*, **9**(1), 1205–1229.

Rodriguez JE, Williams DR, Rast P, Mulder J (2020). "On Formalizing Theoretical Expectations: Bayesian Testing of Central Structures in Psychological Networks." *PsyArXiv*. doi: 10.31234/osf.io/zw7pf.

Examples

```
Y <- na.omit(bfi[,1:25])

Y_explore <- Y[1:1000,]

Y_confirm <- Y[1001:nrow(Y),]

fit <- ggmncv(cor(Y_explore),
              n = nrow(Y_explore),
              progress = FALSE)

confirm <- confirm_edges(fit,
                         Rnew = cor(Y_confirm),
                         method = "fdr",
                         alpha = 0.05)
```

---

constrained *Precision Matrix with Known Graph*

Description

Compute the maximum likelihood estimate of the precision matrix, given a known graphical structure (i.e., an adjacency matrix). This approach was originally described in "The Elements of Statistical Learning" (see pg. 631, Hastie et al. 2009).

Usage

```
constrained(Sigma, adj)

mle_known_graph(Sigma, adj)
```

Arguments

Sigma	Covariance matrix
adj	Adjacency matrix that encodes the constraints, where a zero indicates that element should be zero.

Value

A list containing the following:

- **Theta**: Inverse of the covariance matrix (precision matrix)
- **Sigma**: Covariance matrix.
- **wadj**: Weighted adjacency matrix, corresponding to the partial correlation network.

Note

The algorithm is written in c++, and should scale to high dimensions nicely.

Note there are a variety of algorithms for this purpose. Simulation studies indicated that this approach is both accurate and computationally efficient (HFT therein, Emmert-Streib et al. 2019)

References

Emmert-Streib F, Tripathi S, Dehmer M (2019). "Constrained covariance matrices with a biologically realistic structure: Comparison of methods for generating high-dimensional Gaussian graphical models." *Frontiers in Applied Mathematics and Statistics*, **5**, 17.

Hastie T, Tibshirani R, Friedman J (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science \& Business Media.

Examples

```
# data
y <- ptsd

# fit model
fit <- ggmncv(cor(y), n = nrow(y),
              penalty = "lasso",
              progress = FALSE)

# set negatives to zero (sign restriction)
adj_new <- ifelse( fit$P <= 0, 0, 1)

check_zeros <- TRUE

# track trys
iter <- 0
```

```
# iterate until all positive
while(check_zeros){
  iter <- iter + 1
  fit_new <- constrained(cor(y), adj = adj_new)
  check_zeros <- any(fit_new$wadj < 0)
  adj_new <- ifelse( fit_new$wadj <= 0, 0, 1)
}



# alias

# data
y <- ptsd

# nonreg (lambda = 0)
fit <- ggmncv(cor(y), n = nrow(y),
              lambda = 0,
              progress = FALSE)

# set values less than |0.1| to zero
adj_new <- ifelse( abs(fit$P) <= 0.1, 0, 1)

# mle given the graph
mle_known_graph(cor(y), adj_new)
```

---

desparsify *De-Sparsified Graphical Lasso Estimator*

Description

Compute the de-sparsified (sometimes called "de-biased") glasso estimator with the approach described in Equation 7 of Jankova and Van De Geer (2015). The basic idea is to *undo* L_1-regularization, in order to compute *p*-values and confidence intervals (i.e., to make statistical inference).

Usage

```
desparsify(object, ...)
```

Arguments

object	An object of class ggmncv.
...	Currently ignored.

Details

According to Jankova and Van De Geer (2015), the de-sparisifed estimator, $\hat{\mathbf{T}}$, is defined as

$\hat{\mathbf{T}} = 2\hat{\mathbf{\Theta}} - \hat{\mathbf{\Theta}}\hat{\mathbf{R}}\hat{\mathbf{\Theta}},$

where $\hat{\mathbf{\Theta}}$ denotes the graphical lasso estimator of the precision matrix and $\hat{\mathbf{R}}$ is the sample correlation matrix. Further details can be found in Section 2 ("Main Results") of Jankova and Van De Geer (2015).

This approach is built upon earlier work on the de-sparsified lasso estimator (Javanmard and Montanari 2014; Van de Geer et al. 2014; Zhang and Zhang 2014)

Value

The de-sparsified estimates, including

- `Theta`: De-sparsified precision matrix
- `P`: De-sparsified partial correlation matrix

Note

This assumes (reasonably) Gaussian data, and should not to be expected to work for, say, polychoric correlations. Further, all work to date has only looked at the graphical lasso estimator, and not de-sparsifying nonconvex regularization. Accordingly, it is probably best to set `penalty = "lasso"` in `ggmncv`.

This function only provides the de-sparsified estimator and not *p*-values or confidence intervals (see `inference`).

References

Jankova J, Van De Geer S (2015). "Confidence intervals for high-dimensional inverse covariance estimation." *Electronic Journal of Statistics*, **9**(1), 1205–1229.

Javanmard A, Montanari A (2014). "Confidence intervals and hypothesis testing for high-dimensional regression." *The Journal of Machine Learning Research*, **15**(1), 2869–2909.

Van de Geer S, BÃ¼hlmann P, Ritov Y, Dezeure R (2014). "On asymptotically optimal confidence regions and tests for high-dimensional models." *The Annals of Statistics*, **42**(3), 1166–1202.

Zhang C, Zhang SS (2014). "Confidence intervals for low dimensional parameters in high dimensional linear models." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **76**(1), 217–242.

Examples

```
# data
Y <- GGMncv::Sachs[,1:5]

n <- nrow(Y)
p <- ncol(Y)

# fit model
# note: fix lambda, as in the reference
fit <- ggmncv(cor(Y), n = nrow(Y),
              progress = FALSE,
              penalty = "lasso",
              lambda = sqrt(log(p)/n))

# fit model
# note: no regularization
fit_non_reg <- ggmncv(cor(Y), n = nrow(Y),
                      progress = FALSE,
                      penalty = "lasso",
                      lambda = 0)


# remove (some) bias and sparsity
```

```
That <- desparsify(fit)

# graphical lasso estimator
fit$P

# de-sparsified estimator
That$P

# mle
fit_non_reg$P
```

---

gen_net *Simulate a Partial Correlation Matrix*

Description

Simulate a Partial Correlation Matrix

Usage

```
gen_net(p = 20, edge_prob = 0.3, lb = 0.05, ub = 0.3)
```

Arguments

p	number of variables (nodes)
edge_prob	connectivity
lb	lower bound for the partial correlations
ub	upper bound for the partial correlations

Value

A list containing the following:

- **pcor**: Partial correlation matrix, encoding the conditional (in)dependence structure.
- **cors**: Correlation matrix.
- **adj**: Adjacency matrix.
- **trys**: Number of attempts to obtain a positive definite matrix.

Note

The function checks for a valid matrix (positive definite), but sometimes this will still fail. For example, for larger p, to have large partial correlations this requires a sparse GGM (accomplished by setting edge_prob to a small value).

Examples

```
p <- 20
n <- 500

true_net <- gen_net(p = p, edge_prob = 0.25)

y <- MASS::mvrnorm(n = n,
                   mu = rep(0, p),
                   Sigma = true_net$cors)

# default
fit_atan <- ggmncv(R = cor(y),
                   n = nrow(y),
                   penalty = "atan",
                   progress = FALSE)

# lasso
fit_l1 <- ggmncv(R = cor(y),
                 n = nrow(y),
                 penalty = "lasso",
                 progress = FALSE)

# atan
score_binary(estimate = true_net$adj,
             true = fit_atan$adj,
             model_name = "atan")

# lasso
score_binary(estimate = fit_l1$adj,
             true = true_net$adj,
             model_name = "lasso")
```

---

get_graph — *Extract Graph from* ggmncv *Objects*

Description

The fitted model from ggmncv contains a lot of information, most of which is not immediately useful for most use cases. This function extracts the weighted adjacency (partial correlation network) and adjacency matrices.

Usage

```
get_graph(x, ...)
```

Arguments

x	An object of class ggmncv.
...	Currently ignored.

Value

- P: Weighted adjacency matrix (partial correlation network)
- adj: Adjacency matrix

Examples

```
Y <- na.omit(bfi[,1:5])

fit <- ggmncv(cor(Y),
              n = nrow(Y),
              progress = FALSE)

get_graph(fit)
```

---

ggmncv *GGMncv*

Description

Gaussian graphical modeling with nonconvex regularization. A thorough survey of these penalties, including simulation studies investigating their properties, is provided in Williams (2020).

Usage

```
ggmncv(
  R,
  n,
  penalty = "atan",
  ic = "bic",
  select = "lambda",
  gamma = NULL,
  lambda = NULL,
  n_lambda = 50,
  lambda_min_ratio = 0.01,
  n_gamma = 50,
  initial = NULL,
  LLA = FALSE,
  unreg = FALSE,
  maxit = 10000,
  thr = 1e-04,
  store = TRUE,
  progress = TRUE,
  ebic_gamma = 0.5,
  penalize_diagonal = TRUE,
  ...
)
```

Arguments

- `R` — Matrix. A correlation matrix of dimensions p by p.
- `n` — Numeric. The sample size used to compute the information criterion.
- `penalty` — Character string. Which penalty should be used (defaults to `"atan"`)?
- `ic` — Character string. Which information criterion should be used (defaults to `"bic"`)? The options include `aic`, `ebic` (ebic_gamma defaults to `0.5`), `ric`, or any of the generalized information criteria provided in section 5 of Kim et al. (2012). The options are `gic_1` (i.e., `bic`) to `gic_6` (see '`Details`').
- `select` — Character string. Which tuning parameter should be selected (defaults to `"lambda"`)? The options include `"lambda"` (the regularization parameter), `"gamma"` (governs the 'shape'), and `"both"`.
- `gamma` — Numeric. Hyperparameter for the penalty function. Defaults to 3.7 (`scad`), 2 (`mcp`), 0.5 (`adapt`), and 0.01 with all other penalties. Note care must be taken when departing from the default values (see the references in '`note`')
- `lambda` — Numeric vector. Regularization (or tuning) parameters. The defaults is `NULL` that provides default values with `select = "lambda"` and `sqrt(log(p)/n)` with `select = "gamma"`.
- `n_lambda` — Numeric. The number of λ's to be evaluated. Defaults to 50. This is disregarded if custom values are provided for `lambda`.
- `lambda_min_ratio` — Numeric. The smallest value for `lambda`, as a fraction of the upperbound of the regularization/tuning parameter. The default is `0.01`, which mimics the `R` package **qgraph**. To mimic the `R` package **huge**, set `lambda_min_ratio = 0.1` and `n_lambda = 10`.
- `n_gamma` — Numeric. The number of γ's to be evaluated. Defaults to 50. This is disregarded if custom values are provided in `lambda`.
- `initial` — A matrix (p by p) or custom function that returns the inverse of the covariance matrix . This is used to compute the penalty derivative. The default is `NULL`, which results in using the inverse of `R` (see '`Note`').
- `LLA` — Logical. Should the local linear approximation be used (default to `FALSE`)?
- `unreg` — Logical. Should the models be refitted (or unregularized) with maximum likelihood (defaults to `FALSE`)? Setting to `TRUE` results in the approach of Foygel and Drton (2010), but with the regularization path obtained from nonconvex regularization, as opposed to the ℓ_1-penalty.
- `maxit` — Numeric. The maximum number of iterations for determining convergence of the LLA algorithm (defaults to `1e4`). Note this can be changed to, say, `2` or `3`, which will provide two and three-step estimators without convergence check.
- `thr` — Numeric. Threshold for determining convergence of the LLA algorithm (defaults to `1.0e-4`).
- `store` — Logical. Should all of the fitted models be saved (defaults to `TRUE`)?
- `progress` — Logical. Should a progress bar be included (defaults to `TRUE`)?
- `ebic_gamma` — Numeric. Value for the additional hyper-parameter for the extended Bayesian information criterion (defaults to 0.5, must be between 0 and 1). Setting `ebic_gamma = 0` results in BIC.
- `penalize_diagonal` — Logical. Should the diagonal of the inverse covariance matrix be penalized (defaults to `TRUE`).
- `...` — Additional arguments passed to `initial` when a function is provided and ignored otherwise.

Details

Several of the penalties are (continuous) approximations to the ℓ_0 penalty, that is, best subset selection. However, the solution does not require enumerating all possible models which results in a computationally efficient solution.

L0 Approximations

- Atan: `penalty = "atan"` (Wang and Zhu 2016). This is currently the default.
- Seamless ℓ_0: `penalty = "selo"` (Dicker et al. 2013).
- Exponential: `penalty = "exp"` (Wang et al. 2018)
- Log: `penalty = "log"` (Mazumder et al. 2011).
- Sica: `penalty = "sica"` (Lv and Fan 2009)

Additional penalties:

- SCAD: `penalty = "scad"` (Fan and Li 2001).
- MCP: `penalty = "mcp"` (Zhang 2010).
- Adaptive lasso (`penalty = "adapt"`): Defaults to $\gamma = 0.5$ (Zou 2006). Note that for consistency with the other penalties, $\gamma \to 0$ provides more penalization and $\gamma = 1$ results in ℓ_1 regularization.
- Lasso: `penalty = "lasso"` (Tibshirani 1996).

gamma (γ):

The `gamma` argument corresponds to additional hyperparameter for each penalty. The defaults are set to the recommended values from the respective papers.

LLA

The local linear approximate is noncovex penalties was described in (Fan et al. 2009). This is essentially an iteratively re-weighted (g)lasso. Note that by default `LLA = FALSE`. This is due to the work of Zou and Li (2008), which suggested that, so long as the starting values are good enough, then a one-step estimator is sufficient to obtain an accurate estimate of the conditional dependence structure. In the case of low-dimensional data, the sample based inverse covariance matrix is used for the starting values. This is expected to work well, assuming that n is sufficiently larger than p.

Generalized Information Criteria

The following are the available GIC:

- $\text{GIC}_1 : |\mathbf{E}| \cdot \log(n)$ (`ic = "gic_1"` or `ic = "bic"`)
- $\text{GIC}_2 : |\mathbf{E}| \cdot p^{1/3}$ (`ic = "gic_2"`)
- $\text{GIC}_3 : |\mathbf{E}| \cdot 2 \cdot \log(p)$ (`ic = "gic_3"` or `ic = "ric"`)
- $\text{GIC}_4 : |\mathbf{E}| \cdot 2 \cdot \log(p) + \log\big(\log(p)\big)$ (`ic = "gic_4"`)
- $\text{GIC}_5 : |\mathbf{E}| \cdot \log(p) + \log\big(\log(n)\big) \cdot \log(p)$ (`ic = "gic_5"`)
- $\text{GIC}_6 : |\mathbf{E}| \cdot \log(n) \cdot \log(p)$ (`ic = "gic_6"`)

Note that $|\mathbf{E}|$ denotes the number of edges (nonzero relations) in the graph, p the number of nodes (columns), and n the number of observations (rows). Further each can be understood as a penalty term added to negative 2 times the log-likelihood, that is,

$$-2l_n(\hat{\mathbf{\Theta}}) = -2\Big[\frac{n}{2}\text{logdet}\hat{\mathbf{\Theta}} - \text{tr}(\hat{\mathbf{S}}\hat{\mathbf{\Theta}})\Big]$$

where $\hat{\mathbf{\Theta}}$ is the estimated precision matrix (e.g., for a given λ and γ) and $\hat{\mathbf{S}}$ is the sample-based covariance matrix.

Value

An object of class ggmncv, including:

- Theta Inverse covariance matrix
- Sigma Covariance matrix
- P Weighted adjacency matrix
- adj Adjacency matrix
- lambda Tuning parameter(s)
- fit glasso fitted model (a list)

Note

initial

initial not only affects performance (to some degree) but also computational speed. In high dimensions (defined here as $p > n$), or when p approaches n, the precision matrix can become quite unstable. As a result, with initial = NULL, the algorithm can take a very (very) long time. If this occurs, provide a matrix for initial (e.g., using lw). Alternatively, the penalty can be changed to penalty = "lasso", if desired.

The R package **glassoFast** is under the hood of ggmncv (Sustik and Calderhead 2012), which is much faster than **glasso** when there are many nodes.

References

Dicker L, Huang B, Lin X (2013). "Variable selection and estimation with the seamless-L 0 penalty." *Statistica Sinica*, 929–962.

Fan J, Feng Y, Wu Y (2009). "Network exploration via the adaptive LASSO and SCAD penalties." *The annals of applied statistics*, **3**(2), 521.

Fan J, Li R (2001). "Variable selection via nonconcave penalized likelihood and its oracle properties." *Journal of the American statistical Association*, **96**(456), 1348–1360.

Foygel R, Drton M (2010). "Extended Bayesian Information Criteria for Gaussian Graphical Models." *Advances in Neural Information Processing Systems*, 604–612. 1011.6640.

Kim Y, Kwon S, Choi H (2012). "Consistent model selection criteria on high dimensions." *The Journal of Machine Learning Research*, **13**, 1037–1057.

Lv J, Fan Y (2009). "A unified approach to model selection and sparse recovery using regularized least squares." *The Annals of Statistics*, **37**(6A), 3498–3528.

Mazumder R, Friedman JH, Hastie T (2011). "Sparsenet: Coordinate descent with nonconvex penalties." *Journal of the American Statistical Association*, **106**(495), 1125–1138.

Sustik MA, Calderhead B (2012). "GLASSOFAST: An efficient GLASSO implementation." *UTCS Technical Report TR-12-29 2012*.

Tibshirani R (1996). "Regression shrinkage and selection via the lasso." *Journal of the Royal Statistical Society: Series B (Methodological)*, **58**(1), 267–288.

Wang Y, Fan Q, Zhu L (2018). "Variable selection and estimation using a continuous approximation to the L0 penalty." *Annals of the Institute of Statistical Mathematics*, **70**(1), 191–214.

Wang Y, Zhu L (2016). "Variable selection and parameter estimation with the Atan regularization method." *Journal of Probability and Statistics*.

Williams DR (2020). "Beyond Lasso: A Survey of Nonconvex Regularization in Gaussian Graphical Models." *PsyArXiv*.

Zhang C (2010). "Nearly unbiased variable selection under minimax concave penalty." *The Annals of statistics*, **38**(2), 894–942.

Zou H (2006). "The adaptive lasso and its oracle properties." *Journal of the American statistical association*, **101**(476), 1418–1429.

Zou H, Li R (2008). "One-step sparse estimates in nonconcave penalized likelihood models." *Annals of statistics*, **36**(4), 1509.

Examples

```
# data
Y <- GGMncv::ptsd

S <- cor(Y)

# fit model
# note: atan default
fit_atan <- ggmncv(S, n = nrow(Y),
                   progress = FALSE)

# plot
plot(get_graph(fit_atan),
     edge_magnify = 10,
     node_names = colnames(Y))

# lasso
fit_l1 <- ggmncv(S, n = nrow(Y),
                 progress = FALSE,
                 penalty = "lasso")

# plot
plot(get_graph(fit_l1),
     edge_magnify = 10,
     node_names = colnames(Y))


# for these data, we might expect all relations to be positive
# and thus the red edges are spurious. The following re-estimates
# the graph, given all edges positive (sign restriction).

# set negatives to zero (sign restriction)
adj_new <- ifelse( fit_atan$P <= 0, 0, 1)

check_zeros <- TRUE
```

```
# track trys
iter <- 0

# iterate until all positive
while(check_zeros){
  iter <- iter + 1
  fit_new <- constrained(S, adj = adj_new)
  check_zeros <- any(fit_new$wadj < 0)
  adj_new <- ifelse( fit_new$wadj <= 0, 0, 1)
}

# make graph object
new_graph <- list(P = fit_new$wadj,
                  adj = adj_new)
class(new_graph) <- "graph"

plot(new_graph,
     edge_magnify = 10,
     node_names = colnames(Y))
```

`head.eip` *Print the Head of* `eip` *Objects*

Description

Print the Head of `eip` Objects

Usage

```
## S3 method for class 'eip'
head(x, n = 5, ...)
```

Arguments

`x`	An object of class `eip`
`n`	Numeric. Number of rows to print.
`...`	Currently ignored.

`inference` *Statistical Inference for Regularized Gaussian Graphical Models*

Description

Compute p-values for each relation based on the de-sparsified glasso estimator (Jankova and Van De Geer 2015).

Usage

```
inference(object, method = "fdr", alpha = 0.05, ...)

significance_test(object, method = "fdr", alpha = 0.05, ...)
```

Arguments

object	An object of class ggmncv
method	Character string. A correction method for multiple comparison (defaults to fdr). Can be abbreviated. See p.adjust.
alpha	Numeric. Significance level (defaults to 0.05).
...	Currently ignored.

Value

- Theta De-sparsified precision matrix
- adj Adjacency matrix based on the p-values.
- pval_uncorrected Uncorrected p-values
- pval_corrected Corrected p-values
- method The approach used for multiple comparisons
- alpha Significance level

Note

This assumes (reasonably) Gaussian data, and should not to be expected to work for, say, polychoric correlations. Further, all work to date has only looked at the graphical lasso estimator, and not de-sparsifying nonconvex regularization. Accordingly, it is probably best to set penalty = "lasso" in ggmncv.

Further, whether the de-sparsified estimator provides nominal error rates remains to be seen, at least across a range of conditions. For example, the simulation results in Williams (2021) demonstrated that the confidence intervals can have (severely) compromised coverage properties (whereas non-regularized methods had coverage at the nominal level).

References

Jankova J, Van De Geer S (2015). "Confidence intervals for high-dimensional inverse covariance estimation." *Electronic Journal of Statistics*, **9**(1), 1205–1229.

Williams DR (2021). "The Confidence Interval that Wasn't: Bootstrapped "Confidence Intervals" in L1-Regularized Partial Correlation Networks." *PsyArXiv*. doi: 10.31234/osf.io/kjh2f.

Examples

```
# data
Y <- GGMncv::ptsd[,1:5]

# fit model
fit <- ggmncv(cor(Y), n = nrow(Y),
              progress = FALSE,
              penalty = "lasso")
```

```
# statistical inference
inference(fit)

# alias
all.equal(inference(fit), significance_test(fit))
```

---

kl_mvn *Kullback-Leibler Divergence*

Description

Compute KL divergence for a multivariate normal distribution.

Usage

```
kl_mvn(true, estimate, stein = FALSE)
```

Arguments

`true`	Matrix. The true precision matrix (inverse of the covariance matrix)
`estimate`	Matrix. The estimated precision matrix (inverse of the covariance matrix)
`stein`	Logical. Should Stein's loss be computed (defaults to `TRUE`)? Note KL divergence is half of Stein's loss.

Value

Numeric corresponding to KL divergence.

Note

A lower value is better, with a score of zero indicating that the estimated precision matrix is identical to the true precision matrix.

Examples

```
# nodes
p <- 20

main <- gen_net(p = p, edge_prob = 0.15)

y <- MASS::mvrnorm(250, rep(0, p), main$cors)

fit_l1 <- ggmncv(R = cor(y),
              n = nrow(y),
              penalty = "lasso",
              progress = FALSE)

# lasso
kl_mvn(fit_l1$Theta, solve(main$cors))
```

```
fit_atan <- ggmncv(R = cor(y),
              n = nrow(y),
              penalty = "atan",
              progress = FALSE)

kl_mvn(fit_atan$Theta, solve(main$cors))
```

`ledoit_wolf` *Ledoit and Wolf Shrinkage Estimator*

Description

Compute the Ledoit and Wolf shrinkage estimator of the covariance matrix (Ledoit and Wolf 2004), which can be used for the `initial` inverse covariance matrix in `ggmncv`.

Usage

```
ledoit_wolf(Y, ...)
```

Arguments

`Y`	A data matrix (or data.frame) of dimensions n by p.
`...`	Currently ignored.

Value

Inverse correlation matrix.

References

Ledoit O, Wolf M (2004). “A well-conditioned estimator for large-dimensional covariance matrices.” *Journal of Multivariate Analysis*, **88**(2), 365–411.

Examples

```
# ptsd
Y <- ptsd[,1:5]

# shrinkage
ledoit_wolf(Y)

# non-reg
solve(cor(Y))
```

nct *Network Comparison Test*

Description

A re-implementation and extension of the permutation based network comparison test introduced in Van Borkulo et al. (2017). Such extensions include scaling to networks with many nodes and the option to use custom test-statistics.

Usage

```
nct(
  Y_g1,
  Y_g2,
  iter = 1000,
  desparsify = TRUE,
  method = "pearson",
  FUN = NULL,
  cores = 1,
  progress = TRUE,
  update_progress = 4,
  ...
)
```

Arguments

`Y_g1`	A matrix (or data.frame) of dimensions n by p, corresponding to the first dataset (p must be the same for `Y_g1` and `Y_g2`).
`Y_g2`	A matrix of dimensions n by p, corresponding to the second dataset (p must be the same for `Y_g1` and `Y_g2`).
`iter`	Numeric. Number of (Monte Carlo) permutations (defaults to `1000`).
`desparsify`	Logical. Should the de-sparsified glasso estimator be computed (defaults to `TRUE`)? This is much faster, as the tuning parameter is fixed to $\lambda = \sqrt{log(p)/n}$.
`method`	character string. Which correlation coefficient (or covariance) is to be computed. One of "pearson" (default), "kendall", or "spearman".
`FUN`	A function or list of functions (defaults to `NULL`), specifying custom test-statistics. See **Examples**.
`cores`	Numeric. Number of cores to use when executing the permutations in parallel (defaults to `1`).
`progress`	Logical. Should a progress bar be included (defaults to `TRUE`)?
`update_progress`	How many times should the progress bar be updated (defaults to `4`)? Note that setting this to a large value should result in the worse performance, due to additional overhead communicating among the parallel processes.
`...`	Additional arguments passed to `ggmncv`.

Details

User-Defined Functions

These functions must have two arguments, corresponding to the partial correlation network for each group. An example is provided below.

For user-defined functions (FUN), absolute values are used to compute the p-value, assuming more than one value is returned (e.g., centrality). This is done to mimic the R package **NCT**.

A fail-safe method to ensure the p-value is computed correctly is to access the permutations and observed values from the nct object.

Finally, comparing edges is not implemented. The most straightforward way to do this is with compare_edges, which uses the de-sparsified estimator.

Value

A list of class nct, including the following

- glstr_pvalue: Global strength p-value.
- sse_pvalue: Sum of square error p-value.
- jsd_pvalue: Jensen-Shannon divergence p-value.
- max_pvalue: Maximum difference p-value.
- glstr_obs: Global strength observed.
- sse_obs: Sum of square error observed.
- jsd_obs: Jensen-Shannon divergence observed.
- max_obs: Maximum difference observed.
- glstr_perm: Global strength permutations.
- sse_perm: Sum of square error permutations.
- jsd_perm: Jensen-Shannon divergence permutations.
- max_perm: Maximum difference permutations.

For user-defined functions, i.e., those provided to FUN, the function name is pasted to _pvalue, _obs, and _perm.

Note

In Van Borkulo et al. (2017), it was suggested that these are tests of *invariance*. To avoid confusion, that terminology is not used in **GGMncv**. This is because these tests assume invariance or the null is *true*, and thus can only be used to detect differences. Hence, it would be incorrect to suggest networks are the same, or evidence for invariance, by merely failing to reject the null hypothesis (Williams et al. 2021).

For the defaults, Jensen-Shannon divergence is a symmetrized version of Kullback-Leibler divergence (the average of both directions).

Computational Speed

This implementation has two key features that should make it scale to larger networks: (1) parallel computation and (2) the R package **glassoFast** is used under the hood (as opposed to **glasso**). CPU (time) comparisons are provided in Sustik and Calderhead (2012).

Non-regularized

Non-regularized can be implemented by setting lambda = 0. Note this is provided to ggmncv via ....

References

Sustik MA, Calderhead B (2012). "GLASSOFAST: An efficient GLASSO implementation." *UTCS Technical Report TR-12-29 2012*.

Van Borkulo CD, Boschloo L, Kossakowski J, Tio P, Schoevers RA, Borsboom D, Waldorp LJ (2017). "Comparing network structures on three aspects: A permutation test." *Manuscript submitted for publication*, **10**.

Williams DR, Briganti G, Linkowski P, Mulder J (2021). "On Accepting the Null Hypothesis of Conditional Independence in Partial Correlation Networks: A Bayesian Analysis." *PsyArXiv*. doi: 10.31234/osf.io/7uhx8, https://psyarxiv.com/7uhx8.

Examples

```
# generate network
main <- gen_net(p = 10)

# assume groups are equal
y1 <- MASS::mvrnorm(n = 500,
                    mu = rep(0, 10),
                    Sigma = main$cors)

y2 <- MASS::mvrnorm(n = 500,
                    mu = rep(0, 10),
                    Sigma = main$cors)

compare_ggms <- nct(y1, y2, iter = 500,
                    progress = FALSE)

compare_ggms

# custom function
# note: x & y are partial correlation networks

# correlation
Correlation <- function(x, y){
cor(x[upper.tri(x)], y[upper.tri(y)])
}

compare_ggms <- nct(y1, y2,iter = 100,
                    FUN = Correlation,
                    progress = FALSE)

compare_ggms

# correlation and strength

Strength <- function(x, y){
NetworkToolbox::strength(x) - NetworkToolbox::strength(y)
}

compare_ggms <- nct(y1, y2, iter = 100,
                    FUN = list(Correlation = Correlation,
                               Strength = Strength),
                    progress = FALSE)
```

compare_ggms

penalty_derivative *Penalty Derivative*

Description

Compute the derivative for a nonconvex penalty.

Usage

```
penalty_derivative(
  theta = seq(-5, 5, length.out = 1e+05),
  penalty = "atan",
  lambda = 1,
  gamma = c(0.01, 0.05)
)
```

Arguments

theta	Numeric vector. Values for which the derivative is computed.
penalty	Character string. Which penalty should be used (defaults to "atan")? See ggmncv for the available penalties.
lambda	Numeric. Regularization parameter (defaults to 1).
gamma	Numeric vector. Hyperparameter(s) for the penalty function

Value

A list of class penalty_derivative, including the following:

- deriv: Data frame including the derivative, theta, gamma, and the chosen penalty.
- lambda: Regularization parameter.

Note

Some care is required for specifying gamma. For example, the default value for scad is 3.7 and it *must* be some value greater than 2 (Fan and Li 2001). The default values in **GGMncv** are set to recommended values in the respective papers.

References

Fan J, Li R (2001). "Variable selection via nonconcave penalized likelihood and its oracle properties." *Journal of the American statistical Association*, **96**(456), 1348–1360.

Examples

```
deriv <- penalty_derivative(theta =  seq(-5,5,length.out = 10000),
                            lambda = 1,
                            gamma = c(0.01, 0.05, 0.1))

head(deriv$deriv)
```

penalty_function *Penalty Function*

Description

Compute the penalty function for nonconvex penalties.

Usage

```
penalty_function(
  theta = seq(-5, 5, length.out = 1e+05),
  penalty = "atan",
  lambda = 1,
  gamma = c(0.01, 0.05)
)
```

Arguments

theta	Numeric vector. Values for which the derivative is computed.
penalty	Character string. Which penalty should be used (defaults to "atan")? See ggmncv for the available penalties.
lambda	Numeric. Regularization parameter (defaults to 1).
gamma	Numeric vector. Hyperparameter(s) for the penalty function

Value

A list of class `penalty_function`, including the following:

- `deriv`: Data frame including the penalty function, theta, gamma, and the chosen penalty.

Note

Some care is required for specifying `gamma`. For example, the default value for `scad` is 3.7 and it *must* be some value greater than 2 (Fan and Li 2001). The default values in **GGMncv** are set to recommended values in the respective papers.

References

Fan J, Li R (2001). "Variable selection via nonconcave penalized likelihood and its oracle properties." *Journal of the American statistical Association*, **96**(456), 1348–1360.

Examples

```
func <- penalty_function(theta =  seq(-5,5,length.out = 10000),
                         lambda = 1,
                         gamma = c(0.01, 0.05, 0.1))

head(func$pen)
```

`plot.eip` *Plot Edge Inclusion 'Probabilities'*

Description

Plot Edge Inclusion 'Probabilities'

Usage

```
## S3 method for class 'eip'
plot(x, color = "black", size = 1, ...)
```

Arguments

x	An object of class eip
color	Character string. Color for geom_point.
size	Numeric. Size of geom_point.
...	Currently ignored.

Value

An object of class ggplot

Examples

```
# data
Y <- GGMncv::ptsd[,1:10]

# compute eip's
boot_samps <- boot_eip(Y, B = 10, progress = FALSE)


plot(boot_samps)
```

`plot.ggmncv` *Plot* ggmncv *Objects*

Description

Plot the solution path for the partial correlations.

Usage

```
## S3 method for class 'ggmncv'
plot(x, size = 1, alpha = 0.5, ...)
```

Arguments

x	An object of class ggmncv.
size	Numeric. Line size in geom_line.
alpha	Numeric. The transparency of the lines.
...	Currently ignored.

Value

A ggplot object.

Examples

```
# data
Y <- GGMncv::ptsd[,1:10]

# correlations
S <- cor(Y, method = "spearman")

# fit model
# default: atan
fit <- ggmncv(R = S, n = nrow(Y), progress = FALSE)

# plot
plot(fit)

# lasso
fit <- ggmncv(R = S, n = nrow(Y), progress = FALSE,
              penalty = "lasso")

# plot
plot(fit)
```

---

plot.graph *Network Plot for* select *Objects*

Description

Visualize the conditional dependence structure.

Usage

```
## S3 method for class 'graph'
plot(
  x,
  layout = "circle",
  neg_col = "#D55E00",
  pos_col = "#009E73",
  edge_magnify = 1,
  node_size = 10,
  palette = 2,
  node_names = NULL,
```

```
    node_groups = NULL,
    ...
)
```

Arguments

- `x` An object of class graph obtained from `get_graph`.
- `layout` Character string. Which graph layout (defaults is `circle`) ? See gplot.layout.
- `neg_col` Character string. Color for the positive edges (defaults to a colorblind friendly red).
- `pos_col` Character string. Color for the negative edges (defaults to a colorblind friendly green).
- `edge_magnify` Numeric. A value that is multiplied by the edge weights. This increases (> 1) or decreases (< 1) the line widths (defaults to 1).
- `node_size` Numeric. The size of the nodes (defaults to `10`).
- `palette` A character string sepcifying the palette for the `groups`. (default is `Set3`). See palette options here.
- `node_names` Character string. Names for nodes of length p.
- `node_groups` A character string of length p (the number of nodes in the model). This indicates groups of nodes that should be the same color (e.g., "clusters" or "communities").
- `...` Currently ignored.

Value

An object of class `ggplot`

Examples

```
Y <- na.omit(bfi[,1:25])

fit <- ggmncv(cor(Y), n = nrow(Y),
              progress = FALSE)

plot(get_graph(fit))
```

`plot.penalty_derivative` *Plot* `penalty_derivative` *Objects*

Description

Plot `penalty_derivative` Objects

Usage

```
## S3 method for class 'penalty_derivative'
plot(x, size = 1, ...)
```

Arguments

`x`	An object of class `penalty_derivative`.
`size`	Numeric. Line size in `geom_line`.
`...`	Currently ignored.

Value

An object of class `ggplot`

Examples

```
pen_deriv <- penalty_derivative(theta =  seq(-5,5,length.out = 10000),
                                lambda = 1,
                                gamma = c(0.01, 0.05, 0.1))
plot(pen_deriv)
```

`plot.penalty_function` *Plot* `penalty_function` *Objects*

Description

Plot `penalty_function` Objects

Usage

```
## S3 method for class 'penalty_function'
plot(x, size = 1, ...)
```

Arguments

`x`	An object of class`penalty_function`.
`size`	Numeric. Line size in `geom_line`.
`...`	Currently ignored.

Value

An object of class `ggplot`

Examples

```
func <- penalty_function(theta =  seq(-5,5,length.out = 10000),
                                lambda = 1,
                                gamma = c(0.01, 0.05, 0.1))
plot(func)
```

predict.ggmncv *Predict method for* ggmncv *Objects*

Description

There is a direct correspondence between the inverse covariance matrix and multiple regression (Stephens 1998; Kwan 2014). This readily allows for converting the off diagonal elements to regression coefficients, opening the door to out-of-sample prediction in multiple regression.

Usage

```
## S3 method for class 'ggmncv'
predict(object, train_data = NULL, newdata = NULL, ...)
```

Arguments

object	An object of class ggmncv.
train_data	Data used for model fitting (defaults to NULL).
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the fitted values are used.
...	Currently ignored.

Value

A matrix of predicted values, of dimensions rows (in the training/test data) by the number of nodes (columns).

References

Kwan CC (2014). "A regression-based interpretation of the inverse of the sample covariance matrix." *Spreadsheets in Education*, **7**(1), 4613.

Stephens G (1998). "On the Inverse of the Covariance Matrix in Portfolio Analysis." *The Journal of Finance*, **53**(5), 1821–1827.

Examples

```
# data
Y <- scale(Sachs)

# test data
Ytest <- Y[1:100,]

# training data
Ytrain <- Y[101:nrow(Y),]

fit <- ggmncv(cor(Ytrain), n = nrow(Ytrain),
              progress = FALSE)

pred <- predict(fit, newdata = Ytest)

round(apply((pred - Ytest)^2, 2, mean), 2)
```

`print.eip` *Print* `eip` *Objects*

Description

Print `eip` Objects

Usage

```
## S3 method for class 'eip'
print(x, ...)
```

Arguments

`x`	An object of class `eip`
`...`	Currently ignored.

`print.ggmncv` *Print* `ggmncv` *Objects*

Description

Print ggmncv Objects

Usage

```
## S3 method for class 'ggmncv'
print(x, ...)
```

Arguments

`x`	An object of class ggmncv
`...`	Currently ignored

`print.nct` *Print* `nct` *Objects*

Description

Print `nct` Objects

Usage

```
## S3 method for class 'nct'
print(x, ...)
```

Arguments

`x`	An object of class `nct`
`...`	Currently ignored.

`ptsd` *Data: Post-Traumatic Stress Disorder*

Description

A dataset containing items that measure Post-traumatic stress disorder symptoms (Armour et al. 2017). There are 20 variables (p) and 221 observations (n).

Usage

```
data("ptsd")
```

Format

A dataframe with 221 rows and 20 variables

Details

- Intrusive Thoughts
- Nightmares
- Flashbacks
- Emotional cue reactivity
- Psychological cue reactivity
- Avoidance of thoughts
- Avoidance of reminders
- Trauma-related amnesia
- Negative beliefs
- Negative trauma-related emotions
- Loss of interest
- Detachment
- Restricted affect
- Irritability/anger
- Self-destructive/reckless behavior
- Hypervigilance
- Exaggerated startle response
- Difficulty concentrating
- Sleep disturbance

References

Armour C, Fried EI, Deserno MK, Tsai J, Pietrzak RH (2017). "A network analysis of DSM-5 posttraumatic stress disorder symptoms and correlates in US military veterans." *Journal of anxiety disorders*, **45**, 49–59.

`Sachs` *Data: Sachs Network*

Description

Protein expression in human immune system cells

Usage

```
data("Sachs")
```

Format

A data frame containing 7466 cells (n = 7466) and flow cytometry measurements of 11 (p = 11) phosphorylated proteins and phospholipids (Sachs et al. 2002)

References

Sachs K, Gifford D, Jaakkola T, Sorger P, Lauffenburger DA (2002). "Bayesian network approach to cell signaling pathway modeling." *Science's STKE*, **2002**(148), pe38–pe38.

Examples

```
data("Sachs")
```

`score_binary` *Binary Classification*

Description

Binary Classification

Usage

```
score_binary(estimate, true, model_name = NULL)
```

Arguments

`estimate`	Matrix. Estimated graph (adjacency matrix)
`true`	Matrix. True graph (adjacency matrix)
`model_name`	Character string. Name of the method or penalty (defaults to `NULL`)

Value

A data frame containing specificity (1 - false positive rate), sensitivity (true positive rate), precision (1 - false discovery rate), f1_score, and mcc (Matthews correlation coefficient).

Examples

```
p <- 20
n <- 500

true_net <- gen_net(p = p, edge_prob = 0.25)

y <- MASS::mvrnorm(n = n,
                   mu = rep(0, p),
                   Sigma = true_net$cors)

# default
fit_atan <- ggmncv(R = cor(y),
                   n = nrow(y),
                   penalty = "atan",
                   progress = FALSE)

# lasso
fit_l1 <- ggmncv(R = cor(y),
                 n = nrow(y),
                 penalty = "lasso",
                 progress = FALSE)

# atan scores
score_binary(estimate = true_net$adj,
             true = fit_atan$adj,
             model_name = "atan")

score_binary(estimate = fit_l1$adj,
             true = true_net$adj,
             model_name = "lasso")
```

Index