

Package ‘lessSEM’

October 18, 2025

Type Package

Title Non-Smooth Regularization for Structural Equation Models

Version 1.5.7

Maintainer Jannik H. Orzek <jannik.orzek@mailbox.org>

Description Provides regularized structural equation modeling (regularized SEM) with non-smooth penalty functions (e.g., lasso) building on 'lavaan'. The package is heavily inspired by the ['regsem'](<<https://github.com/Rjacobucci/regsem>>) and ['lslx'](<<https://github.com/psyphh/lslx>>) packages.

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0), lavaan, methods

Imports Rcpp (>= 1.0.8), RcppArmadillo, RcppParallel, ggplot2, ggrepel, tidyr, stringr, numDeriv, utils, stats, graphics, rlang, mvtnorm

Suggests knitr, plotly, rmarkdown, Rsolnp

LinkingTo Rcpp, RcppArmadillo, RcppParallel

VignetteBuilder knitr

SystemRequirements GNU make, C++17

URL <https://github.com/jhorzek/lessSEM>,
<https://jhorzek.github.io/lessSEM/>

BugReports <https://github.com/jhorzek/lessSEM/issues>

NeedsCompilation yes

Author Jannik H. Orzek [aut, cre, cph] (ORCID:
<<https://orcid.org/0000-0002-3123-2248>>)

Repository CRAN

Date/Publication 2025-10-18 13:20:02 UTC

Contents

.adaptBreakingForWls	5
.checkPenalties	6
.labelLavaanParameters	6
.updateLavaan	7
.useElasticNet	7
adaptiveLasso	8
addCappedL1	11
addElasticNet	13
addLasso	15
addLsp	17
addMcp	19
addScad	21
AIC,gpRegularized-method	23
AIC,Rcpp_mgSEM-method	24
AIC,Rcpp_SEMCpp-method	24
AIC,regularizedSEM-method	25
AIC,regularizedSEMMixedPenalty-method	25
bfgs	26
bfgsEnet	27
bfgsEnetMgSEM	27
bfgsEnetSEM	28
BIC,gpRegularized-method	28
BIC,Rcpp_mgSEM-method	29
BIC,Rcpp_SEMCpp-method	29
BIC,regularizedSEM-method	30
BIC,regularizedSEMMixedPenalty-method	30
callFitFunction	31
cappedL1	31
coef,cvRegularizedSEM-method	34
coef,gpRegularized-method	34
coef,Rcpp_mgSEM-method	35
coef,Rcpp_SEMCpp-method	35
coef,regularizedSEM-method	36
coef,regularizedSEMMixedPenalty-method	36
controlBFGS	37
controlGlmnet	39
controlIsta	40
covariances	42
createSubsets	43
curveLambda	43
cvAdaptiveLasso	44
cvCappedL1	47
cvElasticNet	50
cvLasso	53
cvLsp	55
cvMcp	58

cvRegularizedSEM-class	61
cvRidge	62
cvRidgeBfgs	64
cvScad	67
cvScaler	69
cvSmoothAdaptiveLasso	70
cvSmoothElasticNet	72
cvSmoothLasso	75
elasticNet	77
estimates	80
estimates,cvRegularizedSEM-method	81
estimates,regularizedSEM-method	81
estimates,regularizedSEMMixedPenalty-method	82
fit	82
fitIndices	83
fitIndices,cvRegularizedSEM-method	84
fitIndices,regularizedSEM-method	84
fitIndices,regularizedSEMMixedPenalty-method	85
getLavaanParameters	85
getTuningParameterConfiguration	86
glmnetCappedL1MgSEM	87
glmnetCappedL1SEM	88
glmnetEnetGeneralPurpose	88
glmnetEnetGeneralPurposeCpp	89
glmnetEnetMgSEM	89
glmnetEnetSEM	90
glmnetLspMgSEM	90
glmnetLspSEM	91
glmnetMcpMgSEM	91
glmnetMcpSEM	92
glmnetMixedMgSEM	92
glmnetMixedPenaltyGeneralPurpose	93
glmnetMixedPenaltyGeneralPurposeCpp	93
glmnetMixedSEM	94
glmnetScadMgSEM	94
glmnetScadSEM	95
gpAdaptiveLasso	95
gpAdaptiveLassoCpp	98
gpCappedL1	102
gpCappedL1Cpp	106
gpElasticNet	109
gpElasticNetCpp	112
gpLasso	116
gpLassoCpp	119
gpLsp	123
gpLspCpp	126
gpMcp	130
gpMcpCpp	132

gpRegularized-class	136
gpRidge	137
gpRidgeCpp	140
gpScad	143
gpScadCpp	146
istaCappedL1mgSEM	150
istaCappedL1SEM	150
istaEnetGeneralPurpose	151
istaEnetGeneralPurposeCpp	151
istaEnetMgSEM	152
istaEnetSEM	152
istaLSPMgSEM	153
istaLSPSEM	153
istaMcpMgSEM	154
istaMcpSEM	154
istaMixedPenaltyGeneralPurpose	155
istaMixedPenaltyGeneralPurposeCpp	155
istaMixedPenaltymgSEM	156
istaMixedPenaltySEM	156
istaScadMgSEM	157
istaScadSEM	157
lasso	158
lavaan2lsxLabels	161
lessSEM2Lavaan	162
lessSEMCoef-class	163
loadings	163
logicalMatch	164
logLik,Rcpp_mgSEM-method	165
logLik,Rcpp_SEMCpp-method	165
logLikelihood-class	166
lsp	166
makePtrs	169
mcp	170
mcpPenalty_C	172
mgSEM	173
mixedPenalty	174
modifyModel	176
newTau	177
plot,cvRegularizedSEM,missing-method	179
plot,gpRegularized,missing-method	179
plot,regularizedSEM,missing-method	180
plot,stabSel,missing-method	180
regressions	181
regsem2LavaanParameters	182
regularizedSEM-class	183
regularizedSEMMixedPenalty-class	183
ridge	184
ridgeBfgs	186

scad	188
scadPenalty_C	191
SEMCpp	191
show,cvRegularizedSEM-method	192
show,gpRegularized-method	192
show,lessSEMCoeff-method	193
show,logLikelihood-method	193
show,Rcpp_mgSEM-method	194
show,Rcpp_SEMCpp-method	194
show,regularizedSEM-method	195
show,regularizedSEMMixedPenalty-method	195
show,stabSel-method	196
simulateExampleData	196
smoothAdaptiveLasso	197
smoothElasticNet	199
smoothLasso	201
stabilitySelection	203
stabSel-class	205
summary,cvRegularizedSEM-method	206
summary,gpRegularized-method	206
summary,regularizedSEM-method	207
summary,regularizedSEMMixedPenalty-method	207
variances	208

Index 209

.adaptBreakingForWls *.adaptBreakingForWls*

Description

wls needs smaller breaking points than ml

Usage

```
.adaptBreakingForWls(lavaanModel, currentBreaking, selectedDefault)
```

Arguments

<i>lavaanModel</i>	single model or vector of models
<i>currentBreaking</i>	current breaking condition value
<i>selectedDefault</i>	was default breaking condition selected?

Value

updated breaking

<code>.checkPenalties</code>	<i><code>.checkPenalties</code></i>
------------------------------	-------------------------------------

Description

Internal function to check a mixedPenalty object

Usage

`.checkPenalties(mixedPenalty)`

Arguments

`mixedPenalty` object of class `mixedPenalty`. This object can be created with the `mixedPenalty` function. Penalties can be added with the `addCappedL1`, `addLasso`, `addLsp`, `addMcp`, and `addScad` functions.

<code>.labelLavaanParameters</code>	<i><code>.labelLavaanParameters</code></i>
-------------------------------------	--

Description

Adds labels to unlabeled parameters in the lavaan parameter table. Also removes fixed parameters.

Usage

`.labelLavaanParameters(lavaanModel)`

Arguments

`lavaanModel` fitted lavaan model

Value

parameterTable with labeled parameters

.updateLavaan

.updateLavaan

Description

updates a lavaan model. lavaan has an update function that does exactly that, but it seems to not work with testthat. This is an attempt to hack around the issue...

Usage

```
.updateLavaan(lavaanModel, key, value)
```

Arguments

lavaanModel	fitted lavaan model
key	label of the element that should be updated
value	new value for the updated element

Value

lavaan model

.useElasticNet

.useElasticNet

Description

Internal function checking if elastic net is used

Usage

```
.useElasticNet(mixedPenalty)
```

Arguments

mixedPenalty	object of class mixedPenalty. This object can be created with the mixedPenalty function. Penalties can be added with the addCappedL1, addLasso, addLsp, addMcp, and addScad functions.
--------------	--

Value

TRUE if elastic net, FALSE otherwise

adaptiveLasso

*adaptiveLasso***Description**

Implements adaptive lasso regularization for structural equation models. The penalty function is given by:

$$p(x_j) = p(x_j) = \frac{1}{w_j} \lambda |x_j|$$

Adaptive lasso regularization will set parameters to zero if λ is large enough.

Usage

```
adaptiveLasso(
  lavaanModel,
  regularized,
  weights = NULL,
  lambdas = NULL,
  nLambdas = NULL,
  reverse = TRUE,
  curve = 1,
  method = "glmnet",
  modifyModel = lessSEM::modifyModel(),
  control = lessSEM::controlGlmnet()
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
weights	labeled vector with weights for each of the parameters in the model. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object. If set to NULL, the default weights will be used: the inverse of the absolute values of the unregularized parameter estimates
lambdas	numeric vector: values for the tuning parameter lambda
nLambdas	alternative to lambda: If $\alpha = 1$, lessSEM can automatically compute the first lambda value which sets all regularized parameters to zero. It will then generate nLambda values between 0 and the computed lambda.
reverse	if set to TRUE and nLambdas is used, lessSEM will start with the largest lambda and gradually decrease lambda. Otherwise, lessSEM will start with the smallest lambda and gradually increase it.

curve	Allows for unequally spaced lambda steps (e.g., .01,.02,.05,1,5,20). If curve is close to 1 all lambda values will be equally spaced, if curve is large lambda values will be more concentrated close to 0. See ?lessSEM::curveLambda for more information.
method	which optimizer should be used? Currently implemented are ista and glmnet. With ista, the control argument can be used to switch to related procedures (currently gist).
modifyModel	used to modify the lavaanModel. See ?modifyModel.
control	used to control the optimizer. This element is generated with the controlIsta and controlGlmnet functions. See ?controlIsta and ?controlGlmnet for more details.

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

Adaptive lasso regularization:

- Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476), 1418–1429. <https://doi.org/10.1198/016214506000000735>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1198016>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Model of class regularizedSEM

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ l1*y1 + l2*y2 + l3*y3 + l4*y4 + l5*y5 +
      l6*y6 + l7*y7 + l8*y8 + l9*y9 + l10*y10 +
      l11*y11 + l12*y12 + l13*y13 + l14*y14 + l15*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- adaptiveLasso(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("l", 6:15),
  # in case of lasso and adaptive lasso, we can specify the number of lambda
  # values to use. lessSEM will automatically find lambda_max and fit
  # models for nLambda values between 0 and lambda_max. For the other
  # penalty functions, lambdas must be specified explicitly
  nLambdas = 50)

# use the plot-function to plot the regularized parameters:
plot(lsem)

# the coefficients can be accessed with:
coef(lsem)
# if you are only interested in the estimates and not the tuning parameters, use
coef(lsem)$estimates
# or
estimates(lsem)

# elements of lsem can be accessed with the @ operator:
lsem$parameters[1,]

# fit Measures:
```

```

fitIndices(lsem)

# The best parameters can also be extracted with:
coef(lsem, criterion = "AIC")
# or
estimates(lsem, criterion = "AIC")

#### Advanced ####
# Switching the optimizer #
# Use the "method" argument to switch the optimizer. The control argument
# must also be changed to the corresponding function:
lsemIsta <- adaptiveLasso(
  lavaanModel = lavaanModel,
  regularized = paste0("l", 6:15),
  nLambdas = 50,
  method = "ista",
  control = controlIsta())

# Note: The results are basically identical:
lsemIsta@parameters - lsem@parameters

```

addCappedL1

addCappedL1

Description

Implements cappedL1 regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \lambda \min(|x_j|, \theta)$$

where $\theta > 0$. The cappedL1 penalty is identical to the lasso for parameters which are below θ and identical to a constant for parameters above θ . As adding a constant to the fitting function will not change its minimum, larger parameters can stay unregularized while smaller ones are set to zero.

Usage

```
addCappedL1(mixedPenalty, regularized, lambdas, thetas)
```

Arguments

mixedPenalty	model of class mixedPenalty created with the mixedPenalty function (see ?mixed-Penalty)
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use getLavaanParameters(model) with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
thetas	parameters whose absolute value is above this threshold will be penalized with a constant (theta)

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

CappedL1 regularization:

- Zhang, T. (2010). Analysis of Multi-stage Convex Relaxation for Sparse Regularization. *Journal of Machine Learning Research*, 11, 1081–1107.

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Model of class `mixedPenalty`. Use the `fit()` - function to fit the model

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
```

```

                                data = dataset,
                                meanstructure = TRUE,
                                std.lv = TRUE)

# We can add mixed penalties as follows:

regularized <- lavaanModel |>
  # create template for regularized model with mixed penalty:
  mixedPenalty() |>
  # add penalty on loadings l6 - l10:
  addCappedL1(regularized = paste0("l", 11:15),
              lambdas = seq(0,1,.1),
              thetas = 2.3) |>
  # fit the model:
  fit()

```

addElasticNet

addElasticNet

Description

Adds an elastic net penalty to specified parameters. The penalty function is given by:

$$p(x_j) = \alpha \lambda |x_j| + (1 - \alpha) \lambda x_j^2$$

Note that the elastic net combines ridge and lasso regularization. If $\alpha = 0$, the elastic net reduces to ridge regularization. If $\alpha = 1$ it reduces to lasso regularization. In between, elastic net is a compromise between the shrinkage of the lasso and the ridge penalty.

Usage

```
addElasticNet(mixedPenalty, regularized, alphas, lambdas, weights = 1)
```

Arguments

<code>mixedPenalty</code>	model of class <code>mixedPenalty</code> created with the <code>mixedPenalty</code> function (see <code>?mixedPenalty</code>)
<code>regularized</code>	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
<code>alphas</code>	numeric vector: values for the tuning parameter alpha. Set to 1 for lasso and to zero for ridge. Anything in between is an elastic net penalty.
<code>lambdas</code>	numeric vector: values for the tuning parameter lambda
<code>weights</code>	can be used to give different weights to the different parameters

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

Elastic net regularization:

- Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2), 301–320. <https://doi.org/10.1111/j.1467-9868.2005.00503.x>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Model of class `mixedPenalty`. Use the `fit()` - function to fit the model

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
```

```
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# We can add mixed penalties as follows:

regularized <- lavaanModel |>
  # create template for regularized model with mixed penalty:
  mixedPenalty() |>
  # add penalty on loadings 16 - 110:
  addElasticNet(regularized = paste0("1", 11:15),
                lambdas = seq(0,1,.1),
                alphas = .4) |>
  # fit the model:
  fit()
```

addLasso

addLasso

Description

Implements lasso regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \lambda |x_j|$$

Lasso regularization will set parameters to zero if λ is large enough

Usage

```
addLasso(mixedPenalty, regularized, weights = 1, lambdas)
```

Arguments

<code>mixedPenalty</code>	model of class <code>mixedPenalty</code> created with the <code>mixedPenalty</code> function (see <code>?mixedPenalty</code>)
<code>regularized</code>	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object

weights	can be used to give different weights to the different parameters
lambdas	numeric vector: values for the tuning parameter lambda

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

Lasso regularization:

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

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Model of class `mixedPenalty`. Use the `fit()` - function to fit the model

Examples

```

library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# We can add mixed penalties as follows:

regularized <- lavaanModel |>
  # create template for regularized model with mixed penalty:
  mixedPenalty() |>
  # add penalty on loadings 16 - 110:
  addLasso(regularized = paste0("1", 11:15),
           lambdas = seq(0,1,.1)) |>
  # fit the model:
  fit()

```

addLsp

addLsp

Description

Implements lsp regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \lambda \log(1 + |x_j|/\theta)$$

where $\theta > 0$.

Usage

```
addLsp(mixedPenalty, regularized, lambdas, thetas)
```

Arguments

<code>mixedPenalty</code>	model of class <code>mixedPenalty</code> created with the <code>mixedPenalty</code> function (see <code>?mixedPenalty</code>)
<code>regularized</code>	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
<code>lambdas</code>	numeric vector: values for the tuning parameter <code>lambda</code>
<code>thetas</code>	parameters whose absolute value is above this threshold will be penalized with a constant (<code>theta</code>)

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

lsp regularization:

- Candès, E. J., Wakin, M. B., & Boyd, S. P. (2008). Enhancing Sparsity by Reweighted l1 Minimization. *Journal of Fourier Analysis and Applications*, 14(5–6), 877–905. <https://doi.org/10.1007/s00041-008-9045-x>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Model of class mixedPenalty. Use the fit() - function to fit the model

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f ~~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# We can add mixed penalties as follows:

regularized <- lavaanModel |>
  # create template for regularized model with mixed penalty:
  mixedPenalty() |>
  # add penalty on loadings 16 - 110:
  addLsp(regularized = paste0("1", 11:15),
         lambdas = seq(0,1,.1),
         thetas = 2.3) |>
  # fit the model:
  fit()
```

addMcp

addMcp

Description

Implements mcp regularization for structural equation models. The penalty function is given by:
Equation Omitted in Pdf Documentation.

Usage

```
addMcp(mixedPenalty, regularized, lambdas, thetas)
```

Arguments

<code>mixedPenalty</code>	model of class <code>mixedPenalty</code> created with the <code>mixedPenalty</code> function (see <code>?mixedPenalty</code>)
<code>regularized</code>	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
<code>lambdas</code>	numeric vector: values for the tuning parameter <code>lambda</code>
<code>thetas</code>	parameters whose absolute value is above this threshold will be penalized with a constant (<code>theta</code>)

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

mcp regularization:

- Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. The Annals of Statistics, 38(2), 894–942. <https://doi.org/10.1214/09-AOS729>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. Psychometrika, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. Structural Equation Modeling: A Multidisciplinary Journal, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. SIAM Journal on Imaging Sciences, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. Proceedings of the 30th International Conference on Machine Learning, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. Foundations and Trends in Optimization, 1(3), 123–231.

Value

Model of class `mixedPenalty`. Use the `fit()` - function to fit the model

Examples

```

library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f ~~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# We can add mixed penalties as follows:

regularized <- lavaanModel |>
  # create template for regularized model with mixed penalty:
  mixedPenalty() |>
  # add penalty on loadings l6 - l10:
  addMcp(regularized = paste0("l", 11:15),
         lambdas = seq(0,1,.1),
         thetas = 2.3) |>
  # fit the model:
  fit()

```

addScad

addScad

Description

Implements scad regularization for structural equation models. The penalty function is given by:
Equation Omitted in Pdf Documentation.

Usage

```
addScad(mixedPenalty, regularized, lambdas, thetas)
```

Arguments

<code>mixedPenalty</code>	model of class <code>mixedPenalty</code> created with the <code>mixedPenalty</code> function (see <code>?mixed-Penalty</code>)
---------------------------	---

regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
thetas	parameters whose absolute value is above this threshold will be penalized with a constant (theta)

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

scad regularization:

- 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. <https://doi.org/10.1198/01621450175>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1198016>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Model of class `mixedPenalty`. Use the `fit()` - function to fit the model

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()
```

```

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# We can add mixed penalties as follows:

regularized <- lavaanModel |>
  # create template for regularized model with mixed penalty:
  mixedPenalty() |>
  # add penalty on loadings 16 - 110:
  addScad(regularized = paste0("1", 11:15),
          lambdas = seq(0,1,.1),
          thetas = 3.1) |>
  # fit the model:
  fit()

```

AIC, gpRegularized-method
AIC

Description

returns the AIC

Usage

```
## S4 method for signature 'gpRegularized'
AIC(object, ..., k = 2)
```

Arguments

object	object of class gpRegularized
...	not used
k	multiplier for number of parameters

Value

data frame with fit values, appended with AIC

AIC,Rcpp_mgSEM-method *AIC*

Description

AIC

Usage

```
## S4 method for signature 'Rcpp_mgSEM'
AIC(object, ..., k = 2)
```

Arguments

object	object of class Rcpp_mgSEM
...	not used
k	multiplier for number of parameters

Value

AIC values

AIC,Rcpp_SEMCpp-method
AIC

Description

AIC

Usage

```
## S4 method for signature 'Rcpp_SEMCpp'
AIC(object, ..., k = 2)
```

Arguments

object	object of class Rcpp_SEMCpp
...	not used
k	multiplier for number of parameters

Value

AIC values

AIC,regularizedSEM-method
AIC

Description

returns the AIC

Usage

```
## S4 method for signature 'regularizedSEM'  
AIC(object, ..., k = 2)
```

Arguments

object	object of class regularizedSEM
...	not used
k	multiplier for number of parameters

Value

AIC values

AIC,regularizedSEMMixedPenalty-method
AIC

Description

returns the AIC

Usage

```
## S4 method for signature 'regularizedSEMMixedPenalty'  
AIC(object, ..., k = 2)
```

Arguments

object	object of class regularizedSEMMixedPenalty
...	not used
k	multiplier for number of parameters

Value

AIC values

bfgs

*bfgs***Description**

This function allows for optimizing models built in lavaan using the BFGS optimizer implemented in lessSEM. Its elements can be accessed with the "@" operator (see examples). The main purpose is to make transformations of lavaan models more accessible.

Usage

```
bfgs(
  lavaanModel,
  modifyModel = lessSEM::modifyModel(),
  control = lessSEM::controlBFGS()
)
```

Arguments

lavaanModel	model of class lavaan
modifyModel	used to modify the lavaanModel. See ?modifyModel.
control	used to control the optimizer. See ?controlBFGS for more details.

Value

Model of class regularizedSEM

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)
```

```
lsem <- bfgs(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel)

# the coefficients can be accessed with:
coef(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters
```

bfgsEnet

smoothly approximated elastic net

Description

Object for smoothly approximated elastic net optimization with bfgs optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a lambda and an alpha value.

bfgsEnetMgSEM

smoothly approximated elastic net

Description

Object for smoothly approximated elastic net optimization with bfgs optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type `SEM_Cpp`, a `lambda` and an `alpha` value.

bfgsEnetSEM
smoothly approximated elastic net

Description

Object for smoothly approximated elastic net optimization with bfgs optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type `SEM_Cpp`, a `lambda` and an `alpha` value.

BIC, gpRegularized-method
BIC

Description

returns the BIC

Usage

```
## S4 method for signature 'gpRegularized'
BIC(object, ...)
```

Arguments

<code>object</code>	object of class <code>gpRegularized</code>
<code>...</code>	not used

Value

data frame with fit values, appended with BIC

BIC,Rcpp_mgSEM-method *BIC*

Description

BIC

Usage

```
## S4 method for signature 'Rcpp_mgSEM'
BIC(object, ...)
```

Arguments

object	object of class Rcpp_mgSEM
...	not used

Value

BIC values

BIC,Rcpp_SEMCpp-method
BIC

Description

BIC

Usage

```
## S4 method for signature 'Rcpp_SEMCpp'
BIC(object, ...)
```

Arguments

object	object of class Rcpp_SEMCpp
...	not used

Value

BIC values

BIC,regularizedSEM-method
<i>BIC</i>

Description

returns the BIC

Usage

```
## S4 method for signature 'regularizedSEM'  
BIC(object, ...)
```

Arguments

object	object of class regularizedSEM
...	not used

Value

BIC values

BIC,regularizedSEMMixedPenalty-method
<i>BIC</i>

Description

returns the BIC

Usage

```
## S4 method for signature 'regularizedSEMMixedPenalty'  
BIC(object, ...)
```

Arguments

object	object of class regularizedSEMMixedPenalty
...	not used

Value

BIC values

callFitFunction	<i>callFitFunction</i>
-----------------	------------------------

Description

wrapper to call user defined fit function

Usage

```
callFitFunction(fitFunctionSEXP, parameters, userSuppliedElements)
```

Arguments

fitFunctionSEXP	pointer to fit function
parameters	vector with parameter values
userSuppliedElements	list with additional elements

Value

fit value (double)

cappedL1	<i>cappedL1</i>
----------	-----------------

Description

Implements cappedL1 regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \lambda \min(|x_j|, \theta)$$

where $\theta > 0$. The cappedL1 penalty is identical to the lasso for parameters which are below θ and identical to a constant for parameters above θ . As adding a constant to the fitting function will not change its minimum, larger parameters can stay unregularized while smaller ones are set to zero.

Usage

```
cappedL1(
  lavaanModel,
  regularized,
  lambdas,
  thetas,
  modifyModel = lessSEM::modifyModel(),
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
thetas	parameters whose absolute value is above this threshold will be penalized with a constant (theta)
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
method	which optimizer should be used? Currently implemented are <code>ista</code> and <code>glmnet</code> . With <code>ista</code> , the control argument can be used to switch to related procedures
control	used to control the optimizer. This element is generated with the <code>controlIsta</code> (see <code>?controlIsta</code>)

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

CappedL1 regularization:

- Zhang, T. (2010). Analysis of Multi-stage Convex Relaxation for Sparse Regularization. *Journal of Machine Learning Research*, 11, 1081–1107.

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>

- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Model of class `regularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- cappedL1(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("1", 6:15),
  lambdas = seq(0,1,length.out = 20),
  thetas = seq(0.01,2,length.out = 5))

# the coefficients can be accessed with:
coef(lsem)
# if you are only interested in the estimates and not the tuning parameters, use
coef(lsem)$estimates
# or
estimates(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters[1,]
```

```
# fit Measures:
fitIndices(lsem)

# The best parameters can also be extracted with:
coef(lsem, criterion = "AIC")
# or
estimates(lsem, criterion = "AIC")

# optional: plotting the paths requires installation of plotly
# plot(lsem)
```

```
coef, cvRegularizedSEM-method
      coef
```

Description

Returns the parameter estimates of an `cvRegularizedSEM`

Usage

```
## S4 method for signature 'cvRegularizedSEM'
coef(object, ...)
```

Arguments

<code>object</code>	object of class <code>cvRegularizedSEM</code>
<code>...</code>	not used

Value

the parameter estimates of an `cvRegularizedSEM`

```
coef, gpRegularized-method
      coef
```

Description

Returns the parameter estimates of a `gpRegularized`

Usage

```
## S4 method for signature 'gpRegularized'
coef(object, ...)
```

Arguments

object	object of class gpRegularized
...	criterion can be one of: "AIC", "BIC". If set to NULL, all parameters will be returned

Value

parameter estimates

coef,Rcpp_mgSEM-method

coef

Description

coef

Usage

```
## S4 method for signature 'Rcpp_mgSEM'  
coef(object, ...)
```

Arguments

object	object of class Rcpp_mgSEM
...	not used

Value

all coefficients of the model in transformed form

coef,Rcpp_SEMCpp-method

coef

Description

coef

Usage

```
## S4 method for signature 'Rcpp_SEMCpp'  
coef(object, ...)
```

Arguments

object	object of class Rcpp_SEMCpp
...	not used

Value

all coefficients of the model in transformed form

coef,regularizedSEM-method
coef

Description

Returns the parameter estimates of a regularizedSEM

Usage

```
## S4 method for signature 'regularizedSEM'
coef(object, ...)
```

Arguments

object	object of class regularizedSEM
...	criterion can be one of the ones returned by fitIndices. If set to NULL, all parameters will be returned

Value

parameters of the model as data.frame

coef,regularizedSEMMixedPenalty-method
coef

Description

Returns the parameter estimates of a regularizedSEMMixedPenalty

Usage

```
## S4 method for signature 'regularizedSEMMixedPenalty'
coef(object, ...)
```

Arguments

object	object of class regularizedSEMMixedPenalty
...	criterion can be one of: "AIC", "BIC". If set to NULL, all parameters will be returned

Value

parameters of the model as data.frame

controlBFGS

controlBFGS

Description

Control the BFGS optimizer.

Usage

```
controlBFGS(
  startingValues = "est",
  initialHessian = ifelse(all(startingValues == "est"), "lavaan", "compute"),
  saveDetails = FALSE,
  stepSize = 0.9,
  sigma = 1e-05,
  gamma = 0,
  maxIterOut = 1000,
  maxIterIn = 1000,
  maxIterLine = 500,
  breakOuter = 1e-08,
  breakInner = 1e-10,
  convergenceCriterion = 0,
  verbose = 0,
  nCores = 1
)
```

Arguments

startingValues	option to provide initial starting values. Only used for the first lambda. Three options are supported. Setting to "est" will use the estimates from the lavaan model object. Setting to "start" will use the starting values of the lavaan model. Finally, a labeled vector with parameter values can be passed to the function which will then be used as starting values.
initialHessian	option to provide an initial Hessian to the optimizer. Must have row and column names corresponding to the parameter labels. use getLavaanParameters(lavaanModel) to see those labels. If set to "gradNorm", the maximum of the gradients at the starting values times the stepSize will be used. This is adapted from Optim.jl

	https://github.com/JuliaNLSolvers/Optim.jl/blob/f43e6084aacf2dabb2b142952acd3fbb0e268439/src/mu If set to a single value, a diagonal matrix with the single value along the diagonal will be used. The default is "lavaan" which extracts the Hessian from the lavaanModel. This Hessian will typically deviate from that of the internal SEM representation of lessSEM (due to the transformation of the variances), but works quite well in practice.
saveDetails	when set to TRUE, additional details about the individual models are save. Currently, this are the Hessian and the implied means and covariances. Note: This may take a lot of memory!
stepSize	Initial stepSize of the outer iteration ($\theta_{next} = \theta_{previous} + stepSize * Stepdirection$)
sigma	only relevant when lineSearch = 'GLMNET'. Controls the sigma parameter in Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. The Journal of Machine Learning Research, 13, 1999–2030. https://doi.org/10.1145/2020408.2020421 .
gamma	Controls the gamma parameter in Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. The Journal of Machine Learning Research, 13, 1999–2030. https://doi.org/10.1145/2020408.2020421 . Defaults to 0.
maxIterOut	Maximal number of outer iterations
maxIterIn	Maximal number of inner iterations
maxIterLine	Maximal number of iterations for the line search procedure
breakOuter	Stopping criterion for outer iterations
breakInner	Stopping criterion for inner iterations
convergenceCriterion	which convergence criterion should be used for the outer iterations? possible are 0 = GLMNET, 1 = fitChange, 2 = gradients. Note that in case of gradients and GLMNET, we divide the gradients (and the Hessian) of the log-Likelihood by N as it would otherwise be considerably more difficult for larger sample sizes to reach the convergence criteria.
verbose	0 prints no additional information, > 0 prints GLMNET iterations
nCores	number of core to use. Multi-core support is provided by RcppParallel and only supported for SEM, not for general purpose optimization.

Value

object of class controlBFGS

Examples

```
control <- controlBFGS()
```

controlGlmnet

*controlGlmnet***Description**

Control the GLMNET optimizer.

Usage

```
controlGlmnet(
  startingValues = "est",
  initialHessian = ifelse(all(startingValues == "est"), "lavaan", "compute"),
  saveDetails = FALSE,
  stepSize = 0.9,
  sigma = 1e-05,
  gamma = 0,
  maxIterOut = 1000,
  maxIterIn = 1000,
  maxIterLine = 500,
  breakOuter = 1e-08,
  breakInner = 1e-10,
  convergenceCriterion = 0,
  verbose = 0,
  nCores = 1
)
```

Arguments

startingValues option to provide initial starting values. Only used for the first lambda. Three options are supported. Setting to "est" will use the estimates from the lavaan model object. Setting to "start" will use the starting values of the lavaan model. Finally, a labeled vector with parameter values can be passed to the function which will then be used as starting values.

initialHessian option to provide an initial Hessian to the optimizer. Must have row and column names corresponding to the parameter labels. use `getLavaanParameters(lavaanModel)` to see those labels. If set to "gradNorm", the maximum of the gradients at the starting values times the stepSize will be used. This is adapted from `Optim.jl` <https://github.com/JuliaNLSolvers/Optim.jl/blob/f43e6084aacf2dabb2b142952acd3fbb0e268439/src/mu>. If set to "compute", the initial hessian will be computed. If set to a single value, a diagonal matrix with the single value along the diagonal will be used. The default is "lavaan" which extracts the Hessian from the lavaanModel. This Hessian will typically deviate from that of the internal SEM representation of lessSEM (due to the transformation of the variances), but works quite well in practice.

saveDetails when set to TRUE, additional details about the individual models are save. Currently, this are the Hessian and the implied means and covariances. Note: This may take a lot of memory!

stepSize	Initial stepSize of the outer iteration ($\theta_{\text{next}} = \theta_{\text{previous}} + \text{stepSize} * \text{Stepdirection}$)
sigma	only relevant when lineSearch = 'GLMNET'. Controls the sigma parameter in Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. The Journal of Machine Learning Research, 13, 1999–2030. https://doi.org/10.1145/2020408.2020421 .
gamma	Controls the gamma parameter in Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. The Journal of Machine Learning Research, 13, 1999–2030. https://doi.org/10.1145/2020408.2020421 . Defaults to 0.
maxIterOut	Maximal number of outer iterations
maxIterIn	Maximal number of inner iterations
maxIterLine	Maximal number of iterations for the line search procedure
breakOuter	Stopping criterion for outer iterations
breakInner	Stopping criterion for inner iterations
convergenceCriterion	which convergence criterion should be used for the outer iterations? possible are 0 = GLMNET, 1 = fitChange, 2 = gradients. Note that in case of gradients and GLMNET, we divide the gradients (and the Hessian) of the log-Likelihood by N as it would otherwise be considerably more difficult for larger sample sizes to reach the convergence criteria.
verbose	0 prints no additional information, > 0 prints GLMNET iterations
nCores	number of core to use. Multi-core support is provided by RcppParallel and only supported for SEM, not for general purpose optimization.

Value

object of class controlGlmnet

Examples

```
control <- controlGlmnet()
```

controlIsta

controlIsta

Description

controlIsta

Usage

```
controlIsta(
  startingValues = "est",
  saveDetails = FALSE,
  L0 = 0.1,
  eta = 2,
  accelerate = TRUE,
  maxIterOut = 10000,
  maxIterIn = 1000,
  breakOuter = 1e-08,
  convCritInner = 1,
  sigma = 0.1,
  stepSizeInheritance = ifelse(accelerate, 1, 3),
  verbose = 0,
  nCores = 1
)
```

Arguments

startingValues	option to provide initial starting values. Only used for the first lambda. Three options are supported. Setting to "est" will use the estimates from the lavaan model object. Setting to "start" will use the starting values of the lavaan model. Finally, a labeled vector with parameter values can be passed to the function which will then be used as starting values.
saveDetails	when set to TRUE, additional details about the individual models are save. Currently, this are the implied means and covariances. Note: This may take a lot of memory!
L0	L0 controls the step size used in the first iteration
eta	eta controls by how much the step size changes in the inner iterations with $(\eta^i) * L$, where i is the inner iteration
accelerate	boolean: Should the acceleration outlined in Parikh, N., & Boyd, S. (2013). Proximal Algorithms. Foundations and Trends in Optimization, 1(3), 123–231., p. 152 be used?
maxIterOut	maximal number of outer iterations
maxIterIn	maximal number of inner iterations
breakOuter	change in fit required to break the outer iteration. Note: The value will be multiplied internally with sample size N as the $-2\log$ -Likelihood depends directly on the sample size
convCritInner	this is related to the inner breaking condition. 0 = ista, as presented by Beck & Teboulle (2009); see Remark 3.1 on p. 191 (ISTA with backtracking) 1 = gist, as presented by Gong et al. (2013) (Equation 3)
sigma	sigma in (0,1) is used by the gist convergence criterion. larger sigma enforce larger improvement in fit
stepSizeInheritance	how should step sizes be carried forward from iteration to iteration? 0 = resets the step size to L0 in each iteration 1 = takes the previous step size as initial

	value for the next iteration 3 = Barzilai-Borwein procedure 4 = Barzilai-Borwein procedure, but sometimes resets the step size; this can help when the optimizer is caught in a bad spot.
verbose	if set to a value > 0, the fit every "verbose" iterations is printed.
nCores	number of core to use. Multi-core support is provided by RcppParallel and only supported for SEM, not for general purpose optimization.

Value

object of class controlIsta

Examples

```
control <- controlIsta()
```

covariances	<i>covariances</i>
-------------	--------------------

Description

Extract the labels of all covariances found in a lavaan model.

Usage

```
covariances(lavaanModel)
```

Arguments

lavaanModel fitted lavaan model

Value

vector with parameter labels

Examples

```
# The following is adapted from ?lavaan::sem
library(lessSEM)
model <- '
  # latent variable definitions
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + a*y2 + b*y3 + c*y4
  dem65 =~ y5 + a*y6 + b*y7 + c*y8

  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60

  # residual correlations
```

```

    y1 ~~ y5
    y2 ~~ y4 + y6
    y3 ~~ y7
    y4 ~~ y8
    y6 ~~ y8
  ,

  fit <- sem(model, data = PoliticalDemocracy)

  covariances(fit)

```

createSubsets

*createSubsets***Description**

create subsets for cross-validation

Usage

```
createSubsets(N, k)
```

Arguments

N	number of samples in the data set
k	number of subsets to create

Value

matrix with subsets

Examples

```
createSubsets(N=100, k = 5)
```

curveLambda

*curveLambda***Description**

generates lambda values between 0 and lambdaMax using the function described here: <https://math.stackexchange.com/questions/111111/function-with-values-between-0-and-1-for-x-values-between-0-and-1>. The function is identical to the one implemented in the regCtsem package.

Usage

```
curveLambda(maxLambda, lambdasAutoCurve, lambdasAutoLength)
```

Arguments

maxLambda maximal lambda value
 lambdasAutoCurve controls the curve. A value close to 1 will result in a linear increase, larger values in lambdas more concentrated around 0
 lambdasAutoLength number of lambda values to generate

Value

numeric vector

Examples

```
library(lessSEM)
plot(curveLambda(maxLambda = 10, lambdasAutoCurve = 1, lambdasAutoLength = 100))
plot(curveLambda(maxLambda = 10, lambdasAutoCurve = 5, lambdasAutoLength = 100))
plot(curveLambda(maxLambda = 10, lambdasAutoCurve = 100, lambdasAutoLength = 100))
```

cvAdaptiveLasso

cvAdaptiveLasso

Description

Implements cross-validated adaptive lasso regularization for structural equation models. The penalty function is given by:

$$p(x_j) = p(x_j) = \frac{1}{w_j} \lambda |x_j|$$

Adaptive lasso regularization will set parameters to zero if λ is large enough.

Usage

```
cvAdaptiveLasso(
  lavaanModel,
  regularized,
  weights = NULL,
  lambdas,
  k = 5,
  standardize = FALSE,
  returnSubsetParameters = FALSE,
  method = "glmnet",
  modifyModel = lessSEM::modifyModel(),
  control = lessSEM::controlGlmnet()
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
weights	labeled vector with weights for each of the parameters in the model. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object. If set to <code>NULL</code> , the default weights will be used: the inverse of the absolute values of the unregularized parameter estimates
lambdas	numeric vector: values for the tuning parameter lambda
k	the number of cross-validation folds. Alternatively, you can pass a matrix with booleans (<code>TRUE</code> , <code>FALSE</code>) which indicates for each person which subset it belongs to. See <code>?lessSEM::createSubsets</code> for an example of how this matrix should look like.
standardize	Standardizing your data prior to the analysis can undermine the cross-validation. Set <code>standardize=TRUE</code> to automatically standardize the data.
returnSubsetParameters	set to <code>TRUE</code> to return the parameters for each training set
method	which optimizer should be used? Currently implemented are <code>ista</code> and <code>glmnet</code> . With <code>ista</code> , the <code>control</code> argument can be used to switch to related procedures (currently <code>gist</code>).
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
control	used to control the optimizer. This element is generated with the <code>controlIsta</code> and <code>controlGlmnet</code> functions. See <code>?controlIsta</code> and <code>?controlGlmnet</code> for more details.

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

Adaptive lasso regularization:

- Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476), 1418–1429. <https://doi.org/10.1198/016214506000000735>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

model of class `cvRegularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
    16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
    111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- cvAdaptiveLasso(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("1", 6:15),
```

```

    lambdas = seq(0,1,.1))

# use the plot-function to plot the cross-validation fit
plot(lsem)

# the coefficients can be accessed with:
coef(lsem)
# if you are only interested in the estimates and not the tuning parameters, use
coef(lsem)$estimates
# or
estimates(lsem)

# elements of lsem can be accessed with the @ operator:
lsem$parameters

# The best parameters can also be extracted with:
estimates(lsem)

```

cvCappedL1

cvCappedL1

Description

Implements cappedL1 regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \lambda \min(|x_j|, \theta)$$

where $\theta > 0$. The cappedL1 penalty is identical to the lasso for parameters which are below θ and identical to a constant for parameters above θ . As adding a constant to the fitting function will not change its minimum, larger parameters can stay unregularized while smaller ones are set to zero.

Usage

```

cvCappedL1(
  lavaanModel,
  regularized,
  lambdas,
  thetas,
  k = 5,
  standardize = FALSE,
  returnSubsetParameters = FALSE,
  modifyModel = lessSEM::modifyModel(),
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)

```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
thetas	parameters whose absolute value is above this threshold will be penalized with a constant (theta)
k	the number of cross-validation folds. Alternatively, you can pass a matrix with booleans (TRUE, FALSE) which indicates for each person which subset it belongs to. See <code>?lessSEM::createSubsets</code> for an example of how this matrix should look like.
standardize	Standardizing your data prior to the analysis can undermine the cross-validation. Set <code>standardize=TRUE</code> to automatically standardize the data.
returnSubsetParameters	set to TRUE to return the parameters for each training set
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
method	which optimizer should be used? Currently implemented are <code>ista</code> and <code>glmnet</code> . With <code>ista</code> , the <code>control</code> argument can be used to switch to related procedures.
control	used to control the optimizer. This element is generated with the <code>controlIsta</code> function. See <code>?controlIsta</code> for more details.

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

CappedL1 regularization:

- Zhang, T. (2010). Analysis of Multi-stage Convex Relaxation for Sparse Regularization. *Journal of Machine Learning Research*, 11, 1081–1107.

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>

- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

model of class `cvRegularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f ~~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- cvCappedL1(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("1", 6:15),
  lambdas = seq(0,1,length.out = 5),
  thetas = seq(0.01,2,length.out = 3))
```

```
# the coefficients can be accessed with:
coef(lsem)
# if you are only interested in the estimates and not the tuning parameters, use
coef(lsem)$estimates
# or
estimates(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters

# optional: plotting the cross-validation fit requires installation of plotly
# plot(lsem)
```

cvElasticNet

cvElasticNet

Description

Implements elastic net regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \alpha \lambda |x_j| + (1 - \alpha) \lambda x_j^2$$

Note that the elastic net combines ridge and lasso regularization. If $\alpha = 0$, the elastic net reduces to ridge regularization. If $\alpha = 1$ it reduces to lasso regularization. In between, elastic net is a compromise between the shrinkage of the lasso and the ridge penalty.

Usage

```
cvElasticNet(
  lavaanModel,
  regularized,
  lambdas,
  alphas,
  k = 5,
  standardize = FALSE,
  returnSubsetParameters = FALSE,
  method = "glmnet",
  modifyModel = lessSEM::modifyModel(),
  control = lessSEM::controlGlmnet()
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object

lambdas	numeric vector: values for the tuning parameter lambda
alphas	numeric vector with values of the tuning parameter alpha. Must be between 0 and 1. 0 = ridge, 1 = lasso.
k	the number of cross-validation folds. Alternatively, you can pass a matrix with booleans (TRUE, FALSE) which indicates for each person which subset it belongs to. See ?lessSEM::createSubsets for an example of how this matrix should look like.
standardize	Standardizing your data prior to the analysis can undermine the cross-validation. Set standardize=TRUE to automatically standardize the data.
returnSubsetParameters	set to TRUE to return the parameters for each training set
method	which optimizer should be used? Currently implemented are ista and glmnet. With ista, the control argument can be used to switch to related procedures.
modifyModel	used to modify the lavaanModel. See ?modifyModel.
control	used to control the optimizer. This element is generated with the controlIsta and controlGlmnet functions. See ?controlIsta and ?controlGlmnet for more details.

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

Elastic net regularization:

- Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2), 301–320. <https://doi.org/10.1111/j.1467-9868.2005.00503.x>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

model of class `cvRegularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- cvElasticNet(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("1", 6:15),
  lambdas = seq(0,1,length.out = 5),
  alphas = seq(0,1,length.out = 3))

# the coefficients can be accessed with:
coef(lsem)
# if you are only interested in the estimates and not the tuning parameters, use
coef(lsem)$estimates
# or
estimates(lsem)

# elements of lsem can be accessed with the @ operator:
```

```
lsem@parameters

# optional: plotting the cross-validation fit requires installation of plotly
# plot(lsem)
```

cvLasso

cvLasso

Description

Implements cross-validated lasso regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \lambda |x_j|$$

Lasso regularization will set parameters to zero if λ is large enough

Usage

```
cvLasso(
  lavaanModel,
  regularized,
  lambdas,
  k = 5,
  standardize = FALSE,
  returnSubsetParameters = FALSE,
  method = "glmnet",
  modifyModel = lessSEM::modifyModel(),
  control = lessSEM::controlGlmnet()
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
k	the number of cross-validation folds. Alternatively, you can pass a matrix with booleans (TRUE, FALSE) which indicates for each person which subset it belongs to. See <code>?lessSEM::createSubsets</code> for an example of how this matrix should look like.
standardize	Standardizing your data prior to the analysis can undermine the cross-validation. Set <code>standardize=TRUE</code> to automatically standardize the data.
returnSubsetParameters	set to TRUE to return the parameters for each training set
method	which optimizer should be used? Currently implemented are <code>ista</code> and <code>glmnet</code> . With <code>ista</code> , the <code>control</code> argument can be used to switch to related procedures.

<code>modifyModel</code>	used to modify the <code>lavaanModel</code> . See <code>?modifyModel</code> .
<code>control</code>	used to control the optimizer. This element is generated with the <code>controlIsta</code> and <code>controlGlmnet</code> functions. See <code>?controlIsta</code> and <code>?controlGlmnet</code> for more details.

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

Lasso regularization:

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

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

model of class `cvRegularizedSEM`

Examples

```

library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- cvLasso(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("l", 6:15),
  lambdas = seq(0,1,.1),
  k = 5, # number of cross-validation folds
  standardize = TRUE) # automatic standardization

# use the plot-function to plot the cross-validation fit:
plot(lsem)

# the coefficients can be accessed with:
coef(lsem)
# if you are only interested in the estimates and not the tuning parameters, use
coef(lsem)$estimates
# or
estimates(lsem)

# elements of lsem can be accessed with the @ operator:
lsem$parameters

# The best parameters can also be extracted with:
estimates(lsem)

```

Description

Implements lsp regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \lambda \log(1 + |x_j|/\theta)$$

where $\theta > 0$.

Usage

```
cvLsp(
  lavaanModel,
  regularized,
  lambdas,
  thetas,
  k = 5,
  standardize = FALSE,
  returnSubsetParameters = FALSE,
  modifyModel = lessSEM::modifyModel(),
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
thetas	parameters whose absolute value is above this threshold will be penalized with a constant (theta)
k	the number of cross-validation folds. Alternatively, you can pass a matrix with booleans (TRUE, FALSE) which indicates for each person which subset it belongs to. See <code>?lessSEM::createSubsets</code> for an example of how this matrix should look like.
standardize	Standardizing your data prior to the analysis can undermine the cross-validation. Set <code>standardize=TRUE</code> to automatically standardize the data.
returnSubsetParameters	set to TRUE to return the parameters for each training set
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
method	which optimizer should be used? Currently implemented are ista and glmnet. With ista, the control argument can be used to switch to related procedures.
control	used to control the optimizer. This element is generated with the <code>controlIsta</code> function. See <code>?controlIsta</code>

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

lsp regularization:

- Candès, E. J., Wakin, M. B., & Boyd, S. P. (2008). Enhancing Sparsity by Reweighted l1 Minimization. *Journal of Fourier Analysis and Applications*, 14(5–6), 877–905. <https://doi.org/10.1007/s00041-008-9045-x>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

model of class `cvRegularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
```

```

# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f ~~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- cvLsp(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("1", 6:15),
  lambdas = seq(0,1,length.out = 5),
  thetas = seq(0.01,2,length.out = 3))

# the coefficients can be accessed with:
coef(lsem)
# if you are only interested in the estimates and not the tuning parameters, use
coef(lsem)$estimates
# or
estimates(lsem)

# elements of lsem can be accessed with the @ operator:
lsem$parameters

# optional: plotting the cross-validation fit requires installation of plotly
# plot(lsem)

```

cvMcp

cvMcp

Description

Implements mcp regularization for structural equation models. The penalty function is given by:
Equation Omitted in Pdf Documentation.

Usage

```
cvMcp(
```

```

lavaanModel,
regularized,
lambdas,
thetas,
k = 5,
standardize = FALSE,
returnSubsetParameters = FALSE,
modifyModel = lessSEM::modifyModel(),
method = "ista",
control = lessSEM::controlIsta()
)

```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
thetas	parameters whose absolute value is above this threshold will be penalized with a constant (theta)
k	the number of cross-validation folds. Alternatively, you can pass a matrix with booleans (TRUE, FALSE) which indicates for each person which subset it belongs to. See <code>?lessSEM::createSubsets</code> for an example of how this matrix should look like.
standardize	Standardizing your data prior to the analysis can undermine the cross-validation. Set <code>standardize=TRUE</code> to automatically standardize the data.
returnSubsetParameters	set to TRUE to return the parameters for each training set
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
method	which optimizer should be used? Currently implemented are ista and glmnet. With ista, the control argument can be used to switch to related procedures.
control	used to control the optimizer. This element is generated with the <code>controlIsta</code> function. See <code>?controlIsta</code>

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

mcp regularization:

- Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. The Annals of Statistics, 38(2), 894–942. <https://doi.org/10.1214/09-AOS729>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

model of class `cvRegularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
```

```

                                std.lv = TRUE)

# Regularization:

lsem <- cvMcp(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("l", 6:15),
  lambdas = seq(0,1,length.out = 5),
  thetas = seq(0.01,2,length.out = 3))

# the coefficients can be accessed with:
coef(lsem)
# if you are only interested in the estimates and not the tuning parameters, use
coef(lsem)$estimates
# or
estimates(lsem)

# elements of lsem can be accessed with the @ operator:
lsem$parameters

# optional: plotting the cross-validation fit requires installation of plotly
# plot(lsem)

```

cvRegularizedSEM-class

Class for cross-validated regularized SEM

Description

Class for cross-validated regularized SEM

Slots

parameters data.frame with parameter estimates for the best combination of the tuning parameters

transformations transformed parameters

cvfits data.frame with all combinations of the tuning parameters and the sum of the cross-validation fits

parameterLabels character vector with names of all parameters

regularized character vector with names of regularized parameters

cvfitsDetails data.frame with cross-validation fits for each subset

subsets matrix indicating which person is in which subset

subsetParameters optional: data.frame with parameter estimates for all combinations of the tuning parameters in all subsets

misc list with additional return elements

notes internal notes that have come up when fitting the model

cvRidge

cvRidge

Description

Implements ridge regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \lambda x_j^2$$

Note that ridge regularization will not set any of the parameters to zero but result in a shrinkage towards zero.

Usage

```
cvRidge(
  lavaanModel,
  regularized,
  lambdas,
  k = 5,
  standardize = FALSE,
  returnSubsetParameters = FALSE,
  method = "glmnet",
  modifyModel = lessSEM::modifyModel(),
  control = lessSEM::controlGlmnet()
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
k	the number of cross-validation folds. Alternatively, you can pass a matrix with booleans (TRUE, FALSE) which indicates for each person which subset it belongs to. See <code>?lessSEM::createSubsets</code> for an example of how this matrix should look like.
standardize	Standardizing your data prior to the analysis can undermine the cross-validation. Set <code>standardize=TRUE</code> to automatically standardize the data.
returnSubsetParameters	set to TRUE to return the parameters for each training set
method	which optimizer should be used? Currently implemented are ista and glmnet. With ista, the control argument can be used to switch to related procedures (currently gist).
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
control	used to control the optimizer. This element is generated with the <code>controlIsta</code> and <code>controlGlmnet</code> functions. See <code>?controlIsta</code> and <code>?controlGlmnet</code> for more details.

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

Ridge regularization:

- Hoerl, A. E., & Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1), 55–67. <https://doi.org/10.1080/00401706.1970.10488634>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

model of class `cvRegularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.
```

```

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- cvRidge(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("1", 6:15),
  lambdas = seq(0,1,length.out = 20))

# use the plot-function to plot the cross-validation fit:
plot(lsem)

# the coefficients can be accessed with:
coef(lsem)
# if you are only interested in the estimates and not the tuning parameters, use
coef(lsem)$estimates
# or
estimates(lsem)

# elements of lsem can be accessed with the @ operator:
lsem$parameters

```

cvRidgeBfgs

cvRidgeBfgs

Description

Implements cross-validated ridge regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \lambda x_j^2$$

Note that ridge regularization will not set any of the parameters to zero but result in a shrinkage towards zero.

Usage

```
cvRidgeBfgs(
  lavaanModel,
  regularized,
  lambdas,
  k = 5,
  standardize = FALSE,
  returnSubsetParameters = FALSE,
  modifyModel = lessSEM::modifyModel(),
  control = lessSEM::controlBFGS()
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
k	the number of cross-validation folds. Alternatively, you can pass a matrix with booleans (TRUE, FALSE) which indicates for each person which subset it belongs to. See <code>?lessSEM::createSubsets</code> for an example of how this matrix should look like.
standardize	Standardizing your data prior to the analysis can undermine the cross-validation. Set <code>standardize=TRUE</code> to automatically standardize the data.
returnSubsetParameters	set to TRUE to return the parameters for each training set
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
control	used to control the optimizer. This element is generated with the <code>controlBFGS</code> function. See <code>?controlBFGS</code> for more details.

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

Ridge regularization:

- Hoerl, A. E., & Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1), 55–67. <https://doi.org/10.1080/00401706.1970.10488634>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>

- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

Value

model of class `cvRegularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ l1*y1 + l2*y2 + l3*y3 + l4*y4 + l5*y5 +
      l6*y6 + l7*y7 + l8*y8 + l9*y9 + l10*y10 +
      l11*y11 + l12*y12 + l13*y13 + l14*y14 + l15*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- cvRidgeBfgs(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("l", 6:15),
  lambdas = seq(0,1,length.out = 20))

# use the plot-function to plot the cross-validation fit:
plot(lsem)

# the coefficients can be accessed with:
coef(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters
```

cvScad

cvScad

Description

Implements scad regularization for structural equation models. The penalty function is given by:
Equation Omitted in Pdf Documentation.

Usage

```
cvScad(
  lavaanModel,
  regularized,
  lambdas,
  thetas,
  k = 5,
  standardize = FALSE,
  returnSubsetParameters = FALSE,
  modifyModel = lessSEM::modifyModel(),
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
thetas	parameters whose absolute value is above this threshold will be penalized with a constant (theta)
k	the number of cross-validation folds. Alternatively, you can pass a matrix with booleans (TRUE, FALSE) which indicates for each person which subset it belongs to. See <code>?lessSEM::createSubsets</code> for an example of how this matrix should look like.
standardize	Standardizing your data prior to the analysis can undermine the cross-validation. Set <code>standardize=TRUE</code> to automatically standardize the data.
returnSubsetParameters	set to TRUE to return the parameters for each training set
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
method	which optimizer should be used? Currently implemented are ista and glmnet. With ista, the control argument can be used to switch to related procedures.
control	used to control the optimizer. This element is generated with the <code>controlIsta</code> function. See <code>?controlIsta</code>

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

scad regularization:

- 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. <https://doi.org/10.1198/01621450175>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

model of class `cvRegularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.
```

```

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f ~~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- cvScad(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("1", 6:15),
  lambdas = seq(0,1,length.out = 3),
  thetas = seq(2.01,5,length.out = 3))

# the coefficients can be accessed with:
coef(lsem)
# if you are only interested in the estimates and not the tuning parameters, use
coef(lsem)$estimates
# or
estimates(lsem)

# elements of lsem can be accessed with the @ operator:
lsem$parameters

# optional: plotting the cross-validation fit requires installation of plotly
# plot(lsem)

```

cvScaler

cvScaler

Description

uses the means and standard deviations of the training set to standardize the test set. See, e.g., https://scikit-learn.org/stable/modules/cross_validation.html.

Usage

```
cvScaler(testSet, means, standardDeviations)
```

Arguments

testSet	test data set
means	means of the training set
standardDeviations	standard deviations of the training set

Value

scaled test set

Examples

```
library(lessSEM)
data <- matrix(rnorm(50),10,5)

cvScaler(testSet = data,
          means = 1:5,
          standardDeviations = 1:5)
```

cvSmoothAdaptiveLasso *cvSmoothAdaptiveLasso*

Description

Implements cross-validated smooth adaptive lasso regularization for structural equation models. The penalty function is given by:

$$p(x_j) = p(x_j) = \frac{1}{w_j} \lambda \sqrt{(x_j + \epsilon)^2}$$

Usage

```
cvSmoothAdaptiveLasso(
  lavaanModel,
  regularized,
  weights = NULL,
  lambdas,
  epsilon,
  k = 5,
  standardize = FALSE,
  returnSubsetParameters = FALSE,
  modifyModel = lessSEM::modifyModel(),
  control = lessSEM::controlBFGS()
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
weights	labeled vector with weights for each of the parameters in the model. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object. If set to NULL, the default weights will be used: the inverse of the absolute values of the unregularized parameter estimates
lambdas	numeric vector: values for the tuning parameter lambda
epsilon	$\epsilon > 0$; controls the smoothness of the approximation. Larger values = smoother
k	the number of cross-validation folds. Alternatively, you can pass a matrix with booleans (TRUE, FALSE) which indicates for each person which subset it belongs to. See <code>?lessSEM::createSubsets</code> for an example of how this matrix should look like.
standardize	Standardizing your data prior to the analysis can undermine the cross-validation. Set <code>standardize=TRUE</code> to automatically standardize the data.
returnSubsetParameters	set to TRUE to return the parameters for each training set
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
control	used to control the optimizer. This element is generated with the <code>controlBFGS</code> function. See <code>?controlBFGS</code> for more details.

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

Adaptive lasso regularization:

- Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476), 1418–1429. <https://doi.org/10.1198/016214506000000735>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

Value

model of class `cvRegularizedSEM`

Examples

```

library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- cvSmoothAdaptiveLasso(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("1", 6:15),
  lambdas = seq(0,1,.1),
  epsilon = 1e-8)

# use the plot-function to plot the cross-validation fit
plot(lsem)

# the coefficients can be accessed with:
coef(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters

# The best parameters can also be extracted with:
coef(lsem)

```

Description

Implements cross-validated smooth elastic net regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \alpha \lambda \sqrt{(x_j + \epsilon)^2} + (1 - \alpha) \lambda x_j^2$$

Note that the smooth elastic net combines ridge and smooth lasso regularization. If $\alpha = 0$, the elastic net reduces to ridge regularization. If $\alpha = 1$ it reduces to smooth lasso regularization. In between, elastic net is a compromise between the shrinkage of the lasso and the ridge penalty.

Usage

```
cvSmoothElasticNet(
  lavaanModel,
  regularized,
  lambdas,
  alphas,
  epsilon,
  k = 5,
  standardize = FALSE,
  returnSubsetParameters = FALSE,
  modifyModel = lessSEM::modifyModel(),
  control = lessSEM::controlBFGS()
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
alphas	numeric vector with values of the tuning parameter alpha. Must be between 0 and 1. 0 = ridge, 1 = lasso.
epsilon	epsilon > 0; controls the smoothness of the approximation. Larger values = smoother
k	the number of cross-validation folds. Alternatively, you can pass a matrix with booleans (TRUE, FALSE) which indicates for each person which subset it belongs to. See <code>?lessSEM::createSubsets</code> for an example of how this matrix should look like.
standardize	Standardizing your data prior to the analysis can undermine the cross-validation. Set <code>standardize=TRUE</code> to automatically standardize the data.
returnSubsetParameters	set to TRUE to return the parameters for each training set
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
control	used to control the optimizer. This element is generated with the <code>controlBFGS</code> function. See <code>?controlBFGS</code> for more details.

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

Elastic net regularization:

- Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2), 301–320. <https://doi.org/10.1111/j.1467-9868.2005.00503.x>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

Value

model of class `cvRegularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f ~~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- cvSmoothElasticNet(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
```

```

regularized = paste0("1", 6:15),
epsilon = 1e-8,
lambdas = seq(0,1,length.out = 5),
alphas = .3)

# the coefficients can be accessed with:
coef(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters

# optional: plotting the cross-validation fit requires installation of plotly
# plot(lsem)

```

cvSmoothLasso

cvSmoothLasso

Description

Implements cross-validated smooth lasso regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \lambda \sqrt{(x_j + \epsilon)^2}$$

Usage

```

cvSmoothLasso(
  lavaanModel,
  regularized,
  lambdas,
  epsilon,
  k = 5,
  standardize = FALSE,
  returnSubsetParameters = FALSE,
  modifyModel = lessSEM::modifyModel(),
  control = lessSEM::controlBFGS()
)

```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
epsilon	epsilon > 0; controls the smoothness of the approximation. Larger values = smoother

k	the number of cross-validation folds. Alternatively, you can pass a matrix with booleans (TRUE, FALSE) which indicates for each person which subset it belongs to. See ?lessSEM::createSubsets for an example of how this matrix should look like.
standardize	Standardizing your data prior to the analysis can undermine the cross-validation. Set standardize=TRUE to automatically standardize the data.
returnSubsetParameters	set to TRUE to return the parameters for each training set
modifyModel	used to modify the lavaanModel. See ?modifyModel.
control	used to control the optimizer. This element is generated with the controlBFGS function. See ?controlBFGS for more details.

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

Lasso regularization:

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

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

Value

model of class `cvRegularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
```

```

f ~~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- cvSmoothLasso(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("1", 6:15),
  lambdas = seq(0,1,.1),
  k = 5, # number of cross-validation folds
  epsilon = 1e-8,
  standardize = TRUE) # automatic standardization

# use the plot-function to plot the cross-validation fit:
plot(lsem)

# the coefficients can be accessed with:
coef(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters

# The best parameters can also be extracted with:
coef(lsem)

```

elasticNet

elasticNet

Description

Implements elastic net regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \alpha \lambda |x_j| + (1 - \alpha) \lambda x_j^2$$

Note that the elastic net combines ridge and lasso regularization. If $\alpha = 0$, the elastic net reduces to ridge regularization. If $\alpha = 1$ it reduces to lasso regularization. In between, elastic net is a compromise between the shrinkage of the lasso and the ridge penalty.

Usage

```

elasticNet(
  lavaanModel,
  regularized,

```

```

    lambdas,
    alphas,
    method = "glmnet",
    modifyModel = lessSEM::modifyModel(),
    control = lessSEM::controlGlmnet()
  )

```

Arguments

<code>lavaanModel</code>	model of class <code>lavaan</code>
<code>regularized</code>	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your <code>lavaan</code> model object
<code>lambdas</code>	numeric vector: values for the tuning parameter <code>lambda</code>
<code>alphas</code>	numeric vector with values of the tuning parameter <code>alpha</code> . Must be between 0 and 1. 0 = ridge, 1 = lasso.
<code>method</code>	which optimizer should be used? Currently implemented are <code>ista</code> and <code>glmnet</code> . With <code>ista</code> , the <code>control</code> argument can be used to switch to related procedures (currently <code>gist</code>).
<code>modifyModel</code>	used to modify the <code>lavaanModel</code> . See <code>?modifyModel</code> .
<code>control</code>	used to control the optimizer. This element is generated with the <code>lessSEM::controlIsta()</code> and <code>controlGlmnet()</code> functions.

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

Elastic net regularization:

- Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2), 301–320. <https://doi.org/10.1111/j.1467-9868.2005.00503.x>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>

- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Model of class regularizedSEM

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f ~~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- elasticNet(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("1", 6:15),
  lambdas = seq(0,1,length.out = 5),
  alphas = seq(0,1,length.out = 3))
```

```

# the coefficients can be accessed with:
coef(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters[1,]

# optional: plotting the paths requires installation of plotly
# plot(lsem)

#### Advanced ###
# Switching the optimizer #
# Use the "method" argument to switch the optimizer. The control argument
# must also be changed to the corresponding function:
lsemIsta <- elasticNet(
  lavaanModel = lavaanModel,
  regularized = paste0("l", 6:15),
  lambdas = seq(0,1,length.out = 5),
  alphas = seq(0,1,length.out = 3),
  method = "ista",
  control = controlIsta())

# Note: The results are basically identical:
lsemIsta@parameters - lsem@parameters

```

estimates

S4 method to extract the estimates of an object

Description

S4 method to extract the estimates of an object

Usage

```
estimates(object, criterion = NULL, transformations = FALSE)
```

Arguments

object	a model fitted with lessSEM
criterion	fitIndice used to select the parameters
transformations	boolean: Should transformations be returned?

Value

returns a matrix with estimates

estimates,cvRegularizedSEM-method
estimates

Description

estimates

Usage

```
## S4 method for signature 'cvRegularizedSEM'  
estimates(object, criterion = NULL, transformations = FALSE)
```

Arguments

object	object of class cvRegularizedSEM
criterion	not used
transformations	boolean: Should transformations be returned?

Value

returns a matrix with estimates

estimates,regularizedSEM-method
estimates

Description

estimates

Usage

```
## S4 method for signature 'regularizedSEM'  
estimates(object, criterion = NULL, transformations = FALSE)
```

Arguments

object	object of class regularizedSEM
criterion	fit index (e.g., AIC) used to select the parameters
transformations	boolean: Should transformations be returned?

Value

returns a matrix with estimates

estimates,regularizedSEMMixedPenalty-method
<i>estimates</i>

Description

estimates

Usage

```
## S4 method for signature 'regularizedSEMMixedPenalty'
estimates(object, criterion = NULL, transformations = FALSE)
```

Arguments

- object object of class regularizedSEMMixedPenalty
- criterion fit index (e.g., AIC) used to select the parameters
- transformations boolean: Should transformations be returned?

Value

returns a matrix with estimates

fit	<i>fit</i>
-----	------------

Description

Optimizes an object with mixed penalty. See ?mixedPenalty for more details.

Usage

```
fit(mixedPenalty)
```

Arguments

- mixedPenalty object of class mixedPenalty. This object can be created with the mixedPenalty function. Penalties can be added with the addCappedL1, addElastiNet, addLasso, addLsp, addMcp, and addScad functions.

Value

throws error in case of undefined penalty combinations.

Examples

```

library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f ~~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# We can add mixed penalties as follows:

regularized <- lavaanModel |>
  # create template for regularized model with mixed penalty:
  mixedPenalty() |>
  # add penalty on loadings l6 - l10:
  addElasticNet(regularized = paste0("l", 11:15),
                lambdas = seq(0,1,.1),
                alphas = .4) |>
  # fit the model:
  fit()

```

fitIndices

*S4 method to compute fit indices (e.g., AIC, BIC, ...)***Description**

S4 method to compute fit indices (e.g., AIC, BIC, ...)

Usage

```
fitIndices(object)
```

Arguments

object a model fitted with lessSEM

Value

returns a data.frame with fit indices

<code>fitIndices,cvRegularizedSEM-method</code> <i>fitIndices</i>
--

Description

fitIndices

Usage

```
## S4 method for signature 'cvRegularizedSEM'  
fitIndices(object)
```

Arguments

object object of class cvRegularizedSEM

Value

returns a data.frame with fit indices

<code>fitIndices,regularizedSEM-method</code> <i>fitIndices</i>
--

Description

fitIndices

Usage

```
## S4 method for signature 'regularizedSEM'  
fitIndices(object)
```

Arguments

object object of class regularizedSEM

Value

returns a data.frame with fit indices

fitIndices,regularizedSEMMixedPenalty-method
fitIndices

Description

fitIndices

Usage

```
## S4 method for signature 'regularizedSEMMixedPenalty'  
fitIndices(object)
```

Arguments

object object of class regularizedSEMMixedPenalty

Value

returns a data.frame with fit indices

getLavaanParameters *getLavaanParameters*

Description

helper function: returns a labeled vector with parameters from lavaan

Usage

```
getLavaanParameters(lavaanModel, removeDuplicates = TRUE)
```

Arguments

lavaanModel model of class lavaan
removeDuplicates should duplicated parameters be removed?

Value

returns a labeled vector with parameters from lavaan

Examples

```
library(lessSEM)

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)
getLavaanParameters(lavaanModel)
```

```
getTuningParameterConfiguration
```

```
getTuningParameterConfiguration
```

Description

Returns the lambda, theta, and alpha values for the tuning parameters of a regularized SEM with mixed penalty.

Usage

```
getTuningParameterConfiguration(
  regularizedSEMMixedPenalty,
  tuningParameterConfiguration
)
```

Arguments

```
regularizedSEMMixedPenalty
  object of type regularizedSEMMixedPenalty (see ?mixedPenalty)
tuningParameterConfiguration
  integer indicating which tuningParameterConfiguration should be extracted (e.g.,
  1). See the entry in the row tuningParameterConfiguration of regularizedSEM-
  MixedPenalty@fits and regularizedSEMMixedPenalty@parameters.
```

Value

data frame with penalty and tuning parameter settings

Examples

```

library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# We can add mixed penalties as follows:

regularized <- lavaanModel |>
  # create template for regularized model with mixed penalty:
  mixedPenalty() |>
  # add penalty on loadings 16 - 110:
  addLsp(regularized = paste0("1", 11:15),
         lambdas = seq(0,1,.1),
         thetas = 2.3) |>
  # fit the model:
  fit()

getTuningParameterConfiguration(regularizedSEMMixedPenalty = regularized,
                                tuningParameterConfiguration = 2)

```

glmnetCappedL1MgSEM *CappedL1 optimization with glmnet optimizer*

Description

Object for cappedL1 optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (2) a list with control elements
`setHessian` changes the Hessian of the model. Expects a matrix
`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

glmnetCappedL1SEM	<i>CappedL1 optimization with glmnet optimizer</i>
-------------------	--

Description

Object for cappedL1 optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires a list with control elements
`setHessian` changes the Hessian of the model. Expects a matrix
`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

glmnetEnetGeneralPurpose	<i>elastic net optimization with glmnet optimizer</i>
--------------------------	---

Description

Object for elastic net optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements
`setHessian` changes the Hessian of the model. Expects a matrix
`optimize` optimize the model. Expects a vector with starting values, an R function to compute the fit, an R function to compute the gradients, a list with elements the fit and gradient function require, a lambda and an alpha value.

glmnetEnetGeneralPurposeCpp	<i>elastic net optimization with glmnet optimizer</i>
-----------------------------	---

Description

Object for elastic net optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEXP function pointer to compute the fit, a SEXP function pointer to compute the gradients, a list with elements the fit and gradient function require, a lambda and an alpha value.

glmnetEnetMgSEM	<i>elastic net optimization with glmnet optimizer</i>
-----------------	---

Description

Object for elastic net optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a lambda and an alpha value.

glmnetEnetSEM	<i>elastic net optimization with glmnet optimizer</i>
---------------	---

Description

Object for elastic net optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a lambda and an alpha value.

glmnetLspMgSEM	<i>lsp optimization with glmnet optimizer</i>
----------------	---

Description

Object for lsp optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (2) a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

`glmnetLspSEM`*lsp optimization with glmnet optimizer*

Description

Object for lsp optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type `SEM_Cpp`, a `theta` and a `lambda` value.

`glmnetMcpMgSEM`*mcp optimization with glmnet optimizer*

Description

Object for mcp optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (2) a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type `SEM_Cpp`, a `theta` and a `lambda` value.

glmnetMcpSEM	<i>mcp optimization with glmnet optimizer</i>
--------------	---

Description

Object for mcp optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

glmnetMixedMgSEM	<i>mixed optimization with glmnet optimizer</i>
------------------	---

Description

Object for mixed optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (2) a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

`glmnetMixedPenaltyGeneralPurpose`*mixed optimization with glmnet optimizer*

Description

Object for mixed optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

`glmnetMixedPenaltyGeneralPurposeCpp`*mixed optimization with glmnet optimizer*

Description

Object for mixed optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

glmnetMixedSEM	<i>mixed optimization with glmnet optimizer</i>
----------------	---

Description

Object for mixed optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

glmnetScadMgSEM	<i>scad optimization with glmnet optimizer</i>
-----------------	--

Description

Object for scad optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (2) a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

glmnetScadSEM	<i>scad optimization with glmnet optimizer</i>
---------------	--

Description

Object for scad optimization with glmnet optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires a list with control elements

`setHessian` changes the Hessian of the model. Expects a matrix

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

gpAdaptiveLasso	<i>gpAdaptiveLasso</i>
-----------------	------------------------

Description

Implements adaptive lasso regularization for general purpose optimization problems. The penalty function is given by:

$$p(x_j) = p(x_j) = \frac{1}{w_j} \lambda |x_j|$$

Adaptive lasso regularization will set parameters to zero if λ is large enough.

Usage

```
gpAdaptiveLasso(
  par,
  regularized,
  weights = NULL,
  fn,
  gr = NULL,
  lambdas = NULL,
  nLambdas = NULL,
  reverse = TRUE,
  curve = 1,
  ...,
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)
```

Arguments

<code>par</code>	labeled vector with starting values
<code>regularized</code>	vector with names of parameters which are to be regularized.
<code>weights</code>	labeled vector with adaptive lasso weights. NULL will use $1/\text{abs}(\text{par})$
<code>fn</code>	R function which takes the parameters as input and returns the fit value (a single value)
<code>gr</code>	R function which takes the parameters as input and returns the gradients of the objective function. If set to NULL, <code>numDeriv</code> will be used to approximate the gradients
<code>lambdas</code>	numeric vector: values for the tuning parameter lambda
<code>nLambdas</code>	alternative to <code>lambdas</code> : If $\alpha = 1$, <code>lessSEM</code> can automatically compute the first lambda value which sets all regularized parameters to zero. It will then generate <code>nLambda</code> values between 0 and the computed lambda.
<code>reverse</code>	if set to TRUE and <code>nLambdas</code> is used, <code>lessSEM</code> will start with the largest lambda and gradually decrease lambda. Otherwise, <code>lessSEM</code> will start with the smallest lambda and gradually increase it.
<code>curve</code>	Allows for unequally spaced lambda steps (e.g., .01,.02,.05,1,5,20). If curve is close to 1 all lambda values will be equally spaced, if curve is large lambda values will be more concentrated close to 0. See <code>?lessSEM::curveLambda</code> for more information.
<code>...</code>	additional arguments passed to <code>fn</code> and <code>gr</code>
<code>method</code>	which optimizer should be used? Currently implemented are <code>ista</code> and <code>glmnet</code> .
<code>control</code>	used to control the optimizer. This element is generated with the <code>controlIsta</code> and <code>controlGlmnet</code> functions. See <code>?controlIsta</code> and <code>?controlGlmnet</code> for more details.

Details

The interface is similar to that of `optim`. Users have to supply a vector with starting values (important: This vector *must* have labels) and a fitting function. This fitting functions *must* take a labeled vector with parameter values as first argument. The remaining arguments are passed with the `...` argument. This is similar to `optim`.

The gradient function `gr` is optional. If set to NULL, the **numDeriv** package will be used to approximate the gradients. Supplying a gradient function can result in considerable speed improvements.

Adaptive lasso regularization:

- Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476), 1418–1429. <https://doi.org/10.1198/016214506000000735>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.

- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class gpRegularized

Examples

```
# This example shows how to use the optimizers
# for other objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(lessSEM)
set.seed(123)

# first, we simulate data for our
# linear regression.
N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p), nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
       rep(0,6)) # true regression weights
y <- X%%matrix(b, ncol = 1) + rnorm(N, 0, .2)

# First, we must construct a fitting function
# which returns a single value. We will use
# the residual sum squared as fitting function.

# Let's start setting up the fitting function:
fittingFunction <- function(par, y, X, N){
  # par is the parameter vector
  # y is the observed dependent variable
  # X is the design matrix
  # N is the sample size
  pred <- X %% matrix(par, ncol = 1) #be explicit here:
  # we need par to be a column vector
  sse <- sum((y - pred)^2)
  # we scale with .5/N to get the same results as glmnet
```

```

    return((.5/N)*sse)
}

# let's define the starting values:
b <- c(solve(t(X)%*%X)%*%t(X)%*%y) # we will use the lm estimates
names(b) <- paste0("b", 1:length(b))
# names of regularized parameters
regularized <- paste0("b",1:p)

# define the weight for each of the parameters
weights <- 1/abs(b)
# we will re-scale the weights for equivalence to glmnet.
# see ?glmnet for more details
weights <- length(b)*weights/sum(weights)

# optimize
adaptiveLassoPen <- gpAdaptiveLasso(
  par = b,
  regularized = regularized,
  weights = weights,
  fn = fittingFunction,
  lambdas = seq(0,1,.01),
  X = X,
  y = y,
  N = N
)
plot(adaptiveLassoPen)
# You can access the fit results as follows:
adaptiveLassoPen@fits
# Note that we won't compute any fit measures automatically, as
# we cannot be sure how the AIC, BIC, etc are defined for your objective function

# for comparison:
# library(glmnet)
# coef(glmnet(x = X,
#             y = y,
#             penalty.factor = weights,
#             lambda = adaptiveLassoPen@fits$lambda[20],
#             intercept = FALSE,
#             standardize = FALSE))[,1]
# adaptiveLassoPen@parameters[20,]

```

gpAdaptiveLassoCpp

gpAdaptiveLassoCpp

Description

Implements adaptive lasso regularization for general purpose optimization problems with C++ functions. The penalty function is given by:

$$p(x_j) = p(x_j) = \frac{1}{w_j} \lambda |x_j|$$

Adaptive lasso regularization will set parameters to zero if λ is large enough.

Usage

```
gpAdaptiveLassoCpp(
  par,
  regularized,
  weights = NULL,
  fn,
  gr,
  lambdas = NULL,
  nLambdas = NULL,
  curve = 1,
  additionalArguments,
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)
```

Arguments

par	labeled vector with starting values
regularized	vector with names of parameters which are to be regularized.
weights	labeled vector with adaptive lasso weights. NULL will use 1/abs(par)
fn	R function which takes the parameters as input and returns the fit value (a single value)
gr	R function which takes the parameters as input and returns the gradients of the objective function. If set to NULL, numDeriv will be used to approximate the gradients
lambdas	numeric vector: values for the tuning parameter lambda
nLambdas	alternative to lambda: If alpha = 1, lessSEM can automatically compute the first lambda value which sets all regularized parameters to zero. It will then generate nLambda values between 0 and the computed lambda.
curve	Allows for unequally spaced lambda steps (e.g., .01,.02,.05,1,5,20). If curve is close to 1 all lambda values will be equally spaced, if curve is large lambda values will be more concentrated close to 0. See ?lessSEM::curveLambda for more information.
additionalArguments	list with additional arguments passed to fn and gr
method	which optimizer should be used? Currently implemented are ista and glmnet.
control	used to control the optimizer. This element is generated with the controlIsta and controlGlmnet functions. See ?controlIsta and ?controlGlmnet for more details.

Details

The interface is inspired by optim, but a bit more restrictive. Users have to supply a vector with starting values (important: This vector *must* have labels), a fitting function, and a gradient function.

These fitting functions *must* take an `const Rcpp::NumericVector&` with parameter values as first argument and an `Rcpp::List&` as second argument

Adaptive lasso regularization:

- Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476), 1418–1429. <https://doi.org/10.1198/016214506000000735>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class `gpRegularized`

Examples

```
# This example shows how to use the optimizers
# for C++ objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(Rcpp)
library(lessSEM)

linreg <- '
// [[Rcpp::depends(RcppArmadillo)]]
#include <RcppArmadillo.h>

// [[Rcpp::export]]
double fitfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
  // extract all required elements:
  arma::colvec b = Rcpp::as<arma::colvec>(parameters);
```

```

arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

// compute the sum of squared errors:
arma::mat sse = arma::trans(y-X*b)*(y-X*b);

// other packages, such as glmnet, scale the sse with
// 1/(2*N), where N is the sample size. We will do that here as well

sse *= 1.0/(2.0 * y.n_elem);

// note: We must return a double, but the sse is a matrix
// To get a double, just return the single value that is in
// this matrix:
return(sse(0,0));
}

// [[Rcpp::export]]
arma::rowvec gradientfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
  // extract all required elements:
  arma::colvec b = Rcpp::as<arma::colvec>(parameters);
  arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
  arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

  // note: we want to return our gradients as row-vector; therefore,
  // we have to transpose the resulting column-vector:
  arma::rowvec gradients = arma::trans(-2.0*X.t() * y + 2.0*X.t()*X*b);

  // other packages, such as glmnet, scale the sse with
  // 1/(2*N), where N is the sample size. We will do that here as well

  gradients *= (.5/y.n_rows);

  return(gradients);
}

// Dirk Eddelbuettel at
// https://gallery.rcpp.org/articles/passing-cpp-function-pointers/
typedef double (*fitFunPtr)(const Rcpp::NumericVector&, //parameters
                           Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<fitFunPtr> fitFunPtr_t;

typedef arma::rowvec (*gradientFunPtr)(const Rcpp::NumericVector&, //parameters
                                       Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<gradientFunPtr> gradientFunPtr_t;

// [[Rcpp::export]]
fitFunPtr_t fitfunPtr() {
  return(fitFunPtr_t(new fitFunPtr(&fitfunction)));
}

```

```

// [[Rcpp::export]]
gradientFunPtr_t gradfunPtr() {
  return(gradientFunPtr_t(new gradientFunPtr(&gradientfunction)));
}

Rcpp::sourceCpp(code = linreg)

ffp <- fitfunPtr()
gfp <- gradfunPtr()

N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p),nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
      rep(0,6)) # true regression weights
y <- X%%matrix(b,ncol = 1) + rnorm(N,0,.2)

data <- list("y" = y,
            "X" = cbind(1,X))
parameters <- rep(0, ncol(data$X))
names(parameters) <- paste0("b", 0:(length(parameters)-1))

all <- gpAdaptiveLassoCpp(par = parameters,
                        regularized = paste0("b", 1:(length(b)-1)),
                        fn = ffp,
                        gr = gfp,
                        lambdas = seq(0,1,.1),
                        additionalArguments = data)

all@parameters

```

gpCappedL1

gpCappedL1

Description

Implements cappedL1 regularization for general purpose optimization problems. The penalty function is given by:

$$p(x_j) = \lambda \min(|x_j|, \theta)$$

where $\theta > 0$. The cappedL1 penalty is identical to the lasso for parameters which are below θ and identical to a constant for parameters above θ . As adding a constant to the fitting function will not change its minimum, larger parameters can stay unregularized while smaller ones are set to zero.

Usage

```
gpCappedL1(
  par,
```

```

    fn,
    gr = NULL,
    ...,
    regularized,
    lambdas,
    thetas,
    method = "glmnet",
    control = lessSEM::controlGlmnet()
)

```

Arguments

<code>par</code>	labeled vector with starting values
<code>fn</code>	R function which takes the parameters AND their labels as input and returns the fit value (a single value)
<code>gr</code>	R function which takes the parameters AND their labels as input and returns the gradients of the objective function. If set to NULL, numDeriv will be used to approximate the gradients
<code>...</code>	additional arguments passed to <code>fn</code> and <code>gr</code>
<code>regularized</code>	vector with names of parameters which are to be regularized.
<code>lambdas</code>	numeric vector: values for the tuning parameter lambda
<code>thetas</code>	parameters whose absolute value is above this threshold will be penalized with a constant (theta)
<code>method</code>	which optimizer should be used? Currently implemented are ista and glmnet.
<code>control</code>	used to control the optimizer. This element is generated with the <code>controlIsta</code> and <code>controlGlmnet</code> functions. See <code>?controlIsta</code> and <code>?controlGlmnet</code> for more details.

Details

The interface is similar to that of `optim`. Users have to supply a vector with starting values (important: This vector *must* have labels) and a fitting function. This fitting functions *must* take a labeled vector with parameter values as first argument. The remaining arguments are passed with the `...` argument. This is similar to `optim`.

The gradient function `gr` is optional. If set to NULL, the **numDeriv** package will be used to approximate the gradients. Supplying a gradient function can result in considerable speed improvements.

CappedL1 regularization:

- Zhang, T. (2010). Analysis of Multi-stage Convex Relaxation for Sparse Regularization. *Journal of Machine Learning Research*, 11, 1081–1107.

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.

- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class gpRegularized

Examples

```
# This example shows how to use the optimizers
# for other objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

# This example shows how to use the optimizers
# for other objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(lessSEM)
set.seed(123)

# first, we simulate data for our
# linear regression.
N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p),nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
        rep(0,6)) # true regression weights
y <- X%*%matrix(b,ncol = 1) + rnorm(N,0,.2)

# First, we must construct a fitting function
# which returns a single value. We will use
# the residual sum squared as fitting function.

# Let's start setting up the fitting function:
fittingFunction <- function(par, y, X, N){
  # par is the parameter vector
```

```

# y is the observed dependent variable
# X is the design matrix
# N is the sample size
pred <- X %%% matrix(par, ncol = 1) #be explicit here:
# we need par to be a column vector
sse <- sum((y - pred)^2)
# we scale with .5/N to get the same results as glmnet
return((.5/N)*sse)
}

# let's define the starting values:
b <- c(solve(t(X)%%X)%%t(X)%%y) # we will use the lm estimates
names(b) <- paste0("b", 1:length(b))
# names of regularized parameters
regularized <- paste0("b",1:p)

# optimize
cL1 <- gpCappedL1(
  par = b,
  regularized = regularized,
  fn = fittingFunction,
  lambdas = seq(0,1,.1),
  thetas = c(0.001, .5, 1),
  X = X,
  y = y,
  N = N
)

# optional: plot requires plotly package
# plot(cL1)

# for comparison

fittingFunction <- function(par, y, X, N, lambda, theta){
  pred <- X %%% matrix(par, ncol = 1)
  sse <- sum((y - pred)^2)
  smoothAbs <- sqrt(par^2 + 1e-8)
  pen <- lambda * ifelse(smoothAbs < theta, smoothAbs, theta)
  return((.5/N)*sse + sum(pen))
}

round(
  optim(par = b,
    fn = fittingFunction,
    y = y,
    X = X,
    N = N,
    lambda = cL1@fits$lambda[15],
    theta = cL1@fits$theta[15],
    method = "BFGS")$par,
  4)
cL1@parameters[15,]

```

gpCappedL1Cpp

*gpCappedL1Cpp***Description**

Implements cappedL1 regularization for general purpose optimization problems with C++ functions. The penalty function is given by:

$$p(x_j) = \lambda \min(|x_j|, \theta)$$

where $\theta > 0$. The cappedL1 penalty is identical to the lasso for parameters which are below θ and identical to a constant for parameters above θ . As adding a constant to the fitting function will not change its minimum, larger parameters can stay unregularized while smaller ones are set to zero.

Usage

```
gpCappedL1Cpp(
  par,
  fn,
  gr,
  additionalArguments,
  regularized,
  lambdas,
  thetas,
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)
```

Arguments

par	labeled vector with starting values
fn	R function which takes the parameters AND their labels as input and returns the fit value (a single value)
gr	R function which takes the parameters AND their labels as input and returns the gradients of the objective function. If set to NULL, numDeriv will be used to approximate the gradients
additionalArguments	list with additional arguments passed to fn and gr
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use getLavaanParameters(model) with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
thetas	parameters whose absolute value is above this threshold will be penalized with a constant (theta)
method	which optimizer should be used? Currently implemented are ista and glmnet.
control	used to control the optimizer. This element is generated with the controlIsta and controlGlmnet functions. See ?controlIsta and ?controlGlmnet for more details.

Details

The interface is inspired by `optim`, but a bit more restrictive. Users have to supply a vector with starting values (important: This vector *must* have labels), a fitting function, and a gradient function. These fitting functions *must* take an `const Rcpp::NumericVector&` with parameter values as first argument and an `Rcpp::List&` as second argument

CappedL1 regularization:

- Zhang, T. (2010). Analysis of Multi-stage Convex Relaxation for Sparse Regularization. *Journal of Machine Learning Research*, 11, 1081–1107.

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class `gpRegularized`

Examples

```
# This example shows how to use the optimizers
# for C++ objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(Rcpp)
library(lessSEM)

linreg <- '
// [[Rcpp::depends(RcppArmadillo)]]
#include <RcppArmadillo.h>
```

```

// [[Rcpp::export]]
double fitfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
  // extract all required elements:
  arma::colvec b = Rcpp::as<arma::colvec>(parameters);
  arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
  arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

  // compute the sum of squared errors:
  arma::mat sse = arma::trans(y-X*b)*(y-X*b);

  // other packages, such as glmnet, scale the sse with
  // 1/(2*N), where N is the sample size. We will do that here as well

  sse *= 1.0/(2.0 * y.n_elem);

  // note: We must return a double, but the sse is a matrix
  // To get a double, just return the single value that is in
  // this matrix:
  return(sse(0,0));
}

// [[Rcpp::export]]
arma::rowvec gradientfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
  // extract all required elements:
  arma::colvec b = Rcpp::as<arma::colvec>(parameters);
  arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
  arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

  // note: we want to return our gradients as row-vector; therefore,
  // we have to transpose the resulting column-vector:
  arma::rowvec gradients = arma::trans(-2.0*X.t() * y + 2.0*X.t()*X*b);

  // other packages, such as glmnet, scale the sse with
  // 1/(2*N), where N is the sample size. We will do that here as well

  gradients *= (.5/y.n_rows);

  return(gradients);
}

// https://gallery.rcpp.org/articles/passing-cpp-function-pointers/
typedef double (*fitFunPtr)(const Rcpp::NumericVector&, //parameters
                           Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<fitFunPtr> fitFunPtr_t;

typedef arma::rowvec (*gradientFunPtr)(const Rcpp::NumericVector&, //parameters
                                       Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<gradientFunPtr> gradientFunPtr_t;

// [[Rcpp::export]]
fitFunPtr_t fitfunPtr() {

```

```

        return(fitFunPtr_t(new fitFunPtr(&fitfunction)));
    }

    // [[Rcpp::export]]
    gradientFunPtr_t gradfunPtr() {
        return(gradientFunPtr_t(new gradientFunPtr(&gradientfunction)));
    }
    ,

Rcpp::sourceCpp(code = linreg)

ffp <- fitfunPtr()
gfp <- gradfunPtr()

N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p),nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
       rep(0,6)) # true regression weights
y <- X%%matrix(b,ncol = 1) + rnorm(N,0,.2)

data <- list("y" = y,
            "X" = cbind(1,X))
parameters <- rep(0, ncol(data$X))
names(parameters) <- paste0("b", 0:(length(parameters)-1))

cL1 <- gpCappedL1Cpp(par = parameters,
                    regularized = paste0("b", 1:(length(b)-1)),
                    fn = ffp,
                    gr = gfp,
                    lambdas = seq(0,1,.1),
                    thetas = seq(0.1,1,.1),
                    additionalArguments = data)

cL1@parameters

```

gpElasticNet

gpElasticNet

Description

Implements elastic net regularization for general purpose optimization problems. The penalty function is given by:

$$p(x_j) = p(x_j) = \frac{1}{w_j} \lambda |x_j|$$

Note that the elastic net combines ridge and lasso regularization. If $\alpha = 0$, the elastic net reduces to ridge regularization. If $\alpha = 1$ it reduces to lasso regularization. In between, elastic net is a compromise between the shrinkage of the lasso and the ridge penalty.

Usage

```
gpElasticNet(
  par,
  regularized,
  fn,
  gr = NULL,
  lambdas,
  alphas,
  ...,
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)
```

Arguments

<code>par</code>	labeled vector with starting values
<code>regularized</code>	vector with names of parameters which are to be regularized.
<code>fn</code>	R function which takes the parameters AND their labels as input and returns the fit value (a single value)
<code>gr</code>	R function which takes the parameters AND their labels as input and returns the gradients of the objective function. If set to NULL, numDeriv will be used to approximate the gradients
<code>lambdas</code>	numeric vector: values for the tuning parameter lambda
<code>alphas</code>	numeric vector with values of the tuning parameter alpha. Must be between 0 and 1. 0 = ridge, 1 = lasso.
<code>...</code>	additional arguments passed to <code>fn</code> and <code>gr</code>
<code>method</code>	which optimizer should be used? Currently implemented are <code>ista</code> and <code>glmnet</code> .
<code>control</code>	used to control the optimizer. This element is generated with the <code>controlIsta</code> and <code>controlGlmnet</code> functions. See <code>?controlIsta</code> and <code>?controlGlmnet</code> for more details.

Details

The interface is similar to that of `optim`. Users have to supply a vector with starting values (important: This vector *must* have labels) and a fitting function. This fitting functions *must* take a labeled vector with parameter values as first argument. The remaining arguments are passed with the `...` argument. This is similar to `optim`.

The gradient function `gr` is optional. If set to NULL, the **numDeriv** package will be used to approximate the gradients. Supplying a gradient function can result in considerable speed improvements.

Elastic net regularization:

- Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2), 301–320. <https://doi.org/10.1111/j.1467-9868.2005.00503.x>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class gpRegularized

Examples

```
# This example shows how to use the optimizers
# for other objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(lessSEM)
set.seed(123)

# first, we simulate data for our
# linear regression.
N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p), nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
       rep(0,6)) # true regression weights
y <- X%*%matrix(b, ncol = 1) + rnorm(N, 0, .2)

# First, we must construct a fitting function
# which returns a single value. We will use
# the residual sum squared as fitting function.

# Let's start setting up the fitting function:
fittingFunction <- function(par, y, X, N){
  # par is the parameter vector
  # y is the observed dependent variable
```

```

# X is the design matrix
# N is the sample size
pred <- X %*% matrix(par, ncol = 1) #be explicit here:
# we need par to be a column vector
sse <- sum((y - pred)^2)
# we scale with .5/N to get the same results as glmnet
return((.5/N)*sse)
}

# let's define the starting values:
b <- c(solve(t(X)%*%X)%*%t(X)%*%y) # we will use the lm estimates
names(b) <- paste0("b", 1:length(b))
# names of regularized parameters
regularized <- paste0("b",1:p)

# optimize
elasticNetPen <- gpElasticNet(
  par = b,
  regularized = regularized,
  fn = fittingFunction,
  lambdas = seq(0,1,.1),
  alphas = c(0, .5, 1),
  X = X,
  y = y,
  N = N
)

# optional: plot requires plotly package
# plot(elasticNetPen)

# for comparison:
fittingFunction <- function(par, y, X, N, lambda, alpha){
  pred <- X %*% matrix(par, ncol = 1)
  sse <- sum((y - pred)^2)
  return((.5/N)*sse + (1-alpha)*lambda * sum(par^2) + alpha*lambda *sum(sqrt(par^2 + 1e-8)))
}

round(
  optim(par = b,
    fn = fittingFunction,
    y = y,
    X = X,
    N = N,
    lambda = elasticNetPen@fits$lambda[15],
    alpha = elasticNetPen@fits$alpha[15],
    method = "BFGS")$par,
  4)
elasticNetPen@parameters[15,]

```

Description

Implements elastic net regularization for general purpose optimization problems with C++ functions. The penalty function is given by:

$$p(x_j) = p(x_j) = \frac{1}{w_j} \lambda |x_j|$$

Note that the elastic net combines ridge and lasso regularization. If $\alpha = 0$, the elastic net reduces to ridge regularization. If $\alpha = 1$ it reduces to lasso regularization. In between, elastic net is a compromise between the shrinkage of the lasso and the ridge penalty.

Usage

```
gpElasticNetCpp(
  par,
  regularized,
  fn,
  gr,
  lambdas,
  alphas,
  additionalArguments,
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)
```

Arguments

par	labeled vector with starting values
regularized	vector with names of parameters which are to be regularized.
fn	R function which takes the parameters AND their labels as input and returns the fit value (a single value)
gr	R function which takes the parameters AND their labels as input and returns the gradients of the objective function. If set to NULL, numDeriv will be used to approximate the gradients
lambdas	numeric vector: values for the tuning parameter lambda
alphas	numeric vector with values of the tuning parameter alpha. Must be between 0 and 1. 0 = ridge, 1 = lasso.
additionalArguments	list with additional arguments passed to fn and gr
method	which optimizer should be used? Currently implemented are ista and glmnet.
control	used to control the optimizer. This element is generated with the controlIsta and controlGlmnet functions. See ?controlIsta and ?controlGlmnet for more details.

Details

The interface is inspired by optim, but a bit more restrictive. Users have to supply a vector with starting values (important: This vector *must* have labels), a fitting function, and a gradient function.

These fitting functions *must* take an `const Rcpp::NumericVector&` with parameter values as first argument and an `Rcpp::List&` as second argument

Elastic net regularization:

- Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2), 301–320. <https://doi.org/10.1111/j.1467-9868.2005.00503.x>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class `gpRegularized`

Examples

```
# This example shows how to use the optimizers
# for C++ objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(Rcpp)
library(lessSEM)

linreg <- '
// [[Rcpp::depends(RcppArmadillo)]]
#include <RcppArmadillo.h>

// [[Rcpp::export]]
double fitfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
```

```

// extract all required elements:
arma::colvec b = Rcpp::as<arma::colvec>(parameters);
arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

// compute the sum of squared errors:
arma::mat sse = arma::trans(y-X*b)*(y-X*b);

// other packages, such as glmnet, scale the sse with
// 1/(2*N), where N is the sample size. We will do that here as well

sse *= 1.0/(2.0 * y.n_elem);

// note: We must return a double, but the sse is a matrix
// To get a double, just return the single value that is in
// this matrix:
return(sse(0,0));
}

// [[Rcpp::export]]
arma::rowvec gradientfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
  // extract all required elements:
  arma::colvec b = Rcpp::as<arma::colvec>(parameters);
  arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
  arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

  // note: we want to return our gradients as row-vector; therefore,
  // we have to transpose the resulting column-vector:
  arma::rowvec gradients = arma::trans(-2.0*X.t() * y + 2.0*X.t()*X*b);

  // other packages, such as glmnet, scale the sse with
  // 1/(2*N), where N is the sample size. We will do that here as well

  gradients *= (.5/y.n_rows);

  return(gradients);
}

// Dirk Eddelbuettel at
// https://gallery.rcpp.org/articles/passing-cpp-function-pointers/
typedef double (*fitFunPtr)(const Rcpp::NumericVector&, //parameters
                             Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<fitFunPtr> fitFunPtr_t;

typedef arma::rowvec (*gradientFunPtr)(const Rcpp::NumericVector&, //parameters
                                       Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<gradientFunPtr> gradientFunPtr_t;

// [[Rcpp::export]]
fitFunPtr_t fitfunPtr() {
  return(fitFunPtr_t(new fitFunPtr(&fitfunction)));
}

```

```

}

// [[Rcpp::export]]
gradientFunPtr_t gradfunPtr() {
    return(gradientFunPtr_t(new gradientFunPtr(&gradientfunction)));
}
'

Rcpp::sourceCpp(code = linreg)

ffp <- fitfunPtr()
gfp <- gradfunPtr()

N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p),nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
       rep(0,6)) # true regression weights
y <- X%%matrix(b,ncol = 1) + rnorm(N,0,.2)

data <- list("y" = y,
            "X" = cbind(1,X))
parameters <- rep(0, ncol(data$X))
names(parameters) <- paste0("b", 0:(length(parameters)-1))

en <- gpElasticNetCpp(par = parameters,
                     regularized = paste0("b", 1:(length(b)-1)),
                     fn = ffp,
                     gr = gfp,
                     lambdas = seq(0,1,.1),
                     alphas = c(0,.5,1),
                     additionalArguments = data)

en@parameters

```

gpLasso

gpLasso

Description

Implements lasso regularization for general purpose optimization problems. The penalty function is given by:

$$p(x_j) = \lambda |x_j|$$

Lasso regularization will set parameters to zero if λ is large enough

Usage

```
gpLasso(
```

```

    par,
    regularized,
    fn,
    gr = NULL,
    lambdas = NULL,
    nLambdas = NULL,
    reverse = TRUE,
    curve = 1,
    ...,
    method = "glmnet",
    control = lessSEM::controlGlmnet()
)

```

Arguments

par	labeled vector with starting values
regularized	vector with names of parameters which are to be regularized.
fn	R function which takes the parameters as input and returns the fit value (a single value)
gr	R function which takes the parameters as input and returns the gradients of the objective function. If set to NULL, numDeriv will be used to approximate the gradients
lambdas	numeric vector: values for the tuning parameter lambda
nLambdas	alternative to lambda: If alpha = 1, lessSEM can automatically compute the first lambda value which sets all regularized parameters to zero. It will then generate nLambda values between 0 and the computed lambda.
reverse	if set to TRUE and nLambdas is used, lessSEM will start with the largest lambda and gradually decrease lambda. Otherwise, lessSEM will start with the smallest lambda and gradually increase it.
curve	Allows for unequally spaced lambda steps (e.g., .01,.02,.05,1,5,20). If curve is close to 1 all lambda values will be equally spaced, if curve is large lambda values will be more concentrated close to 0. See ?lessSEM::curveLambda for more information.
...	additional arguments passed to fn and gr
method	which optimizer should be used? Currently implemented are ista and glmnet.
control	used to control the optimizer. This element is generated with the controlIsta and controlGlmnet functions. See ?controlIsta and ?controlGlmnet for more details.

Details

The interface is similar to that of `optim`. Users have to supply a vector with starting values (important: This vector *must* have labels) and a fitting function. This fitting functions *must* take a labeled vector with parameter values as first argument. The remaining arguments are passed with the `...` argument. This is similar to `optim`.

The gradient function `gr` is optional. If set to NULL, the **numDeriv** package will be used to approximate the gradients. Supplying a gradient function can result in considerable speed improvements.

Lasso regularization:

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

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class gpRegularized

Examples

```
# This example shows how to use the optimizers
# for other objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(lessSEM)
set.seed(123)

# first, we simulate data for our
# linear regression.
N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p), nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
        rep(0,6)) # true regression weights
y <- X%*%matrix(b, ncol = 1) + rnorm(N, 0, .2)

# First, we must construct a fitting function
```

```

# which returns a single value. We will use
# the residual sum squared as fitting function.

# Let's start setting up the fitting function:
fittingFunction <- function(par, y, X, N){
  # par is the parameter vector
  # y is the observed dependent variable
  # X is the design matrix
  # N is the sample size
  pred <- X %*% matrix(par, ncol = 1) #be explicit here:
  # we need par to be a column vector
  sse <- sum((y - pred)^2)
  # we scale with .5/N to get the same results as glmnet
  return((.5/N)*sse)
}

# let's define the starting values:
b <- rep(0,p)
names(b) <- paste0("b", 1:length(b))
# names of regularized parameters
regularized <- paste0("b",1:p)

# optimize
lassoPen <- gpLasso(
  par = b,
  regularized = regularized,
  fn = fittingFunction,
  nLambdas = 100,
  X = X,
  y = y,
  N = N
)
plot(lassoPen)

# You can access the fit results as follows:
lassoPen@fits
# Note that we won't compute any fit measures automatically, as
# we cannot be sure how the AIC, BIC, etc are defined for your objective function

```

gpLassoCpp

gpLassoCpp

Description

Implements lasso regularization for general purpose optimization problems with C++ functions. The penalty function is given by:

$$p(x_j) = \lambda |x_j|$$

Lasso regularization will set parameters to zero if λ is large enough

Usage

```
gpLassoCpp(
  par,
  regularized,
  fn,
  gr,
  lambdas = NULL,
  nLambdas = NULL,
  curve = 1,
  additionalArguments,
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)
```

Arguments

par	labeled vector with starting values
regularized	vector with names of parameters which are to be regularized.
fn	pointer to Rcpp function which takes the parameters as input and returns the fit value (a single value)
gr	pointer to Rcpp function which takes the parameters as input and returns the gradients of the objective function.
lambdas	numeric vector: values for the tuning parameter lambda
nLambdas	alternative to lambda: If alpha = 1, lessSEM can automatically compute the first lambda value which sets all regularized parameters to zero. It will then generate nLambda values between 0 and the computed lambda.
curve	Allows for unequally spaced lambda steps (e.g., .01,.02,.05,1,5,20). If curve is close to 1 all lambda values will be equally spaced, if curve is large lambda values will be more concentrated close to 0. See ?lessSEM::curveLambda for more information.
additionalArguments	list with additional arguments passed to fn and gr
method	which optimizer should be used? Currently implemented are ista and glmnet.
control	used to control the optimizer. This element is generated with the controlIsta and controlGlmnet functions. See ?controlIsta and ?controlGlmnet for more details.

Details

The interface is inspired by optim, but a bit more restrictive. Users have to supply a vector with starting values (important: This vector *must* have labels), a fitting function, and a gradient function. These fitting functions *must* take an `const Rcpp::NumericVector&` with parameter values as first argument and an `Rcpp::List&` as second argument

Lasso regularization:

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

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class gpRegularized

Examples

```
# This example shows how to use the optimizers
# for C++ objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(Rcpp)
library(lessSEM)

linreg <- '
// [[Rcpp::depends(RcppArmadillo)]]
#include <RcppArmadillo.h>

// [[Rcpp::export]]
double fitfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
  // extract all required elements:
  arma::colvec b = Rcpp::as<arma::colvec>(parameters);
  arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
  arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

  // compute the sum of squared errors:
  arma::mat sse = arma::trans(y-X*b)*(y-X*b);

  // other packages, such as glmnet, scale the sse with
```

```

    // 1/(2*N), where N is the sample size. We will do that here as well
    sse *= 1.0/(2.0 * y.n_elem);

    // note: We must return a double, but the sse is a matrix
    // To get a double, just return the single value that is in
    // this matrix:
    return(sse(0,0));
}

// [[Rcpp::export]]
arma::rowvec gradientfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
    // extract all required elements:
    arma::colvec b = Rcpp::as<arma::colvec>(parameters);
    arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
    arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

    // note: we want to return our gradients as row-vector; therefore,
    // we have to transpose the resulting column-vector:
    arma::rowvec gradients = arma::trans(-2.0*X.t() * y + 2.0*X.t()*X*b);

    // other packages, such as glmnet, scale the sse with
    // 1/(2*N), where N is the sample size. We will do that here as well

    gradients *= (.5/y.n_rows);

    return(gradients);
}

// Dirk Eddelbuettel at
// https://gallery.rcpp.org/articles/passing-cpp-function-pointers/
typedef double (*fitFunPtr)(const Rcpp::NumericVector&, //parameters
    Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<fitFunPtr> fitFunPtr_t;

typedef arma::rowvec (*gradientFunPtr)(const Rcpp::NumericVector&, //parameters
    Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<gradientFunPtr> gradientFunPtr_t;

// [[Rcpp::export]]
fitFunPtr_t fitfunPtr() {
    return(fitFunPtr_t(new fitFunPtr(&fitfunction)));
}

// [[Rcpp::export]]
gradientFunPtr_t gradfunPtr() {
    return(gradientFunPtr_t(new gradientFunPtr(&gradientfunction)));
}

```

Rcpp::sourceCpp(code = linreg)

```

ffp <- fitfunPtr()
gfp <- gradfunPtr()

N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p),nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
       rep(0,6)) # true regression weights
y <- X%%matrix(b,ncol = 1) + rnorm(N,0,.2)

data <- list("y" = y,
            "X" = cbind(1,X))
parameters <- rep(0, ncol(data$X))
names(parameters) <- paste0("b", 0:(length(parameters)-1))

l1 <- gpLassoCpp(par = parameters,
                regularized = paste0("b", 1:(length(b)-1)),
                fn = ffp,
                gr = gfp,
                lambdas = seq(0,1,.1),
                additionalArguments = data)

l1@parameters

```

gpLsp

gpLsp

Description

Implements lsp regularization for general purpose optimization problems. The penalty function is given by:

Usage

```

gpLsp(
  par,
  fn,
  gr = NULL,
  ...,
  regularized,
  lambdas,
  thetas,
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)

```

Arguments

<code>par</code>	labeled vector with starting values
<code>fn</code>	R function which takes the parameters AND their labels as input and returns the fit value (a single value)
<code>gr</code>	R function which takes the parameters AND their labels as input and returns the gradients of the objective function. If set to NULL, <code>numDeriv</code> will be used to approximate the gradients
<code>...</code>	additional arguments passed to <code>fn</code> and <code>gr</code>
<code>regularized</code>	vector with names of parameters which are to be regularized.
<code>lambdas</code>	numeric vector: values for the tuning parameter <code>lambda</code>
<code>thetas</code>	numeric vector: values for the tuning parameter <code>theta</code>
<code>method</code>	which optimizer should be used? Currently implemented are <code>ista</code> and <code>glmnet</code> .
<code>control</code>	used to control the optimizer. This element is generated with the <code>controlIsta</code> and <code>controlGlmnet</code> functions. See <code>?controlIsta</code> and <code>?controlGlmnet</code> for more details.

Details

The interface is similar to that of `optim`. Users have to supply a vector with starting values (important: This vector must have labels) and a fitting function. This fitting functions must take a labeled vector with parameter values as first argument. The remaining arguments are passed with the `...` argument. This is similar to `optim`.

The gradient function `gr` is optional. If set to NULL, the **numDeriv** package will be used to approximate the gradients. Supplying a gradient function can result in considerable speed improvements.

Lsp regularization:

- Candès, E. J., Wakin, M. B., & Boyd, S. P. (2008). Enhancing Sparsity by Reweighted ℓ_1 Minimization. *Journal of Fourier Analysis and Applications*, 14(5–6), 877–905. <https://doi.org/10.1007/s00041-008-9045-x>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale ℓ_1 -regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for ℓ_1 -regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class gpRegularized

Examples

```
library(lessSEM)
set.seed(123)

# first, we simulate data for our
# linear regression.
N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p), nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
        rep(0,6)) # true regression weights
y <- X%%matrix(b, ncol = 1) + rnorm(N, 0, .2)

# First, we must construct a fitting function
# which returns a single value. We will use
# the residual sum squared as fitting function.

# Let's start setting up the fitting function:
fittingFunction <- function(par, y, X, N){
  # par is the parameter vector
  # y is the observed dependent variable
  # X is the design matrix
  # N is the sample size
  pred <- X %% matrix(par, ncol = 1) #be explicit here:
  # we need par to be a column vector
  sse <- sum((y - pred)^2)
  # we scale with .5/N to get the same results as glmnet
  return((.5/N)*sse)
}

# let's define the starting values:
b <- c(solve(t(X)%%X)%%t(X)%%y) # we will use the lm estimates
names(b) <- paste0("b", 1:length(b))
# names of regularized parameters
regularized <- paste0("b", 1:p)

# optimize
lspPen <- gpLsp(
  par = b,
  regularized = regularized,
  fn = fittingFunction,
  lambdas = seq(0, 1, .1),
  thetas = c(0.001, .5, 1),
  X = X,
  y = y,
  N = N
)
```

```

# optional: plot requires plotly package
# plot(lspPen)

# for comparison

fittingFunction <- function(par, y, X, N, lambda, theta){
  pred <- X %%% matrix(par, ncol = 1)
  sse <- sum((y - pred)^2)
  smoothAbs <- sqrt(par^2 + 1e-8)
  pen <- lambda * log(1.0 + smoothAbs / theta)
  return((.5/N)*sse + sum(pen))
}

round(
  optim(par = b,
        fn = fittingFunction,
        y = y,
        X = X,
        N = N,
        lambda = lspPen@fits$lambda[15],
        theta = lspPen@fits$theta[15],
        method = "BFGS")$par,
  4)
lspPen@parameters[15,]

```

gpLspCpp

gpLspCpp

Description

Implements lsp regularization for general purpose optimization problems with C++ functions. The penalty function is given by:

$$p(x_j) = \lambda \log(1 + |x_j|/\theta)$$

where $\theta > 0$.

Usage

```

gpLspCpp(
  par,
  fn,
  gr,
  additionalArguments,
  regularized,
  lambdas,
  thetas,
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)

```

Arguments

<code>par</code>	labeled vector with starting values
<code>fn</code>	R function which takes the parameters AND their labels as input and returns the fit value (a single value)
<code>gr</code>	R function which takes the parameters AND their labels as input and returns the gradients of the objective function. If set to NULL, <code>numDeriv</code> will be used to approximate the gradients
<code>additionalArguments</code>	list with additional arguments passed to <code>fn</code> and <code>gr</code>
<code>regularized</code>	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
<code>lambdas</code>	numeric vector: values for the tuning parameter λ
<code>thetas</code>	numeric vector: values for the tuning parameter θ
<code>method</code>	which optimizer should be used? Currently implemented are <code>ista</code> and <code>glmnet</code> .
<code>control</code>	used to control the optimizer. This element is generated with the <code>controlIsta</code> and <code>controlGlmnet</code> functions. See <code>?controlIsta</code> and <code>?controlGlmnet</code> for more details.

Details

The interface is inspired by `optim`, but a bit more restrictive. Users have to supply a vector with starting values (important: This vector must have labels), a fitting function, and a gradient function. These fitting functions must take an `const Rcpp::NumericVector&` with parameter values as first argument and an `Rcpp::List&` as second argument

lsp regularization:

- Candès, E. J., Wakin, M. B., & Boyd, S. P. (2008). Enhancing Sparsity by Reweighted ℓ_1 Minimization. *Journal of Fourier Analysis and Applications*, 14(5–6), 877–905. <https://doi.org/10.1007/s00041-008-9045-x>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale ℓ_1 -regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for ℓ_1 -regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.

- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. Foundations and Trends in Optimization, 1(3), 123–231.

Value

Object of class gpRegularized

Examples

```
# This example shows how to use the optimizers
# for C++ objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(Rcpp)
library(lessSEM)

linreg <- '
// [[Rcpp::depends(RcppArmadillo)]]
#include <RcppArmadillo.h>

// [[Rcpp::export]]
double fitfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
  // extract all required elements:
  arma::colvec b = Rcpp::as<arma::colvec>(parameters);
  arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
  arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

  // compute the sum of squared errors:
  arma::mat sse = arma::trans(y-X*b)*(y-X*b);

  // other packages, such as glmnet, scale the sse with
  // 1/(2*N), where N is the sample size. We will do that here as well

  sse *= 1.0/(2.0 * y.n_elem);

  // note: We must return a double, but the sse is a matrix
  // To get a double, just return the single value that is in
  // this matrix:
  return(sse(0,0));
}

// [[Rcpp::export]]
arma::rowvec gradientfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
  // extract all required elements:
  arma::colvec b = Rcpp::as<arma::colvec>(parameters);
  arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
  arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

  // note: we want to return our gradients as row-vector; therefore,
  // we have to transpose the resulting column-vector:
```

```

    arma::rowvec gradients = arma::trans(-2.0*X.t() * y + 2.0*X.t()*X*b);

    // other packages, such as glmnet, scale the sse with
    // 1/(2*N), where N is the sample size. We will do that here as well

    gradients *= (.5/y.n_rows);

    return(gradients);
}

// Dirk Eddelbuettel at
// https://gallery.rcpp.org/articles/passing-cpp-function-pointers/
typedef double (*fitFunPtr)(const Rcpp::NumericVector&, //parameters
                           Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<fitFunPtr> fitFunPtr_t;

typedef arma::rowvec (*gradientFunPtr)(const Rcpp::NumericVector&, //parameters
                                       Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<gradientFunPtr> gradientFunPtr_t;

// [[Rcpp::export]]
fitFunPtr_t fitfunPtr() {
    return(fitFunPtr_t(new fitFunPtr(&fitfunction)));
}

// [[Rcpp::export]]
gradientFunPtr_t gradfunPtr() {
    return(gradientFunPtr_t(new gradientFunPtr(&gradientfunction)));
}

Rcpp::sourceCpp(code = linreg)

ffp <- fitfunPtr()
gfp <- gradfunPtr()

N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p),nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
       rep(0,6)) # true regression weights
y <- X%%matrix(b,ncol = 1) + rnorm(N,0,.2)

data <- list("y" = y,
            "X" = cbind(1,X))
parameters <- rep(0, ncol(data$X))
names(parameters) <- paste0("b", 0:(length(parameters)-1))

l <- gpLspCpp(par = parameters,
             regularized = paste0("b", 1:(length(b)-1)),
             fn = ffp,

```

```

      gr = gfp,
      lambdas = seq(0,1,.1),
      thetas = seq(0.1,1,.1),
      additionalArguments = data)

l@parameters

```

gpMcp

gpMcp

Description

Implements mcp regularization for general purpose optimization problems. The penalty function is given by:

Equation Omitted in Pdf Documentation.

Usage

```

gpMcp(
  par,
  fn,
  gr = NULL,
  ...,
  regularized,
  lambdas,
  thetas,
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)

```

Arguments

par	labeled vector with starting values
fn	R function which takes the parameters AND their labels as input and returns the fit value (a single value)
gr	R function which takes the parameters AND their labels as input and returns the gradients of the objective function. If set to NULL, numDeriv will be used to approximate the gradients
...	additional arguments passed to fn and gr
regularized	vector with names of parameters which are to be regularized.
lambdas	numeric vector: values for the tuning parameter lambda
thetas	numeric vector: values for the tuning parameter theta
method	which optimizer should be used? Currently implemented are ista and glmnet.
control	used to control the optimizer. This element is generated with the controlIsta and controlGlmnet functions. See ?controlIsta and ?controlGlmnet for more details.

Details

The interface is similar to that of `optim`. Users have to supply a vector with starting values (important: This vector *must* have labels) and a fitting function. This fitting functions *must* take a labeled vector with parameter values as first argument. The remaining arguments are passed with the `...` argument. This is similar to `optim`.

The gradient function `gr` is optional. If set to `NULL`, the **numDeriv** package will be used to approximate the gradients. Supplying a gradient function can result in considerable speed improvements.

`mcp` regularization:

- Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2), 894–942. <https://doi.org/10.1214/09-AOS729>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class `gpRegularized`

Examples

```
# This example shows how to use the optimizers
# for other objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(lessSEM)
set.seed(123)

# first, we simulate data for our
```

```

# linear regression.
N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p),nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
        rep(0,6)) # true regression weights
y <- X%%matrix(b,ncol = 1) + rnorm(N,0,.2)

# First, we must construct a fitting function
# which returns a single value. We will use
# the residual sum squared as fitting function.

# Let's start setting up the fitting function:
fittingFunction <- function(par, y, X, N){
  # par is the parameter vector
  # y is the observed dependent variable
  # X is the design matrix
  # N is the sample size
  pred <- X %% matrix(par, ncol = 1) #be explicit here:
  # we need par to be a column vector
  sse <- sum((y - pred)^2)
  # we scale with .5/N to get the same results as glmnet
  return((.5/N)*sse)
}

# let's define the starting values:
# first, let's add an intercept
X <- cbind(1, X)

b <- c(solve(t(X)%%X)%%t(X)%%y) # we will use the lm estimates
names(b) <- paste0("b", 0:(length(b)-1))
# names of regularized parameters
regularized <- paste0("b",1:p)

# optimize
mcpPen <- gpMcp(
  par = b,
  regularized = regularized,
  fn = fittingFunction,
  lambdas = seq(0,1,.1),
  thetas = c(1.001, 1.5, 2),
  X = X,
  y = y,
  N = N
)

# optional: plot requires plotly package
# plot(mcpPen)

```

Description

Implements mcp regularization for general purpose optimization problems with C++ functions. The penalty function is given by:

Equation Omitted in Pdf Documentation.

Usage

```
gpMcpCpp(
  par,
  fn,
  gr,
  additionalArguments,
  regularized,
  lambdas,
  thetas,
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)
```

Arguments

par	labeled vector with starting values
fn	R function which takes the parameters AND their labels as input and returns the fit value (a single value)
gr	R function which takes the parameters AND their labels as input and returns the gradients of the objective function. If set to NULL, numDeriv will be used to approximate the gradients
additionalArguments	list with additional arguments passed to fn and gr
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use getLavaanParameters(model) with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
thetas	numeric vector: values for the tuning parameter theta
method	which optimizer should be used? Currently implemented are ista and glmnet.
control	used to control the optimizer. This element is generated with the controlIsta and controlGlmnet functions. See ?controlIsta and ?controlGlmnet for more details.

Details

The interface is inspired by optim, but a bit more restrictive. Users have to supply a vector with starting values (important: This vector *must* have labels), a fitting function, and a gradient function. These fitting functions *must* take an `const Rcpp::NumericVector&` with parameter values as first argument and an `Rcpp::List&` as second argument

mcp regularization:

- Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2), 894–942. <https://doi.org/10.1214/09-AOS729>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class `gpRegularized`

Examples

```
# This example shows how to use the optimizers
# for C++ objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(Rcpp)
library(lessSEM)

linreg <- '
// [[Rcpp::depends(RcppArmadillo)]]
#include <RcppArmadillo.h>

// [[Rcpp::export]]
double fitfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
  // extract all required elements:
  arma::colvec b = Rcpp::as<arma::colvec>(parameters);
  arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
  arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

  // compute the sum of squared errors:
```

```

    arma::mat sse = arma::trans(y-X*b)*(y-X*b);

    // other packages, such as glmnet, scale the sse with
    // 1/(2*N), where N is the sample size. We will do that here as well

    sse *= 1.0/(2.0 * y.n_elem);

    // note: We must return a double, but the sse is a matrix
    // To get a double, just return the single value that is in
    // this matrix:
    return(sse(0,0));
}

// [[Rcpp::export]]
arma::rowvec gradientfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
    // extract all required elements:
    arma::colvec b = Rcpp::as<arma::colvec>(parameters);
    arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
    arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

    // note: we want to return our gradients as row-vector; therefore,
    // we have to transpose the resulting column-vector:
    arma::rowvec gradients = arma::trans(-2.0*X.t() * y + 2.0*X.t()*X*b);

    // other packages, such as glmnet, scale the sse with
    // 1/(2*N), where N is the sample size. We will do that here as well

    gradients *= (.5/y.n_rows);

    return(gradients);
}

// Dirk Eddelbuettel at
// https://gallery.rcpp.org/articles/passing-cpp-function-pointers/
typedef double (*fitFunPtr)(const Rcpp::NumericVector&, //parameters
                           Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<fitFunPtr> fitFunPtr_t;

typedef arma::rowvec (*gradientFunPtr)(const Rcpp::NumericVector&, //parameters
                                       Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<gradientFunPtr> gradientFunPtr_t;

// [[Rcpp::export]]
fitFunPtr_t fitfunPtr() {
    return(fitFunPtr_t(new fitFunPtr(&fitfunction)));
}

// [[Rcpp::export]]
gradientFunPtr_t gradfunPtr() {
    return(gradientFunPtr_t(new gradientFunPtr(&gradientfunction)));
}

```

```

,

Rcpp::sourceCpp(code = linreg)

ffp <- fitfunPtr()
gfp <- gradfunPtr()

N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p),nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
      rep(0,6)) # true regression weights
y <- X%%matrix(b,ncol = 1) + rnorm(N,0,.2)

data <- list("y" = y,
            "X" = cbind(1,X))
parameters <- rep(0, ncol(data$X))
names(parameters) <- paste0("b", 0:(length(parameters)-1))

m <- gpMcpCpp(par = parameters,
             regularized = paste0("b", 1:(length(b)-1)),
             fn = ffp,
             gr = gfp,
             lambdas = seq(0,1,.1),
             thetas = seq(.1,1,.1),
             additionalArguments = data)

m@parameters

```

gpRegularized-class	<i>Class for regularized model using general purpose optimization interface</i>
---------------------	---

Description

Class for regularized model using general purpose optimization interface

Slots

penalty penalty used (e.g., "lasso")
parameters data.frame with all parameter estimates
fits data.frame with all fit results
parameterLabels character vector with names of all parameters
weights vector with weights given to each of the parameters in the penalty
regularized character vector with names of regularized parameters
internalOptimization list of elements used internally
inputArguments list with elements passed by the user to the general purpose optimizer

gpRidge

*gpRidge***Description**

Implements ridge regularization for general purpose optimization problems. The penalty function is given by:

$$p(x_j) = \lambda x_j^2$$

Note that ridge regularization will not set any of the parameters to zero but result in a shrinkage towards zero.

Usage

```
gpRidge(
  par,
  regularized,
  fn,
  gr = NULL,
  lambdas,
  ...,
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)
```

Arguments

<code>par</code>	labeled vector with starting values
<code>regularized</code>	vector with names of parameters which are to be regularized.
<code>fn</code>	R function which takes the parameters as input and returns the fit value (a single value)
<code>gr</code>	R function which takes the parameters as input and returns the gradients of the objective function. If set to <code>NULL</code> , <code>numDeriv</code> will be used to approximate the gradients
<code>lambdas</code>	numeric vector: values for the tuning parameter <code>lambda</code>
<code>...</code>	additional arguments passed to <code>fn</code> and <code>gr</code>
<code>method</code>	which optimizer should be used? Currently implemented are <code>ista</code> and <code>glmnet</code> .
<code>control</code>	used to control the optimizer. This element is generated with the <code>controlIsta</code> and <code>controlGlmnet</code> functions. See <code>?controlIsta</code> and <code>?controlGlmnet</code> for more details.

Details

The interface is similar to that of `optim`. Users have to supply a vector with starting values (important: This vector *must* have labels) and a fitting function. This fitting functions *must* take a labeled vector with parameter values as first argument. The remaining arguments are passed with the `...` argument. This is similar to `optim`.

The gradient function `gr` is optional. If set to `NULL`, the **numDeriv** package will be used to approximate the gradients. Supplying a gradient function can result in considerable speed improvements.

Ridge regularization:

- Hoerl, A. E., & Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1), 55–67. <https://doi.org/10.1080/00401706.1970.10488634>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class `gpRegularized`

Examples

```
# This example shows how to use the optimizers
# for other objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(lessSEM)
set.seed(123)

# first, we simulate data for our
# linear regression.
N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p), nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
      rep(0,6)) # true regression weights
y <- X%*%matrix(b, ncol = 1) + rnorm(N, 0, .2)
```

```

# First, we must construct a fitting function
# which returns a single value. We will use
# the residual sum squared as fitting function.

# Let's start setting up the fitting function:
fittingFunction <- function(par, y, X, N){
  # par is the parameter vector
  # y is the observed dependent variable
  # X is the design matrix
  # N is the sample size
  pred <- X %*% matrix(par, ncol = 1) #be explicit here:
  # we need par to be a column vector
  sse <- sum((y - pred)^2)
  # we scale with .5/N to get the same results as glmnet
  return((.5/N)*sse)
}

# let's define the starting values:
b <- c(solve(t(X)%*%X)%*%t(X)%*%y) # we will use the lm estimates
names(b) <- paste0("b", 1:length(b))
# names of regularized parameters
regularized <- paste0("b",1:p)

# optimize
ridgePen <- gpRidge(
  par = b,
  regularized = regularized,
  fn = fittingFunction,
  lambdas = seq(0,1,.01),
  X = X,
  y = y,
  N = N
)
plot(ridgePen)

# for comparison:
# fittingFunction <- function(par, y, X, N, lambda){
#   pred <- X %*% matrix(par, ncol = 1)
#   sse <- sum((y - pred)^2)
#   return((.5/N)*sse + lambda * sum(par^2))
# }
#
# optim(par = b,
#       fn = fittingFunction,
#       y = y,
#       X = X,
#       N = N,
#       lambda = ridgePen@fits$lambda[20],
#       method = "BFGS")$par
# ridgePen@parameters[20,]

```

gpRidgeCpp

*gpRidgeCpp***Description**

Implements ridge regularization for general purpose optimization problems with C++ functions. The penalty function is given by:

$$p(x_j) = \lambda x_j^2$$

Note that ridge regularization will not set any of the parameters to zero but result in a shrinkage towards zero.

Usage

```
gpRidgeCpp(
  par,
  regularized,
  fn,
  gr,
  lambdas,
  additionalArguments,
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)
```

Arguments

par	labeled vector with starting values
regularized	vector with names of parameters which are to be regularized.
fn	R function which takes the parameters as input and returns the fit value (a single value)
gr	R function which takes the parameters as input and returns the gradients of the objective function. If set to NULL, numDeriv will be used to approximate the gradients
lambdas	numeric vector: values for the tuning parameter lambda
additionalArguments	list with additional arguments passed to fn and gr
method	which optimizer should be used? Currently implemented are ista and glmnet.
control	used to control the optimizer. This element is generated with the controlIsta and controlGlmnet functions. See ?controlIsta and ?controlGlmnet for more details.

Details

The interface is inspired by `optim`, but a bit more restrictive. Users have to supply a vector with starting values (important: This vector *must* have labels), a fitting function, and a gradient function. These fitting functions *must* take an `const Rcpp::NumericVector&` with parameter values as first argument and an `Rcpp::List&` as second argument

Ridge regularization:

- Hoerl, A. E., & Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1), 55–67. <https://doi.org/10.1080/00401706.1970.10488634>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class `gpRegularized`

Examples

```
# This example shows how to use the optimizers
# for C++ objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(Rcpp)
library(lessSEM)

linreg <- '
// [[Rcpp::depends(RcppArmadillo)]]
#include <RcppArmadillo.h>
```

```

// [[Rcpp::export]]
double fitfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
  // extract all required elements:
  arma::colvec b = Rcpp::as<arma::colvec>(parameters);
  arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
  arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

  // compute the sum of squared errors:
  arma::mat sse = arma::trans(y-X*b)*(y-X*b);

  // other packages, such as glmnet, scale the sse with
  // 1/(2*N), where N is the sample size. We will do that here as well

  sse *= 1.0/(2.0 * y.n_elem);

  // note: We must return a double, but the sse is a matrix
  // To get a double, just return the single value that is in
  // this matrix:
  return(sse(0,0));
}

// [[Rcpp::export]]
arma::rowvec gradientfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
  // extract all required elements:
  arma::colvec b = Rcpp::as<arma::colvec>(parameters);
  arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
  arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

  // note: we want to return our gradients as row-vector; therefore,
  // we have to transpose the resulting column-vector:
  arma::rowvec gradients = arma::trans(-2.0*X.t() * y + 2.0*X.t()*X*b);

  // other packages, such as glmnet, scale the sse with
  // 1/(2*N), where N is the sample size. We will do that here as well

  gradients *= (.5/y.n_rows);

  return(gradients);
}

// https://gallery.rcpp.org/articles/passing-cpp-function-pointers/
typedef double (*fitFunPtr)(const Rcpp::NumericVector&, //parameters
                           Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<fitFunPtr> fitFunPtr_t;

typedef arma::rowvec (*gradientFunPtr)(const Rcpp::NumericVector&, //parameters
                                       Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<gradientFunPtr> gradientFunPtr_t;

// [[Rcpp::export]]
fitFunPtr_t fitfunPtr() {

```

```

        return(fitFunPtr_t(new fitFunPtr(&fitfunction)));
    }

    // [[Rcpp::export]]
    gradientFunPtr_t gradfunPtr() {
        return(gradientFunPtr_t(new gradientFunPtr(&gradientfunction)));
    }
    ,

Rcpp::sourceCpp(code = linreg)

ffp <- fitfunPtr()
gfp <- gradfunPtr()

N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p),nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
       rep(0,6)) # true regression weights
y <- X%%matrix(b,ncol = 1) + rnorm(N,0,.2)

data <- list("y" = y,
            "X" = cbind(1,X))
parameters <- rep(0, ncol(data$X))
names(parameters) <- paste0("b", 0:(length(parameters)-1))

r <- gpRidgeCpp(par = parameters,
               regularized = paste0("b", 1:(length(b)-1)),
               fn = ffp,
               gr = gfp,
               lambdas = seq(0,1,.1),
               additionalArguments = data)

r@parameters

```

gpScad

*gpScad***Description**

Implements scad regularization for general purpose optimization problems. The penalty function is given by:

Equation Omitted in Pdf Documentation.

Usage

```
gpScad(
  par,
```

```

    fn,
    gr = NULL,
    ...,
    regularized,
    lambdas,
    thetas,
    method = "glmnet",
    control = lessSEM::controlGlmnet()
  )

```

Arguments

<code>par</code>	labeled vector with starting values
<code>fn</code>	R function which takes the parameters AND their labels as input and returns the fit value (a single value)
<code>gr</code>	R function which takes the parameters AND their labels as input and returns the gradients of the objective function. If set to <code>NULL</code> , <code>numDeriv</code> will be used to approximate the gradients
<code>...</code>	additional arguments passed to <code>fn</code> and <code>gr</code>
<code>regularized</code>	vector with names of parameters which are to be regularized.
<code>lambdas</code>	numeric vector: values for the tuning parameter <code>lambda</code>
<code>thetas</code>	numeric vector: values for the tuning parameter <code>theta</code>
<code>method</code>	which optimizer should be used? Currently implemented are <code>ista</code> and <code>glmnet</code> .
<code>control</code>	used to control the optimizer. This element is generated with the <code>controlIsta</code> and <code>controlGlmnet</code> functions. See <code>?controlIsta</code> and <code>?controlGlmnet</code> for more details.

Details

The interface is similar to that of `optim`. Users have to supply a vector with starting values (important: This vector *must* have labels) and a fitting function. This fitting functions *must* take a labeled vector with parameter values as first argument. The remaining arguments are passed with the `...` argument. This is similar to `optim`.

The gradient function `gr` is optional. If set to `NULL`, the **numDeriv** package will be used to approximate the gradients. Supplying a gradient function can result in considerable speed improvements.

scad regularization:

- 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. <https://doi.org/10.1198/016214501753>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.

- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class gpRegularized

Examples

```
# This example shows how to use the optimizers
# for other objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(lessSEM)
set.seed(123)

# first, we simulate data for our
# linear regression.
N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p),nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),
        rep(0,6)) # true regression weights
y <- X%%matrix(b,ncol = 1) + rnorm(N,0,.2)

# First, we must construct a fitting function
# which returns a single value. We will use
# the residual sum squared as fitting function.

# Let's start setting up the fitting function:
fittingFunction <- function(par, y, X, N){
  # par is the parameter vector
  # y is the observed dependent variable
  # X is the design matrix
  # N is the sample size
  pred <- X %% matrix(par, ncol = 1) #be explicit here:
  # we need par to be a column vector
  sse <- sum((y - pred)^2)
  # we scale with .5/N to get the same results as glmnet
```

```

    return((.5/N)*sse)
  }

  # let's define the starting values:
  # first, let's add an intercept
  X <- cbind(1, X)

  b <- c(solve(t(X)%*%X)%*%t(X)%*%y) # we will use the lm estimates
  names(b) <- paste0("b", 0:(length(b)-1))
  # names of regularized parameters
  regularized <- paste0("b", 1:p)

  # optimize
  scadPen <- gpScad(
    par = b,
    regularized = regularized,
    fn = fittingFunction,
    lambdas = seq(0,1,.1),
    thetas = c(2.001, 2.5, 5),
    X = X,
    y = y,
    N = N
  )

  # optional: plot requires plotly package
  # plot(scadPen)

  # for comparison
  #library(ncvreg)
  #scadFit <- ncvreg(X = X[,-1],
  #                  y = y,
  #                  penalty = "SCAD",
  #                  lambda = scadPen@fits$lambda[15],
  #                  gamma = scadPen@fits$theta[15])
  #coef(scadFit)
  #scadPen@parameters[15,]

```

gpScadCpp

gpScadCpp

Description

Implements scad regularization for general purpose optimization problems with C++ functions. The penalty function is given by:

Equation Omitted in Pdf Documentation.

Usage

```

gpScadCpp(
  par,

```

```

    fn,
    gr,
    additionalArguments,
    regularized,
    lambdas,
    thetas,
    method = "glmnet",
    control = lessSEM::controlGlmnet()
)

```

Arguments

par	labeled vector with starting values
fn	R function which takes the parameters AND their labels as input and returns the fit value (a single value)
gr	R function which takes the parameters AND their labels as input and returns the gradients of the objective function. If set to NULL, numDeriv will be used to approximate the gradients
additionalArguments	list with additional arguments passed to fn and gr
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
thetas	numeric vector: values for the tuning parameter theta
method	which optimizer should be used? Currently implemented are ista and glmnet.
control	used to control the optimizer. This element is generated with the <code>controlIsta</code> and <code>controlGlmnet</code> functions. See <code>?controlIsta</code> and <code>?controlGlmnet</code> for more details.

Details

The interface is inspired by `optim`, but a bit more restrictive. Users have to supply a vector with starting values (important: This vector *must* have labels), a fitting function, and a gradient function. These fitting functions *must* take an `const Rcpp::NumericVector&` with parameter values as first argument and an `Rcpp::List&` as second argument

scad regularization:

- 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. <https://doi.org/10.1198/016214501753>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.

- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Object of class gpRegularized

Examples

```
# This example shows how to use the optimizers
# for C++ objective functions. We will use
# a linear regression as an example. Note that
# this is not a useful application of the optimizers
# as there are specialized packages for linear regression
# (e.g., glmnet)

library(Rcpp)
library(lessSEM)

linreg <- '
// [[Rcpp::depends(RcppArmadillo)]]
#include <RcppArmadillo.h>

// [[Rcpp::export]]
double fitfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
  // extract all required elements:
  arma::colvec b = Rcpp::as<arma::colvec>(parameters);
  arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
  arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

  // compute the sum of squared errors:
  arma::mat sse = arma::trans(y-X*b)*(y-X*b);

  // other packages, such as glmnet, scale the sse with
  // 1/(2*N), where N is the sample size. We will do that here as well

  sse *= 1.0/(2.0 * y.n_elem);

  // note: We must return a double, but the sse is a matrix
  // To get a double, just return the single value that is in
  // this matrix:
  return(sse(0,0));
```

```

}

// [[Rcpp::export]]
arma::rowvec gradientfunction(const Rcpp::NumericVector& parameters, Rcpp::List& data){
  // extract all required elements:
  arma::colvec b = Rcpp::as<arma::colvec>(parameters);
  arma::colvec y = Rcpp::as<arma::colvec>(data["y"]); // the dependent variable
  arma::mat X = Rcpp::as<arma::mat>(data["X"]); // the design matrix

  // note: we want to return our gradients as row-vector; therefore,
  // we have to transpose the resulting column-vector:
  arma::rowvec gradients = arma::trans(-2.0*X.t() * y + 2.0*X.t()*X*b);

  // other packages, such as glmnet, scale the sse with
  // 1/(2*N), where N is the sample size. We will do that here as well

  gradients *= (.5/y.n_rows);

  return(gradients);
}

// Dirk Eddelbuettel at
// https://gallery.rcpp.org/articles/passing-cpp-function-pointers/
typedef double (*fitFunPtr)(const Rcpp::NumericVector&, //parameters
                             Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<fitFunPtr> fitFunPtr_t;

typedef arma::rowvec (*gradientFunPtr)(const Rcpp::NumericVector&, //parameters
                                       Rcpp::List& //additional elements
);
typedef Rcpp::XPtr<gradientFunPtr> gradientFunPtr_t;

// [[Rcpp::export]]
fitFunPtr_t fitfunPtr() {
  return(fitFunPtr_t(new fitFunPtr(&fitfunction)));
}

// [[Rcpp::export]]
gradientFunPtr_t gradfunPtr() {
  return(gradientFunPtr_t(new gradientFunPtr(&gradientfunction)));
}

Rcpp::sourceCpp(code = linreg)

ffp <- fitfunPtr()
gfp <- gradfunPtr()

N <- 100 # number of persons
p <- 10 # number of predictors
X <- matrix(rnorm(N*p),nrow = N, ncol = p) # design matrix
b <- c(rep(1,4),

```

```

      rep(0,6)) # true regression weights
y <- X%%matrix(b,ncol = 1) + rnorm(N,0,.2)

data <- list("y" = y,
            "X" = cbind(1,X))
parameters <- rep(0, ncol(data$X))
names(parameters) <- paste0("b", 0:(length(parameters)-1))

s <- gpScadCpp(par = parameters,
              regularized = paste0("b", 1:(length(b)-1)),
              fn = ffp,
              gr = gfp,
              lambdas = seq(0,1,.1),
              thetas = seq(2.1,3,.1),
              additionalArguments = data)

s@parameters

```

istaCappedL1mgSEM	<i>cappedL1 optimization with ista</i>
-------------------	--

Description

Object for elastic net optimization with ista optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta value, a lambda and an alpha value (alpha must be 1).

istaCappedL1SEM	<i>cappedL1 optimization with ista</i>
-----------------	--

Description

Object for elastic net optimization with ista optimizer

Value

a list with fit results

Fields

- `new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements
- `optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta value, a lambda and an alpha value (alpha must be 1).

istaEnetGeneralPurpose

elastic net optimization with ista

Description

Object for elastic net optimization with ista optimizer

Value

a list with fit results

Fields

- `new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements
- `optimize` optimize the model. Expects a vector with starting values, an R function to compute the fit, an R function to compute the gradients, a list with elements the fit and gradient function require, a lambda and an alpha value.

istaEnetGeneralPurposeCpp

elastic net optimization with ista

Description

Object for elastic net optimization with ista optimizer

Value

a list with fit results

Fields

- `new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements
- `optimize` optimize the model. Expects a vector with starting values, a SEXP function pointer to compute the fit, a SEXP function pointer to compute the gradients, a list with elements the fit and gradient function require, a lambda and an alpha value.

<code>istaEnetMgSEM</code>	<i>elastic net optimization with ista optimizer</i>
----------------------------	---

Description

Object for elastic net optimization with glmnet optimizer

Value

a list with fit results

Fields

- `new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements
- `optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a lambda and an alpha value.

<code>istaEnetSEM</code>	<i>elastic net optimization with ista optimizer</i>
--------------------------	---

Description

Object for elastic net optimization with glmnet optimizer

Value

a list with fit results

Fields

- `new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements
- `optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a lambda and an alpha value.

istaLSPMgSEM	<i>lsp optimization with ista</i>
--------------	-----------------------------------

Description

Object for lsp optimization with ista optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

istaLSPSEM	<i>lsp optimization with ista</i>
------------	-----------------------------------

Description

Object for lsp optimization with ista optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

istaMcpMgSEM	<i>mcp optimization with ista</i>
--------------	-----------------------------------

Description

Object for mcp optimization with ista optimizer

Value

a list with fit results

Fields

- new creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements
- optimize optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

istaMcpSEM	<i>mcp optimization with ista</i>
------------	-----------------------------------

Description

Object for mcp optimization with ista optimizer

Value

a list with fit results

Fields

- new creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements
- optimize optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

istaMixedPenaltyGeneralPurpose
mixed penalty optimization with ista

Description

Object for elastic net optimization with ista optimizer

Value

a list with fit results

Fields

new creates a new object.

optimize optimize the model.

istaMixedPenaltyGeneralPurposeCpp
mixed penalty optimization with ista

Description

Object for elastic net optimization with ista optimizer

Value

a list with fit results

Fields

new creates a new object. Requires (1) a vector with weights for each parameter, (2) a vector indicating which penalty is used, and (3) a list with control elements

optimize optimize the model.

<code>istaMixedPenaltymgSEM</code>	<i>mixed penalty optimization with ista</i>
------------------------------------	---

Description

Object for elastic net optimization with ista optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (1) a vector with weights for each parameter, (2) a vector indicating which penalty is used, and (3) a list with control elements

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta value, a lambda and an alpha value (alpha must be 1).

<code>istaMixedPenaltySEM</code>	<i>mixed penalty optimization with ista</i>
----------------------------------	---

Description

Object for elastic net optimization with ista optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (1) a vector with weights for each parameter, (2) a vector indicating which penalty is used, and (3) a list with control elements

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta value, a lambda and an alpha value (alpha must be 1).

istaScadMgSEM	<i>scad optimization with ista</i>
---------------	------------------------------------

Description

Object for scad optimization with ista optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

istaScadSEM	<i>scad optimization with ista</i>
-------------	------------------------------------

Description

Object for scad optimization with ista optimizer

Value

a list with fit results

Fields

`new` creates a new object. Requires (1) a vector with weights for each parameter and (2) a list with control elements

`optimize` optimize the model. Expects a vector with starting values, a SEM of type SEM_Cpp, a theta and a lambda value.

lasso

*lasso***Description**

Implements lasso regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \lambda|x_j|$$

Lasso regularization will set parameters to zero if λ is large enough

Usage

```
lasso(
  lavaanModel,
  regularized,
  lambdas = NULL,
  nLambdas = NULL,
  reverse = TRUE,
  curve = 1,
  method = "glmnet",
  modifyModel = lessSEM::modifyModel(),
  control = lessSEM::controlGlmnet()
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
nLambdas	alternative to lambda: If $\alpha = 1$, lessSEM can automatically compute the first lambda value which sets all regularized parameters to zero. It will then generate nLambda values between 0 and the computed lambda.
reverse	if set to TRUE and nLambdas is used, lessSEM will start with the largest lambda and gradually decrease lambda. Otherwise, lessSEM will start with the smallest lambda and gradually increase it.
curve	Allows for unequally spaced lambda steps (e.g., .01,.02,.05,1,5,20). If curve is close to 1 all lambda values will be equally spaced, if curve is large lambda values will be more concentrated close to 0. See <code>?lessSEM::curveLambda</code> for more information.
method	which optimizer should be used? Currently implemented are ista and glmnet. With ista, the control argument can be used to switch to related procedures (currently gist).
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
control	used to control the optimizer. This element is generated with the <code>controlIsta</code> and <code>controlGlmnet</code> functions. See <code>?controlIsta</code> and <code>?controlGlmnet</code> for more details.

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

Lasso regularization:

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

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Model of class `regularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.
```

```

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f ~~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- lasso(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("1", 6:15),
  # in case of lasso and adaptive lasso, we can specify the number of lambda
  # values to use. lessSEM will automatically find lambda_max and fit
  # models for nLambda values between 0 and lambda_max. For the other
  # penalty functions, lambdas must be specified explicitly
  nLambdas = 50)

# use the plot-function to plot the regularized parameters:
plot(lsem)

# the coefficients can be accessed with:
coef(lsem)
# if you are only interested in the estimates and not the tuning parameters, use
coef(lsem)$estimates
# or
estimates(lsem)

# elements of lsem can be accessed with the @ operator:
lsem$parameters[1,]

# fit Measures:
fitIndices(lsem)

# The best parameters can also be extracted with:
coef(lsem, criterion = "AIC")
# or
estimates(lsem, criterion = "AIC")

#### Advanced ####
# Switching the optimizer #
# Use the "method" argument to switch the optimizer. The control argument
# must also be changed to the corresponding function:

```

```
lsemIsta <- lasso(
  lavaanModel = lavaanModel,
  regularized = paste0("l", 6:15),
  nLambdas = 50,
  method = "ista",
  control = controlIsta())

# Note: The results are basically identical:
lsemIsta@parameters - lsem@parameters
```

lavaan2lsxLabels	<i>lavaan2lsxLabels</i>
------------------	-------------------------

Description

helper function: lsx and lavaan use slightly different parameter labels. This function can be used to get both sets of labels.

Usage

```
lavaan2lsxLabels(lavaanModel)
```

Arguments

lavaanModel model of class lavaan

Value

list with lavaan labels and lsx labels

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
  data = dataset,
  meanstructure = TRUE,
```

```
std.lv = TRUE)

lavaan2ls1xLabels(lavaanModel)
```

lessSEM2Lavaan	<i>lessSEM2Lavaan</i>
----------------	-----------------------

Description

Creates a lavaan model object from lessSEM (only if possible). Pass either a criterion or a combination of lambda, alpha, and theta.

Usage

```
lessSEM2Lavaan(
  regularizedSEM,
  criterion = NULL,
  lambda = NULL,
  alpha = NULL,
  theta = NULL
)
```

Arguments

- regularizedSEM object created with lessSEM
- criterion criterion used for model selection. Currently supported are "AIC" or "BIC"
- lambda value for tuning parameter lambda
- alpha value for tuning parameter alpha
- theta value for tuning parameter theta

Value

lavaan model

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
     16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
     111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
```

```
f ~~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:
regularized <- lasso(lavaanModel,
                    regularized = paste0("1", 11:15),
                    lambdas = seq(0,1,.1))

# using criterion
lessSEM2Lavaan(regularizedSEM = regularized,
               criterion = "AIC")

# using tuning parameters (note: we only have to specify the tuning
# parameters that are actually used by the penalty function. In case
# of lasso, this is lambda):
lessSEM2Lavaan(regularizedSEM = regularized,
               lambda = 1)
```

lessSEMCoef-class	<i>Class for the coefficients estimated by lessSEM.</i>
-------------------	---

Description

Class for the coefficients estimated by lessSEM.

Slots

- tuningParameters tuning parameters
- estimates parameter estimates
- transformations transformations of parameters

loadings	<i>loadings</i>
----------	-----------------

Description

Extract the labels of all loadings found in a lavaan model.

Usage

loadings(lavaanModel)

Arguments

lavaanModel fitted lavaan model

Value

vector with parameter labels

Examples

```
# The following is adapted from ?lavaan::sem
library(lessSEM)
model <- '
  # latent variable definitions
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + a*y2 + b*y3 + c*y4
  dem65 =~ y5 + a*y6 + b*y7 + c*y8

  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60

  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
  '

fit <- sem(model, data = PoliticalDemocracy)

loadings(fit)
```

logicalMatch	<i>logicalMatch</i>
--------------	---------------------

Description

Returns the rows for which all elements of a boolean matrix X are equal to the elements in boolean vector x

Usage

```
logicalMatch(X, x)
```

Arguments

X matrix with booleans
x vector of booleans

Value

numerical vector with indices of matching rows

logLik,Rcpp_mgSEM-method
<i>logLik</i>

Description

logLik

Usage

```
## S4 method for signature 'Rcpp_mgSEM'
logLik(object, ...)
```

Arguments

object	object of class Rcpp_mgSEM
...	not used

Value

log-likelihood of the model

logLik,Rcpp_SEMCpp-method
<i>logLik</i>

Description

logLik

Usage

```
## S4 method for signature 'Rcpp_SEMCpp'
logLik(object, ...)
```

Arguments

object	object of class Rcpp_SEMCpp
...	not used

Value

log-likelihood of the model

logLikelihood-class	<i>Class for log-likelihood of regularized SEM. Note: we define a custom logLik - Function because the generic one is using df = number of parameters which might be confusing.</i>
---------------------	---

Description

Class for log-likelihood of regularized SEM. Note: we define a custom logLik - Function because the generic one is using df = number of parameters which might be confusing.

Slots

logLik log-Likelihood
 nParameters number of parameters in the model
 N number of persons in the data set

lsp	<i>lsp</i>
-----	------------

Description

Implements lsp regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \lambda \log(1 + |x_j|/\theta)$$

where $\theta > 0$.

Usage

```
lsp(
  lavaanModel,
  regularized,
  lambdas,
  thetas,
  modifyModel = lessSEM::modifyModel(),
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
thetas	parameters whose absolute value is above this threshold will be penalized with a constant (theta)
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
method	which optimizer should be used? Currently implemented are <code>ista</code> and <code>glmnet</code> . With <code>ista</code> , the <code>control</code> argument can be used to switch to related procedures
control	used to control the optimizer. This element is generated with the <code>controlIsta</code> (see <code>?controlIsta</code>)

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

lsp regularization:

- Candès, E. J., Wakin, M. B., & Boyd, S. P. (2008). Enhancing Sparsity by Reweighted l1 Minimization. *Journal of Fourier Analysis and Applications*, 14(5–6), 877–905. <https://doi.org/10.1007/s00041-008-9045-x>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>

- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Model of class `regularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- lsp(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("1", 6:15),
  lambdas = seq(0,1,length.out = 20),
  thetas = seq(0.01,2,length.out = 5))

# the coefficients can be accessed with:
coef(lsem)
# if you are only interested in the estimates and not the tuning parameters, use
coef(lsem)$estimates
# or
estimates(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters[1,]
```

```
# fit Measures:
fitIndices(lsem)

# The best parameters can also be extracted with:
coef(lsem, criterion = "AIC")
# or
estimates(lsem, criterion = "AIC")

# optional: plotting the paths requires installation of plotly
# plot(lsem)
```

*makePtrs**makePtrs*

Description

This function helps you create the pointers necessary to use the Cpp interface

Usage

```
makePtrs(fitFunName, gradFunName)
```

Arguments

fitFunName	name of your C++ fit function (IMPORTANT: This must be the name used in C++)
gradFunName	name of your C++ gradient function (IMPORTANT: This must be the name used in C++)

Value

a string which can be copied in the C++ function to create the pointers.

Examples

```
# see vignette("General-Purpose-Optimization", package = "lessSEM") for an example
```

mcp

mcp

Description

Implements mcp regularization for structural equation models. The penalty function is given by:
Equation Omitted in Pdf Documentation.

Usage

```
mcp(
  lavaanModel,
  regularized,
  lambdas,
  thetas,
  modifyModel = lessSEM::modifyModel(),
  method = "ista",
  control = lessSEM::controlIsta()
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
thetas	parameters whose absolute value is above this threshold will be penalized with a constant (theta)
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
method	which optimizer should be used? Currently implemented are <code>ista</code> and <code>glmnet</code> . With <code>ista</code> , the <code>control</code> argument can be used to switch to related procedures (currently <code>gist</code>).
control	used to control the optimizer. This element is generated with the <code>controlIsta</code> (see <code>?controlIsta</code>)

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

In our experience, the `glmnet` optimizer can run in issues with the mcp penalty. Therefor, we default to using `ista`.

mcp regularization:

- Zhang, C.-H. (2010). Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2), 894–942. <https://doi.org/10.1214/09-AOS729>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Model of class regularizedSEM

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"
```

```

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- mcp(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("l", 6:15),
  lambdas = seq(0,1,length.out = 20),
  thetas = seq(0.01,2,length.out = 5))

# the coefficients can be accessed with:
coef(lsem)

# if you are only interested in the estimates and not the tuning parameters, use
coef(lsem)@estimates
# or
estimates(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters[1,]

# fit Measures:
fitIndices(lsem)

# The best parameters can also be extracted with:
coef(lsem, criterion = "AIC")
# or
estimates(lsem, criterion = "AIC")

# optional: plotting the paths requires installation of plotly
# plot(lsem)

```

mcpPenalty_C

mcpPenalty_C

Description

mcpPenalty_C

Usage

mcpPenalty_C(par, lambda_p, theta)

Arguments

par	single parameter value
lambda_p	lambda value for this parameter
theta	theta value for this parameter

Value

penalty value

mgSEM	<i>mgSEM class</i>
-------	--------------------

Description

internal mgSEM representation

Fields

`new` Creates a new mgSEM.

`addModel` add a model. Expects `Rcpp::List`

`addTransformation` adds transformations to a model

`implied` Computes implied means and covariance matrix

`fit` Fits the model. Returns objective value of the fitting function

`getParameters` Returns a data frame with model parameters.

`getParameterLabels` Returns a vector with unique parameter labels as used internally.

`getEstimator` Returns a vector with names of the estimators used in the submodels.

`getGradients` Returns a matrix with scores.

`getScores` Returns a matrix with scores. Not yet implemented

`getHessian` Returns the hessian of the model. Expects the labels of the parameters and the values of the parameters as well as a boolean indicating if these are raw. Finally, a double (eps) controls the precision of the approximation.

`computeTransformations` compute the transformations.

`setTransformationGradientStepSize` change the step size of the gradient computation for the transformations

mixedPenalty

mixedPenalty

Description

Provides possibility to impose different penalties on different parameters.

Usage

```
mixedPenalty(
  lavaanModel,
  modifyModel = lessSEM::modifyModel(),
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)
```

Arguments

lavaanModel	model of class lavaan
modifyModel	used to modify the lavaanModel. See ?modifyModel.
method	which optimizer should be used? Currently supported are "glmnet" and "ista".
control	used to control the optimizer. This element is generated with the controlIsta and controlGlmnet functions. See ?controlIsta and ?controlGlmnet for more details.

Details

The mixedPenalty function allows you to add multiple penalties to a single model. For instance, you may want to regularize both loadings and regressions in a SEM. In this case, using the same penalty (e.g., lasso) for both types of penalties may actually not be what you want to use because the penalty function is sensitive to the scales of the parameters. Instead, you may want to use two separate lasso penalties for loadings and regressions. Similarly, separate penalties for different parameters have, for instance, been proposed in multi-group models (Geminiani et al., 2021).

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well. Models are fitted with the glmnet or ista optimizer. Note that the optimizers differ in which penalties they support. The following table provides an overview:

Penalty	Function	glmnet	ista
lasso	addLasso	x	x
elastic net	addElasticNet	x*	-
cappedL1	addCappedL1	x	x
lsp	addLsp	x	x
scad	addScad	x	x
mcp	addMcp	x	x

By default, glmnet will be used. Note that the elastic net penalty can only be combined with other elastic net penalties.

Check vignette(topic = "Mixed-Penalties", package = "lessSEM") for more details.

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Geminiani, E., Marra, G., & Moustaki, I. (2021). Single- and multiple-group penalized factor analysis: A trust-region algorithm approach with integrated automatic multiple tuning parameter selection. *Psychometrika*, 86(1), 65–95. <https://doi.org/10.1007/s11336-021-09751-8>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Model of class regularizedSEM

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
```

```

meanstructure = TRUE,
std.lv = TRUE)

# Regularization:

# In this example, we want to regularize the loadings 16-110
# independently of the loadings 111-15. This could, for instance,
# reflect that the items y6-y10 and y11-y15 may belong to different
# subscales.

regularized <- lavaanModel |>
  # create template for regularized model with mixed penalty:
  mixedPenalty() |>
  # add lasso penalty on loadings 16 - 110:
  addLasso(regularized = paste0("1", 6:10),
           lambdas = seq(0,1,length.out = 4)) |>
  # add scad penalty on loadings 111 - 115:
  addScad(regularized = paste0("1", 11:15),
          lambdas = seq(0,1,length.out = 3),
          thetas = 3.1) |>
  # fit the model:
  fit()

# elements of regularized can be accessed with the @ operator:
regularized@parameters[1,]

# AIC and BIC:
AIC(regularized)
BIC(regularized)

# The best parameters can also be extracted with:
coef(regularized, criterion = "AIC")
coef(regularized, criterion = "BIC")

# The tuningParameterConfiguration corresponds to the rows
# in the lambda, theta, and alpha matrices in regularized@tuningParameterConfigurations.
# Configuration 3, for example, is given by
regularized@tuningParameterConfigurations$lambda[3,]
regularized@tuningParameterConfigurations$theta[3,]
regularized@tuningParameterConfigurations$alpha[3,]
# Note that lambda, theta, and alpha may correspond to tuning parameters
# of different penalties for different parameters (e.g., lambda for 16 is the lambda
# of the lasso penalty, while lambda for 112 is the lambda of the scad penalty).

```

modifyModel

modifyModel

Description

Modify the model from lavaan to fit your needs

Usage

```
modifyModel(  
  addMeans = FALSE,  
  activeSet = NULL,  
  dataSet = NULL,  
  transformations = NULL,  
  transformationList = list(),  
  transformationGradientStepSize = 1e-06  
)
```

Arguments

addMeans	If lavaanModel has meanstructure = FALSE, addMeans = TRUE will add a mean structure. FALSE will set the means of the observed variables to their observed means.
activeSet	Option to only use a subset of the individuals in the data set. Logical vector of length N indicating which subjects should remain in the sample.
dataSet	option to replace the data set in the lavaan model with a different data set. Can be useful for cross-validation
transformations	allows for transformations of parameters - useful for measurement invariance tests etc.
transformationList	optional list used within the transformations. NOTE: This must be used as an Rcpp::List.
transformationGradientStepSize	step size used to compute the gradients of the transformations

Value

Object of class modifyModel

Examples

```
modification <- modifyModel(addMeans = TRUE) # adds intercepts to a lavaan object  
# that was fitted without explicit intercepts
```

newTau	<i>newTau</i>
--------	---------------

Description

assign new value to parameter tau used by approximate optimization. Any regularized value below tau will be evaluated as zeroed which directly impacts the AIC, BIC, etc.

Usage

```
newTau(regularizedSEM, tau)
```

Arguments

```
regularizedSEM  object fitted with approximate optimization
tau             new tau value
```

Value

regularizedSEM, but with new regularizedSEM@fits\$nonZeroParameters

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ l1*y1 + l2*y2 + l3*y3 + l4*y4 + l5*y5 +
      l6*y6 + l7*y7 + l8*y8 + l9*y9 + l10*y10 +
      l11*y11 + l12*y12 + l13*y13 + l14*y14 + l15*y15
f ~~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- smoothLasso(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("l", 6:15),
  epsilon = 1e-10,
  tau = 1e-4,
  lambdas = seq(0,1,length.out = 50))
newTau(regularizedSEM = lsem, tau = .1)
```

plot,cvRegularizedSEM,missing-method
plots the cross-validation fits

Description

plots the cross-validation fits

Usage

```
## S4 method for signature 'cvRegularizedSEM,missing'  
plot(x, y, ...)
```

Arguments

x	object of class cvRegularizedSEM
y	not used
...	not used

Value

either an object of ggplot2 or of plotly

plot,gpRegularized,missing-method
*plots the regularized and unregularized parameters for all levels of
lambda*

Description

plots the regularized and unregularized parameters for all levels of lambda

Usage

```
## S4 method for signature 'gpRegularized,missing'  
plot(x, y, ...)
```

Arguments

x	object of class gpRegularized
y	not used
...	use regularizedOnly=FALSE to plot all parameters

Value

either an object of ggplot2 or of plotly

```
plot,regularizedSEM,missing-method
```

plots the regularized and unregularized parameters for all levels of lambda

Description

plots the regularized and unregularized parameters for all levels of lambda

Usage

```
## S4 method for signature 'regularizedSEM,missing'
plot(x, y, ...)
```

Arguments

x	object of class gpRegularized
y	not used
...	use regularizedOnly = FALSE to plot all parameters and addLabels = FALSE to remove labels to the lines

Value

either an object of ggplot2 or of plotly

```
plot,stabSel,missing-method
```

plots the regularized and unregularized parameters for all levels of the tuning parameters

Description

plots the regularized and unregularized parameters for all levels of the tuning parameters

Usage

```
## S4 method for signature 'stabSel,missing'
plot(x, y, ...)
```

Arguments

x	object of class stabSel
y	not used
...	use regularizedOnly=FALSE to plot all parameters

Value

either an object of `ggplot2` or of `plotly`

regressions

regressions

Description

Extract the labels of all regressions found in a lavaan model.

Usage

```
regressions(lavaanModel)
```

Arguments

lavaanModel fitted lavaan model

Value

vector with parameter labels

Examples

```
# The following is adapted from ?lavaan::sem
library(lessSEM)
model <- '
  # latent variable definitions
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + a*y2 + b*y3 + c*y4
  dem65 =~ y5 + a*y6 + b*y7 + c*y8

  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60

  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

fit <- sem(model, data = PoliticalDemocracy)

regressions(fit)
```

```
regsem2LavaanParameters
      regsem2LavaanParameters
```

Description

helper function: regsem and lavaan use slightly different parameter labels. This function can be used to translate the parameter labels of a cv_regsem object to lavaan labels

Usage

```
regsem2LavaanParameters(regsemModel, lavaanModel)
```

Arguments

regsemModel	model of class regsem
lavaanModel	model of class lavaan

Value

regsem parameters with lavaan labels

Examples

```
## The following is adapted from ?regsem::regsem.
#library(lessSEM)
#library(regsem)
## put variables on same scale for regsem
#HS <- data.frame(scale(HolzingerSwineford1939[,7:15]))
#
#mod <- '
#f =~ 1*x1 + 11*x2 + 12*x3 + 13*x4 + 14*x5 + 15*x6 + 16*x7 + 17*x8 + 18*x9
#'
## Recommended to specify meanstructure in lavaan
#lavaanModel <- cfa(mod, HS, meanstructure=TRUE)
#
#regsemModel <- regsem(lavaanModel,
#                      lambda = 0.3,
#                      gradFun = "ram",
#                      type="lasso",
#                      pars_pen=c("l1", "l2", "l6", "l7", "l8"))
# regsem2LavaanParameters(regsemModel = regsemModel,
#                         lavaanModel = lavaanModel)
```

regularizedSEM-class *Class for regularized SEM*

Description

Class for regularized SEM

Slots

penalty penalty used (e.g., "lasso")
 parameters data.frame with parameter estimates
 fits data.frame with all fit results
 parameterLabels character vector with names of all parameters
 weights vector with weights given to each of the parameters in the penalty
 regularized character vector with names of regularized parameters
 transformations if the model has transformations, the transformed parameters are returned
 internalOptimization list of elements used internally
 inputArguments list with elements passed by the user to the general
 notes internal notes that have come up when fitting the model

regularizedSEMMixedPenalty-class
 Class for regularized SEM

Description

Class for regularized SEM

Slots

penalty penalty used (e.g., "lasso")
 tuningParameterConfigurations list with settings for the lambda, theta, and alpha tuning parameters.
 parameters data.frame with parameter estimates
 fits data.frame with all fit results
 parameterLabels character vector with names of all parameters
 weights vector with weights given to each of the parameters in the penalty
 regularized character vector with names of regularized parameters
 transformations if the model has transformations, the transformed parameters are returned
 internalOptimization list of elements used internally
 inputArguments list with elements passed by the user to the general
 notes internal notes that have come up when fitting the model

ridge	<i>ridge</i>
-------	--------------

Description

Implements ridge regularization for structural equation models. The penalty function is given by:

$$p(x_j) = \lambda x_j^2$$

Note that ridge regularization will not set any of the parameters to zero but result in a shrinkage towards zero.

Usage

```
ridge(  
  lavaanModel,  
  regularized,  
  lambdas,  
  method = "glmnet",  
  modifyModel = lessSEM::modifyModel(),  
  control = lessSEM::controlGlmnet()  
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
method	which optimizer should be used? Currently implemented are <code>ista</code> and <code>glmnet</code> . With <code>ista</code> , the <code>control</code> argument can be used to switch to related procedures (currently <code>gist</code>).
modifyModel	used to modify the <code>lavaanModel</code> . See <code>?modifyModel</code> .
control	used to control the optimizer. This element is generated with the <code>controlIsta</code> and <code>controlGlmnet</code> functions. See <code>?controlIsta</code> and <code>?controlGlmnet</code> for more details.

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

Ridge regularization:

- Hoerl, A. E., & Kennard, R. W. (1970). Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12(1), 55–67. <https://doi.org/10.1080/00401706.1970.10488634>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Model of class regularizedSEM

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
```

```

meanstructure = TRUE,
std.lv = TRUE)

# Regularization:

lsem <- ridge(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("1", 6:15),
  lambdas = seq(0,1,length.out = 20))

# use the plot-function to plot the regularized parameters:
plot(lsem)

# the coefficients can be accessed with:
coef(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters[1,]

#### Advanced ###
# Switching the optimizer #
# Use the "method" argument to switch the optimizer. The control argument
# must also be changed to the corresponding function:
lsemIsta <- ridge(
  lavaanModel = lavaanModel,
  regularized = paste0("1", 6:15),
  lambdas = seq(0,1,length.out = 20),
  method = "ista",
  control = controlIsta())

# Note: The results are basically identical:
lsemIsta@parameters - lsem@parameters

```

ridgeBfgs

ridgeBfgs

Description

This function allows for regularization of models built in lavaan with the ridge penalty. Its elements can be accessed with the "@" operator (see examples).

Usage

```

ridgeBfgs(
  lavaanModel,
  regularized,
  lambdas = NULL,

```

```

    modifyModel = lessSEM::modifyModel(),
    control = lessSEM::controlBFGS()
  )

```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
control	used to control the optimizer. This element is generated with the <code>controlBFGS</code> function. See <code>?controlBFGS</code> for more details.

Details

For more details, see:

1. Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>
2. Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>

Value

Model of class `regularizedSEM`

Examples

```

library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

```

```

# Regularization:

# names of the regularized parameters:
regularized = paste0("1", 6:15)

lsem <- ridgeBfgs(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  regularized = regularized,
  lambdas = seq(0,1,length.out = 50))

plot(lsem)

# the coefficients can be accessed with:
coef(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters[1,]

```

scad

scad

Description

Implements scad regularization for structural equation models. The penalty function is given by:
Equation Omitted in Pdf Documentation.

Usage

```

scad(
  lavaanModel,
  regularized,
  lambdas,
  thetas,
  modifyModel = lessSEM::modifyModel(),
  method = "glmnet",
  control = lessSEM::controlGlmnet()
)

```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda

thetas	parameters whose absolute value is above this threshold will be penalized with a constant (theta)
modifyModel	used to modify the lavaanModel. See ?modifyModel.
method	which optimizer should be used? Currently implemented are ista and glmnet. With ista, the control argument can be used to switch to related procedures (currently gist).
control	used to control the optimizer. This element is generated with the controlIsta (see ?controlIsta)

Details

Identical to **regsem**, models are specified using **lavaan**. Currently, most standard SEM are supported. **lessSEM** also provides full information maximum likelihood for missing data. To use this functionality, fit your **lavaan** model with the argument `sem(..., missing = 'ml')`. **lessSEM** will then automatically switch to full information maximum likelihood as well.

scad regularization:

- 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. <https://doi.org/10.1198/01621450175>

Regularized SEM

- Huang, P.-H., Chen, H., & Weng, L.-J. (2017). A Penalized Likelihood Method for Structural Equation Modeling. *Psychometrika*, 82(2), 329–354. <https://doi.org/10.1007/s11336-017-9566-9>
- Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016>

For more details on GLMNET, see:

- Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1), 1–20. <https://doi.org/10.18637/jss.v033.i01>
- Yuan, G.-X., Chang, K.-W., Hsieh, C.-J., & Lin, C.-J. (2010). A Comparison of Optimization Methods and Software for Large-scale L1-regularized Linear Classification. *Journal of Machine Learning Research*, 11, 3183–3234.
- Yuan, G.-X., Ho, C.-H., & Lin, C.-J. (2012). An improved GLMNET for l1-regularized logistic regression. *The Journal of Machine Learning Research*, 13, 1999–2030. <https://doi.org/10.1145/2020408.2020421>

For more details on ISTA, see:

- Beck, A., & Teboulle, M. (2009). A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences*, 2(1), 183–202. <https://doi.org/10.1137/080716542>
- Gong, P., Zhang, C., Lu, Z., Huang, J., & Ye, J. (2013). A General Iterative Shrinkage and Thresholding Algorithm for Non-convex Regularized Optimization Problems. *Proceedings of the 30th International Conference on Machine Learning*, 28(2)(2), 37–45.
- Parikh, N., & Boyd, S. (2013). Proximal Algorithms. *Foundations and Trends in Optimization*, 1(3), 123–231.

Value

Model of class `regularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ l1*y1 + l2*y2 + l3*y3 + l4*y4 + l5*y5 +
      l6*y6 + l7*y7 + l8*y8 + l9*y9 + l10*y10 +
      l11*y11 + l12*y12 + l13*y13 + l14*y14 + l15*y15
f ~~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- scad(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
  regularized = paste0("l", 6:15),
  lambdas = seq(0,1,length.out = 20),
  thetas = seq(2.01,5,length.out = 5))

# the coefficients can be accessed with:
coef(lsem)

# if you are only interested in the estimates and not the tuning parameters, use
coef(lsem)@estimates
# or
estimates(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters[1,]

# fit Measures:
fitIndices(lsem)

# The best parameters can also be extracted with:
coef(lsem, criterion = "AIC")
# or
```

```
estimates(lsem, criterion = "AIC")

# optional: plotting the paths requires installation of plotly
# plot(lsem)
```

scadPenalty_C	<i>scadPenalty_C</i>
---------------	----------------------

Description

scadPenalty_C

Usage

```
scadPenalty_C(par, lambda_p, theta)
```

Arguments

- par single parameter value
- lambda_p lambda value for this parameter
- theta theta value for this parameter

Value

penalty value

SEMCpp	<i>SEMCpp class</i>
--------	---------------------

Description

internal SEM representation

Fields

- new Creates a new SEMCpp.
- fill fills the SEM with the elements from an Rcpp::List
- addTransformation adds transforamtions to a model
- implied Computes implied means and covariance matrix
- fit Fits the model. Returns objective value of the fitting function
- getParameters Returns a data frame with model parameters.
- getEstimator returns the estimator used in the model (e.g., fiml)
- getParameterLabels Returns a vector with unique parameter labels as used internally.

getGradients Returns a matrix with scores.
 getScores Returns a matrix with scores.
 getHessian Returns the hessian of the model. Expects the labels of the parameters and the values of the parameters as well as a boolean indicating if these are raw. Finally, a double (eps) controls the precision of the approximation.
 computeTransformations compute the transformations.
 setTransformationGradientStepSize change the step size of the gradient computation for the transformations

show, cvRegularizedSEM-method

Show method for objects of class cvRegularizedSEM.

Description

Show method for objects of class cvRegularizedSEM.

Usage

```
## S4 method for signature 'cvRegularizedSEM'
show(object)
```

Arguments

object object of class cvRegularizedSEM

Value

No return value, just prints estimates

show, gpRegularized-method

show

Description

show

Usage

```
## S4 method for signature 'gpRegularized'
show(object)
```

Arguments

object object of class gpRegularized

Value

No return value, just prints estimates

show, lessSEMCoef-method

show

Description

show

Usage

```
## S4 method for signature 'lessSEMCoef'  
show(object)
```

Arguments

object object of class lessSEMCoef

Value

No return value, just prints estimates

show, logLikelihood-method

show

Description

show

Usage

```
## S4 method for signature 'logLikelihood'  
show(object)
```

Arguments

object object of class logLikelihood

Value

No return value, just prints estimates

show,Rcpp_mgSEM-method

show

Description

show

Usage

```
## S4 method for signature 'Rcpp_mgSEM'  
show(object)
```

Arguments

object object of class Rcpp_mgSEM

Value

No return value, just prints estimates

show,Rcpp_SEMCpp-method

show

Description

show

Usage

```
## S4 method for signature 'Rcpp_SEMCpp'  
show(object)
```

Arguments

object object of class Rcpp_SEMCpp

Value

No return value, just prints estimates

show,regularizedSEM-method
show

Description

show

Usage

```
## S4 method for signature 'regularizedSEM'  
show(object)
```

Arguments

object object of class regularizedSEM

Value

No return value, just prints estimates

show,regularizedSEMMixedPenalty-method
show

Description

show

Usage

```
## S4 method for signature 'regularizedSEMMixedPenalty'  
show(object)
```

Arguments

object object of class regularizedSEM

Value

No return value, just prints estimates

show, stabSel-method	<i>show</i>
----------------------	-------------

Description

show

Usage

```
## S4 method for signature 'stabSel'
show(object)
```

Arguments

object object of class stabSel

Value

No return value, just prints estimates

simulateExampleData	<i>simulateExampleData</i>
---------------------	----------------------------

Description

simulate data for a simple CFA model

Usage

```
simulateExampleData(
  N = 100,
  loadings = c(rep(1, 5), rep(0.4, 5), rep(0, 5)),
  percentMissing = 0
)
```

Arguments

N number of persons in the data set
loadings loadings of the latent variable on the manifest observations
percentMissing percentage of missing data

Value

data set for a single-factor CFA.

Examples

```
y <- lessSEM::simulateExampleData()
```

smoothAdaptiveLasso	<i>smoothAdaptiveLasso</i>
---------------------	----------------------------

Description

This function allows for regularization of models built in lavaan with the smooth adaptive lasso penalty. The returned object is an S4 class; its elements can be accessed with the "@" operator (see examples).

Usage

```
smoothAdaptiveLasso(  
  lavaanModel,  
  regularized,  
  weights = NULL,  
  lambdas,  
  epsilon,  
  tau,  
  modifyModel = lessSEM::modifyModel(),  
  control = lessSEM::controlBFGS()  
)
```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
weights	labeled vector with weights for each of the parameters in the model. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object. If set to NULL, the default weights will be used: the inverse of the absolute values of the unregularized parameter estimates
lambdas	numeric vector: values for the tuning parameter lambda
epsilon	epsilon > 0; controls the smoothness of the approximation. Larger values = smoother
tau	parameters below threshold tau will be seen as zeroed
modifyModel	used to modify the lavaanModel. See <code>?modifyModel</code> .
control	used to control the optimizer. This element is generated with the <code>controlBFGS</code> function. See <code>?controlBFGS</code> for more details.

Details

For more details, see:

1. Zou, H. (2006). The Adaptive Lasso and Its Oracle Properties. Journal of the American Statistical Association, 101(476), 1418–1429. <https://doi.org/10.1198/016214506000000735>

2. Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>
3. Lee, S.-I., Lee, H., Abbeel, P., & Ng, A. Y. (2006). Efficient L1 Regularized Logistic Regression. *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, 401–408.

Value

Model of class `regularizedSEM`

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

# names of the regularized parameters:
regularized = paste0("1", 6:15)

# define adaptive lasso weights:
# We use the inverse of the absolute unregularized parameters
# (this is the default in adaptiveLasso and can also specified
# by setting weights = NULL)
weights <- 1/abs(getLavaanParameters(lavaanModel))
weights[!names(weights) %in% regularized] <- 0

lsem <- smoothAdaptiveLasso(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  regularized = regularized,
  weights = weights,
  epsilon = 1e-10,
  tau = 1e-4,
  lambdas = seq(0,1,length.out = 50))
```

```

# use the plot-function to plot the regularized parameters:
plot(lsem)

# the coefficients can be accessed with:
coef(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters[1,]

# AIC and BIC:
AIC(lsem)
BIC(lsem)

# The best parameters can also be extracted with:
coef(lsem, criterion = "AIC")
coef(lsem, criterion = "BIC")

```

smoothElasticNet	<i>smoothElasticNet</i>
------------------	-------------------------

Description

This function allows for regularization of models built in lavaan with the smooth elastic net penalty. Its elements can be accessed with the "@" operator (see examples).

Usage

```

smoothElasticNet(
  lavaanModel,
  regularized,
  lambdas = NULL,
  nLambdas = NULL,
  alphas,
  epsilon,
  tau,
  modifyModel = lessSEM::modifyModel(),
  control = lessSEM::controlBFGS()
)

```

Arguments

lavaanModel	model of class lavaan
regularized	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object
lambdas	numeric vector: values for the tuning parameter lambda

nLambdas	alternative to lambda: If alpha = 1, lessSEM can automatically compute the first lambda value which sets all regularized parameters to zero. It will then generate nLambda values between 0 and the computed lambda.
alphas	numeric vector with values of the tuning parameter alpha. Must be between 0 and 1. 0 = ridge, 1 = lasso.
epsilon	epsilon > 0; controls the smoothness of the approximation. Larger values = smoother
tau	parameters below threshold tau will be seen as zeroed
modifyModel	used to modify the lavaanModel. See ?modifyModel.
control	used to control the optimizer. This element is generated with the controlBFGS function. See ?controlBFGS for more details.

Details

For more details, see:

1. Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B*, 67(2), 301–320. <https://doi.org/10.1111/j.1467-9868.2005.00503.x> for the details of this regularization technique.
2. Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. *Structural Equation Modeling: A Multidisciplinary Journal*, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>
3. Lee, S.-I., Lee, H., Abbeel, P., & Ng, A. Y. (2006). Efficient L1 Regularized Logistic Regression. *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, 401–408.

Value

Model of class regularizedSEM

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f =~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
```

```

                                std.lv = TRUE)

# Regularization:

# names of the regularized parameters:
regularized = paste0("1", 6:15)

lsem <- smoothElasticNet(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  regularized = regularized,
  epsilon = 1e-10,
  tau = 1e-4,
  lambdas = seq(0,1,length.out = 5),
  alphas = seq(0,1,length.out = 3))

# the coefficients can be accessed with:
coef(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters[1,]

```

smoothLasso

smoothLasso

Description

This function allows for regularization of models built in lavaan with the smoothed lasso penalty. The returned object is an S4 class; its elements can be accessed with the "@" operator (see examples). We don't recommend using this function. Use `lasso()` instead.

Usage

```

smoothLasso(
  lavaanModel,
  regularized,
  lambdas,
  epsilon,
  tau,
  modifyModel = lessSEM::modifyModel(),
  control = lessSEM::controlBFGS()
)

```

Arguments

<code>lavaanModel</code>	model of class lavaan
<code>regularized</code>	vector with names of parameters which are to be regularized. If you are unsure what these parameters are called, use <code>getLavaanParameters(model)</code> with your lavaan model object

lambdas	numeric vector: values for the tuning parameter lambda
epsilon	epsilon > 0; controls the smoothness of the approximation. Larger values = smoother
tau	parameters below threshold tau will be seen as zeroed
modifyModel	used to modify the lavaanModel. See ?modifyModel.
control	used to control the optimizer. This element is generated with the controlBFGS function. See ?controlBFGS for more details.

Details

For more details, see:

1. Lee, S.-I., Lee, H., Abbeel, P., & Ng, A. Y. (2006). Efficient L1 Regularized Logistic Regression. Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06), 401–408.
2. Jacobucci, R., Grimm, K. J., & McArdle, J. J. (2016). Regularized Structural Equation Modeling. Structural Equation Modeling: A Multidisciplinary Journal, 23(4), 555–566. <https://doi.org/10.1080/10705511.2016.1191111>

Value

Model of class regularizedSEM

Examples

```
library(lessSEM)

# Identical to regsem, lessSEM builds on the lavaan
# package for model specification. The first step
# therefore is to implement the model in lavaan.

dataset <- simulateExampleData()

lavaanSyntax <- "
f =~ 11*y1 + 12*y2 + 13*y3 + 14*y4 + 15*y5 +
      16*y6 + 17*y7 + 18*y8 + 19*y9 + 110*y10 +
      111*y11 + 112*y12 + 113*y13 + 114*y14 + 115*y15
f ~~ 1*f
"

lavaanModel <- lavaan::sem(lavaanSyntax,
                           data = dataset,
                           meanstructure = TRUE,
                           std.lv = TRUE)

# Regularization:

lsem <- smoothLasso(
  # pass the fitted lavaan model
  lavaanModel = lavaanModel,
  # names of the regularized parameters:
```

```

regularized = paste0("l", 6:15),
epsilon = 1e-10,
tau = 1e-4,
lambdas = seq(0,1,length.out = 50))

# use the plot-function to plot the regularized parameters:
plot(lsem)

# the coefficients can be accessed with:
coef(lsem)

# elements of lsem can be accessed with the @ operator:
lsem@parameters[1,]

# AIC and BIC:
AIC(lsem)
BIC(lsem)

# The best parameters can also be extracted with:
coef(lsem, criterion = "AIC")
coef(lsem, criterion = "BIC")

```

stabilitySelection	<i>stabilitySelection</i>
--------------------	---------------------------

Description

Provides rudimentary stability selection for regularized SEM. Stability selection has been proposed by Meinshausen & Bühlmann (2010) and was extended to SEM by Li & Jacobucci (2021). The problem that stability selection tries to solve is the instability of regularization procedures: Small changes in the data set may result in different parameters being selected. To address this issue, stability selection uses random subsamples from the initial data set and fits models in these subsamples. For each parameter, we can now check how often it is included in the model for a given set of tuning parameters. Plotting these probabilities can provide an overview over which of the parameters are often removed and which remain in the model most of the time. To get a final selection, a threshold t can be defined: If a parameter is in the model $t\%$ of the time, it is retained.

Usage

```

stabilitySelection(
  modelSpecification,
  subsampleSize,
  numberOfSubsamples = 100,
  threshold = 70,
  maxTries = 10 * numberOfSubsamples
)

```



```

stabSel <- stabilitySelection(
  # IMPORTANT: Wrap your call to the penalty function in an rlang::expr-Block:
  modelSpecification =
    rlang::expr(
      lasso(
        # pass the fitted lavaan model
        lavaanModel = lavaanModel,
        # names of the regularized parameters:
        regularized = paste0("1", 6:15),
        # in case of lasso and adaptive lasso, we can specify the number of lambda
        # values to use. lessSEM will automatically find lambda_max and fit
        # models for nLambda values between 0 and lambda_max. For the other
        # penalty functions, lambdas must be specified explicitly
        nLambdas = 50)
      ),
  subsampleSize = 80,
  numberOfSubsamples = 5, # should be set to a much higher number (e.g., 100)
  threshold = 70
)
stabSel
plot(stabSel)

```

stabSel-class

Class for stability selection

Description

Class for stability selection

Slots

regularized names of regularized parameters

tuningParameters data.frame with tuning parameter values

stabilityPaths matrix with percentage of parameters being non-zero averaged over all subsets for each setting of the tuning parameters

percentSelected percentage with which a parameter was selected over all tuning parameter settings

selectedParameters final selected parameters

settings internal

summary, cvRegularizedSEM-method

summary method for objects of class cvRegularizedSEM.

Description

summary method for objects of class cvRegularizedSEM.

Usage

```
## S4 method for signature 'cvRegularizedSEM'
summary(object, ...)
```

Arguments

object	object of class cvRegularizedSEM
...	not used

Value

No return value, just prints estimates

summary, gpRegularized-method

summary

Description

summary

Usage

```
## S4 method for signature 'gpRegularized'
summary(object, ...)
```

Arguments

object	object of class gpRegularized
...	not used

Value

No return value, just prints estimates

summary,regularizedSEM-method
<i>summary</i>

Description

summary

Usage

```
## S4 method for signature 'regularizedSEM'  
summary(object, ...)
```

Arguments

object	object of class regularizedSEM
...	not used

Value

No return value, just prints estimates

summary,regularizedSEMMixedPenalty-method
<i>summary</i>

Description

summary

Usage

```
## S4 method for signature 'regularizedSEMMixedPenalty'  
summary(object, ...)
```

Arguments

object	object of class regularizedSEMMixedPenalty
...	not used

Value

No return value, just prints estimates

variances

variances

Description

Extract the labels of all variances found in a lavaan model.

Usage

```
variances(lavaanModel)
```

Arguments

lavaanModel fitted lavaan model

Value

vector with parameter labels

Examples

```
# The following is adapted from ?lavaan::sem
library(lessSEM)
model <- '
  # latent variable definitions
  ind60 =~ x1 + x2 + x3
  dem60 =~ y1 + a*y2 + b*y3 + c*y4
  dem65 =~ y5 + a*y6 + b*y7 + c*y8

  # regressions
  dem60 ~ ind60
  dem65 ~ ind60 + dem60

  # residual correlations
  y1 ~~ y5
  y2 ~~ y4 + y6
  y3 ~~ y7
  y4 ~~ y8
  y6 ~~ y8
'

fit <- sem(model, data = PoliticalDemocracy)

variances(fit)
```

Index

.adaptBreakingForWls, 5
.checkPenalties, 6
.labelLavaanParameters, 6
.updateLavaan, 7
.useElasticNet, 7

adaptiveLasso, 8
addCappedL1, 11
addElasticNet, 13
addLasso, 15
addLsp, 17
addMcp, 19
addScad, 21
AIC, gpRegularized-method, 23
AIC, Rcpp_mgSEM-method, 24
AIC, Rcpp_SEMCpp-method, 24
AIC, regularizedSEM-method, 25
AIC, regularizedSEMMixedPenalty-method, 25

bfgs, 26
bfgsEnet, 27
bfgsEnetMgSEM, 27
bfgsEnetSEM, 28
BIC, gpRegularized-method, 28
BIC, Rcpp_mgSEM-method, 29
BIC, Rcpp_SEMCpp-method, 29
BIC, regularizedSEM-method, 30
BIC, regularizedSEMMixedPenalty-method, 30

callFitFunction, 31
cappedL1, 31
coef, cvRegularizedSEM-method, 34
coef, gpRegularized-method, 34
coef, Rcpp_mgSEM-method, 35
coef, Rcpp_SEMCpp-method, 35
coef, regularizedSEM-method, 36
coef, regularizedSEMMixedPenalty-method, 36

controlBFGS, 37
controlGlmnet, 39
controlIsta, 40
covariances, 42
createSubsets, 43
curveLambda, 43
cvAdaptiveLasso, 44
cvCappedL1, 47
cvElasticNet, 50
cvLasso, 53
cvLsp, 55
cvMcp, 58
cvRegularizedSEM-class, 61
cvRidge, 62
cvRidgeBfgs, 64
cvScad, 67
cvScaler, 69
cvSmoothAdaptiveLasso, 70
cvSmoothElasticNet, 72
cvSmoothLasso, 75

elasticNet, 77
estimates, 80
estimates, cvRegularizedSEM-method, 81
estimates, regularizedSEM-method, 81
estimates, regularizedSEMMixedPenalty-method, 82

fit, 82
fitIndices, 83
fitIndices, cvRegularizedSEM-method, 84
fitIndices, regularizedSEM-method, 84
fitIndices, regularizedSEMMixedPenalty-method, 85

getLavaanParameters, 85
getTuningParameterConfiguration, 86
glmnetCappedL1MgSEM, 87
glmnetCappedL1SEM, 88
glmnetEnetGeneralPurpose, 88

- glmnetEnetGeneralPurposeCpp, 89
- glmnetEnetMgSEM, 89
- glmnetEnetSEM, 90
- glmnetLspMgSEM, 90
- glmnetLspSEM, 91
- glmnetMcpMgSEM, 91
- glmnetMcpSEM, 92
- glmnetMixedMgSEM, 92
- glmnetMixedPenaltyGeneralPurpose, 93
- glmnetMixedPenaltyGeneralPurposeCpp, 93
- glmnetMixedSEM, 94
- glmnetScadMgSEM, 94
- glmnetScadSEM, 95
- gpAdaptiveLasso, 95
- gpAdaptiveLassoCpp, 98
- gpCappedL1, 102
- gpCappedL1Cpp, 106
- gpElasticNet, 109
- gpElasticNetCpp, 112
- gpLasso, 116
- gpLassoCpp, 119
- gpLsp, 123
- gpLspCpp, 126
- gpMcp, 130
- gpMcpCpp, 132
- gpRegularized-class, 136
- gpRidge, 137
- gpRidgeCpp, 140
- gpScad, 143
- gpScadCpp, 146
- istaCappedL1mgSEM, 150
- istaCappedL1SEM, 150
- istaEnetGeneralPurpose, 151
- istaEnetGeneralPurposeCpp, 151
- istaEnetMgSEM, 152
- istaEnetSEM, 152
- istaLSPMgSEM, 153
- istaLSPSEM, 153
- istaMcpMgSEM, 154
- istaMcpSEM, 154
- istaMixedPenaltyGeneralPurpose, 155
- istaMixedPenaltyGeneralPurposeCpp, 155
- istaMixedPenaltymgSEM, 156
- istaMixedPenaltySEM, 156
- istaScadMgSEM, 157
- istaScadSEM, 157
- lasso, 158
- lavaan2ls1xLabels, 161
- lessSEM2Lavaan, 162
- lessSEMCoef-class, 163
- loadings, 163
- logicalMatch, 164
- logLik,Rcpp_mgSEM-method, 165
- logLik,Rcpp_SEMCpp-method, 165
- logLikelihood-class, 166
- lsp, 166
- makePtrs, 169
- mcp, 170
- mcpPenalty_C, 172
- mgSEM, 173
- mixedPenalty, 174
- modifyModel, 176
- newTau, 177
- plot,cvRegularizedSEM,missing-method, 179
- plot,gpRegularized,missing-method, 179
- plot,regularizedSEM,missing-method, 180
- plot,stabSel,missing-method, 180
- regressions, 181
- regsem2LavaanParameters, 182
- regularizedSEM-class, 183
- regularizedSEMMixedPenalty-class, 183
- ridge, 184
- ridgeBfgs, 186
- scad, 188
- scadPenalty_C, 191
- SEMCpp, 191
- show,cvRegularizedSEM-method, 192
- show,gpRegularized-method, 192
- show,lessSEMCoef-method, 193
- show,logLikelihood-method, 193
- show,Rcpp_mgSEM-method, 194
- show,Rcpp_SEMCpp-method, 194
- show,regularizedSEM-method, 195
- show,regularizedSEMMixedPenalty-method, 195
- show,stabSel-method, 196
- simulateExampleData, 196
- smoothAdaptiveLasso, 197

smoothElasticNet, [199](#)
smoothLasso, [201](#)
stabilitySelection, [203](#)
stabSel-class, [205](#)
summary, cvRegularizedSEM-method, [206](#)
summary, gpRegularized-method, [206](#)
summary, regularizedSEM-method, [207](#)
summary, regularizedSEMMixedPenalty-method,
[207](#)
variances, [208](#)