

# Package ‘gofar’

March 2, 2022

**Type** Package

**Title** Generalized Co-Sparse Factor Regression

**Version** 0.1

**Date** 2022-02-26

**Maintainer** Aditya Mishra <amishra@flatironinstitute.org>

## Description

Divide and conquer approach for estimating low-rank and sparse coefficient matrix in the generalized co-sparse factor regression. Please refer the manuscript 'Mishra, Aditya, Dipak K. Dey, Yong Chen, and Kun Chen. Generalized co-sparse factor regression. Computational Statistics & Data Analysis 157 (2021): 107127' for more details.

**URL** <https://github.com/amishra-stats/gofar>,

<https://www.sciencedirect.com/science/article/pii/S0167947320302188>

**Depends** R (>= 3.5), stats, utils

**Imports** Rcpp (>= 0.12.9), MASS, magrittr, rrpak, glmnet

**License** GPL (>= 3.0)

**LazyData** TRUE

**Encoding** UTF-8

**LinkingTo** Rcpp, RcppArmadillo

**NeedsCompilation** yes

**RoxygenNote** 7.1.2

**Language** en-US

**Author** Aditya Mishra [aut, cre],  
Kun Chen [aut]

**Repository** CRAN

**Date/Publication** 2022-03-02 08:50:10 UTC

## R topics documented:

gofar_control	2
gofar_p	3
gofar_s	5
gofar_sim	8
simulate_gofar	11

<b>Index</b>	<b>12</b>
--------------	-----------

---

gofar_control	<i>Control parameters for the estimation procedure of GOFAR(S) and GOFAR(P)</i>
---------------	---

---

### Description

Default control parameters for Generalized co-sparse factor regression

### Usage

```
gofar_control(
  maxit = 5000,
  epsilon = 1e-06,
  elnetAlpha = 0.95,
  gamma0 = 1,
  se1 = 1,
  spU = 0.5,
  spV = 0.5,
  lamMaxFac = 1,
  lamMinFac = 1e-06,
  initmaxit = 2000,
  initepsilon = 1e-06,
  equalphi = 1,
  objI = 1,
  alp = 60
)
```

### Arguments

maxit	maximum iteration for each sequential steps
epsilon	tolerance value set for convergene of gcure
elnetAlpha	elastic net penalty parameter
gamma0	power parameter in the adaptive weights
se1	apply 1se rule for the model;
spU	maximum proportion of nonzero elements in each column of U
spV	maximum proportion of nonzero elements in each column of V

lamMaxFac	a multiplier of calculated lambda_max
lamMinFac	a multiplier of determining lambda_min as a fraction of lambda_max
initmaxit	maximum iteration for initialization problem
initedpsilon	tolerance value for convergence in the initialization problem
equalphi	dispersion parameter for all gaussian outcome equal or not 0/1
objI	1 or 0 convergence on the basis of objective function or not
alp	scaling factor corresponding to poisson outcomes

**Value**

a list of controlling parameter.

**References**

*Mishra, Aditya, Dipak K. Dey, Yong Chen, and Kun Chen. Generalized co-sparse factor regression. Computational Statistics & Data Analysis 157 (2021): 107127*

**Examples**

```
# control variable for GOFAR(S) and GOFAR(P)
control <- gofar_control()
```

---

gofar_p	<i>Generalize Exclusive factor extraction via co-sparse unit-rank estimation (GOFAR(P)) using k-fold crossvalidation</i>
---------	--

---

**Description**

Divide and conquer approach for low-rank and sparse coefficient matrix estimation: Exclusive extraction

**Usage**

```
gofar_p(
  Yt,
  X,
  nrank = 3,
  nlambda = 40,
  family,
  familygroup = NULL,
  cIndex = NULL,
  ofset = NULL,
  control = list(),
  nfold = 5,
  PATH = FALSE
)
```

**Arguments**

Yt	response matrix
X	covariate matrix; when X = NULL, the function performs unsupervised learning
nrank	an integer specifying the desired rank/number of factors
nlambda	number of lambda values to be used along each path
family	set of family gaussian, bernoulli, poisson
familygroup	index set of the type of multivariate outcomes: "1" for Gaussian, "2" for Bernoulli, "3" for Poisson outcomes
cIndex	control index, specifying index of control variable in the design matrix X
offset	offset matrix specified
control	a list of internal parameters controlling the model fitting
nfold	number of fold for cross-validation
PATH	TRUE/FALSE for generating solution path of sequential estimate after cross-validation step

**Value**

C	estimated coefficient matrix; based on GIC
Z	estimated control variable coefficient matrix
Phi	estimated dispersion parameters
U	estimated U matrix (generalize latent factor weights)
D	estimated singular values
V	estimated V matrix (factor loadings)
lam	selected lambda values based on the chosen information criterion
lampath	sequences of lambda values used in model fitting. In each sequential unit-rank estimation step, a sequence of length nlambda is first generated between (lamMaxlamMaxFac, lamMaxlamMaxFac*lamMinFac) equally spaced on the log scale, in which lamMax is estimated and the other parameters are specified in gofar_control. The model fitting starts from the largest lambda and stops when the maximum proportion of nonzero elements is reached in either u or v, as specified by spU and spV in gofar_control.
IC	values of information criteria
Upath	solution path of U
Dpath	solution path of D
Vpath	solution path of V
ObjDec	boolean type matrix outcome showing if objective function is monotone decreasing or not.
familygroup	specified familygroup of outcome variables.

**References**

Mishra, Aditya, Dipak K. Dey, Yong Chen, and Kun Chen. *Generalized co-sparse factor regression. Computational Statistics & Data Analysis* 157 (2021): 107127

**Examples**

```

family <- list(gaussian(), binomial(), poisson())
control <- gofar_control()
nlam <- 40 # number of tuning parameter
SD <- 123

# Simulated data for testing

data('simulate_gofar')
attach(simulate_gofar)
q <- ncol(Y)
p <- ncol(X)
# Simulate data with 20% missing entries
miss <- 0.20 # Proportion of entries missing
t.ind <- sample.int(n * q, size = miss * n * q)
y <- as.vector(Y)
y[t.ind] <- NA
Ym <- matrix(y, n, q)
naind <- (!is.na(Ym)) + 0 # matrix(1,n,q)
misind <- any(naind == 0) + 0
#
# Model fitting begins:
control$epsilon <- 1e-7
control$spU <- 50 / p
control$spV <- 25 / q
control$maxit <- 1000
# Model fitting: GOFAR(P) (full data)
set.seed(SD)
rank.est <- 5

fit.eea <- gofar_p(Y, X,
  nrank = rank.est, nlambda = nlam,
  family = family, familygroup = familygroup,
  control = control, nfold = 5
)

# Model fitting: GOFAR(P) (missing data)
set.seed(SD)
rank.est <- 5
fit.eea.m <- gofar_p(Ym, X,
  nrank = rank.est, nlambda = nlam,
  family = family, familygroup = familygroup,
  control = control, nfold = 5
)

```

**Description**

Divide and conquer approach for low-rank and sparse coefficient matrix estimation: Sequential

**Usage**

```
gofar_s(
  Yt,
  X,
  nrank = 3,
  nlambda = 40,
  family,
  familygroup = NULL,
  cIndex = NULL,
  ofset = NULL,
  control = list(),
  nfold = 5,
  PATH = FALSE
)
```

**Arguments**

Yt	response matrix
X	covariate matrix; when X = NULL, the function performs unsupervised learning
nrank	an integer specifying the desired rank/number of factors
nlambda	number of lambda values to be used along each path
family	set of family gaussian, bernoulli, poisson
familygroup	index set of the type of multivariate outcomes: "1" for Gaussian, "2" for Bernoulli, "3" for Poisson outcomes
cIndex	control index, specifying index of control variable in the design matrix X
ofset	offset matrix specified
control	a list of internal parameters controlling the model fitting
nfold	number of folds in k-fold crossvalidation
PATH	TRUE/FALSE for generating solution path of sequential estimate after cross-validation step

**Value**

C	estimated coefficient matrix; based on GIC
Z	estimated control variable coefficient matrix
Phi	estimated dispersion parameters
U	estimated U matrix (generalize latent factor weights)
D	estimated singular values
V	estimated V matrix (factor loadings)
lam	selected lambda values based on the chosen information criterion
familygroup	specified familygroup of outcome variables.
fitCV	output from crossvalidation step, for each sequential step

## References

*Mishra, Aditya, Dipak K. Dey, Yong Chen, and Kun Chen. Generalized co-sparse factor regression. Computational Statistics & Data Analysis 157 (2021): 107127*

## Examples

```
family <- list(gaussian(), binomial(), poisson())
control <- gofar_control()
nlam <- 40 # number of tuning parameter
SD <- 123

# Simulated data for testing

data('simulate_gofar')
attach(simulate_gofar)
q <- ncol(Y)
p <- ncol(X)
#
# Simulate data with 20% missing entries
miss <- 0.20 # Proportion of entries missing
t.ind <- sample.int(n * q, size = miss * n * q)
y <- as.vector(Y)
y[t.ind] <- NA
Ym <- matrix(y, n, q)
naind <- (!is.na(Ym)) + 0 # matrix(1,n,q)
misind <- any(naind == 0) + 0
#
# Model fitting begins:
control$epsilon <- 1e-7
control$spU <- 50 / p
control$spV <- 25 / q
control$maxit <- 1000

# Model fitting: GOFAR(S) (full data)
set.seed(SD)
rank.est <- 5
fit.seq <- gofar_s(Y, X,
  nrank = rank.est, family = family,
  nlambdas = nlam, familygroup = familygroup,
  control = control, nfold = 5
)

# Model fitting: GOFAR(S) (missing data)
set.seed(SD)
rank.est <- 5
fit.seq.m <- gofar_s(Ym, X,
  nrank = rank.est, family = family,
  nlambdas = nlam, familygroup = familygroup,
```

```

    control = control, nfold = 5
  )

```

---

gofar\_sim

*Simulate data for GOFAR*


---

### Description

Generates random samples from a generalized sparse factor regression model

### Usage

```
gofar_sim(U, D, V, n, Xsigma, C0, familygroup, snr)
```

### Arguments

U	specified value of U
D	specified value of D
V	specified value of V
n	sample size
Xsigma	covariance matrix for generating sample of X
C0	Specified coefficient matrix with first row being intercept
familygroup	index set of the type of multivariate outcomes: "1" for Gaussian, "2" for Bernoulli, "3" for Poisson outcomes
snr	signal to noise ratio specified for gaussian type outcomes

### Value

Y	Generated response matrix
X	Generated predictor matrix
sigmaG	standard deviation for gaussian error

### References

*Mishra, Aditya, Dipak K. Dey, Yong Chen, and Kun Chen. Generalized co-sparse factor regression. Computational Statistics & Data Analysis 157 (2021): 107127*



## Examples

```

## Model specification:
SD <- 123
set.seed(SD)
n <- 200
p <- 100
pz <- 0
# Model I in the paper
# n <- 200; p <- 300; pz <- 0 ;           # Model II in the paper
# q1 <- 0; q2 <- 30; q3 <- 0             # Similar response cases
q1 <- 15
q2 <- 15
q3 <- 0 # mixed response cases
nrank <- 3 # true rank
rank.est <- 4 # estimated rank
nlam <- 40 # number of tuning parameter
s <- 1 # multiplying factor to singular value
snr <- 0.25 # SNR for variance Gaussian error
#
q <- q1 + q2 + q3
respFamily <- c("gaussian", "binomial", "poisson")
family <- list(gaussian(), binomial(), poisson())
familygroup <- c(rep(1, q1), rep(2, q2), rep(3, q3))
cfamily <- unique(familygroup)
nfamily <- length(cfamily)
#
control <- gofar_control()
#
#
## Generate data
D <- rep(0, nrank)
V <- matrix(0, ncol = nrank, nrow = q)
U <- matrix(0, ncol = nrank, nrow = p)
#
U[, 1] <- c(sample(c(1, -1), 8, replace = TRUE), rep(0, p - 8))
U[, 2] <- c(rep(0, 5), sample(c(1, -1), 9, replace = TRUE), rep(0, p - 14))
U[, 3] <- c(rep(0, 11), sample(c(1, -1), 9, replace = TRUE), rep(0, p - 20))
#
if (nfamily == 1) {
  # for similar type response type setting
  V[, 1] <- c(rep(0, 8), sample(c(1, -1), 8,
    replace =
    TRUE
  ) * runif(8, 0.3, 1), rep(0, q - 16))
  V[, 2] <- c(rep(0, 20), sample(c(1, -1), 8,
    replace =
    TRUE
  ) * runif(8, 0.3, 1), rep(0, q - 28))
  V[, 3] <- c(
    sample(c(1, -1), 5, replace = TRUE) * runif(5, 0.3, 1), rep(0, 23),
    sample(c(1, -1), 2, replace = TRUE) * runif(2, 0.3, 1), rep(0, q - 30)
  )
}

```

```

)
} else {
  # for mixed type response setting
  # V is generated such that joint learning can be emphasised
  V1 <- matrix(0, ncol = nrank, nrow = q / 2)
  V1[, 1] <- c(sample(c(1, -1), 5, replace = TRUE), rep(0, q / 2 - 5))
  V1[, 2] <- c(
    rep(0, 3), V1[4, 1], -1 * V1[5, 1],
    sample(c(1, -1), 3, replace = TRUE), rep(0, q / 2 - 8)
  )
  V1[, 3] <- c(
    V1[1, 1], -1 * V1[2, 1], rep(0, 4),
    V1[7, 2], -1 * V1[8, 2], sample(c(1, -1), 2, replace = TRUE),
    rep(0, q / 2 - 10)
  )
  #
  V2 <- matrix(0, ncol = nrank, nrow = q / 2)
  V2[, 1] <- c(sample(c(1, -1), 5, replace = TRUE), rep(0, q / 2 - 5))
  V2[, 2] <- c(
    rep(0, 3), V2[4, 1], -1 * V2[5, 1],
    sample(c(1, -1), 3, replace = TRUE), rep(0, q / 2 - 8)
  )
  V2[, 3] <- c(
    V2[1, 1], -1 * V2[2, 1], rep(0, 4),
    V2[7, 2], -1 * V2[8, 2],
    sample(c(1, -1), 2, replace = TRUE), rep(0, q / 2 - 10)
  )
  #
  V <- rbind(V1, V2)
}
U[, 1:3] <- apply(U[, 1:3], 2, function(x) x / sqrt(sum(x^2)))
V[, 1:3] <- apply(V[, 1:3], 2, function(x) x / sqrt(sum(x^2)))
#
D <- s * c(4, 6, 5) # signal strength varries as per the value of s
or <- order(D, decreasing = TRUE)
U <- U[, or]
V <- V[, or]
D <- D[or]
C <- U %*% (D * t(V)) # simulated coefficient matrix
intercept <- rep(0.5, q) # specifying intercept to the model:
C0 <- rbind(intercept, C)
#
Xsigma <- 0.5^abs(outer(1:p, 1:p, FUN = "-"))
# Simulated data
sim.sample <- gofar_sim(U, D, V, n, Xsigma, C0, familygroup, snr)
# Dispersion parameter
pHI <- c(rep(sim.sample$sigmaG, q1), rep(1, q2), rep(1, q3))
X <- sim.sample$X[1:n, ]
Y <- sim.sample$Y[1:n, ]
simulate_gofar <- list(Y = Y, X = X, U = U, D = D, V = V, n=n,
  Xsigma = Xsigma, C0 = C0, familygroup = familygroup)

```

---

simulate_gofar	<i>Simulated data for GOFAR</i>
----------------	---------------------------------

---

**Description**

Simulated data with low-rank and sparse coefficient matrix.

**Usage**

```
data(simulate_gofar)
```

**Format**

A list of variables for the analysis using GOFAR(S) and GOFAR(P):

**Y** Generated response matrix

**X** Generated predictor matrix

**U** specified value of U

**V** specified value of V

**D** specified value of D

**n** sample size

**Xsigma** covariance matrix used to generate predictors in X

**C0** intercept value in the coefficient matrix

**familygroup** index set of the type of multivariate outcomes: "1" for Gaussian, "2" for Bernoulli, "3" for Poisson outcomes Mishra, Aditya, Dipak K. Dey, Yong Chen, and Kun Chen. Generalized co-sparse factor regression. Computational Statistics & Data Analysis 157 (2021): 107127

# Index

## \* datasets

simulate\_gofar, 11

gofar\_control, 2

gofar\_p, 3

gofar\_s, 5

gofar\_sim, 8

simulate\_gofar, 11