

# Package: robregcc (via r-universe)

November 6, 2024

**Type** Package

**Title** Robust Regression with Compositional Covariates

**Version** 1.1

**Date** 2020-10-10

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

**Description** We implement the algorithm estimating the parameters of the robust regression model with compositional covariates. The model simultaneously treats outliers and provides reliable parameter estimates. Publication reference: Mishra, A., Mueller, C.,(2019) <[arXiv:1909.04990](https://arxiv.org/abs/1909.04990)>.

**URL** <https://arxiv.org/abs/1909.04990>,  
<https://github.com/amishra-stats/robregcc>

**Depends** R (>= 3.5.0), stats, utils

**License** GPL (>= 3.0)

**LazyData** true

**Imports** Rcpp (>= 0.12.0), MASS, magrittr, graphics

**LinkingTo** Rcpp, RcppArmadillo

**NeedsCompilation** yes

**RoxygenNote** 7.1.1

**Encoding** UTF-8

**Repository** <https://amishra-stats.r-universe.dev>

**RemoteUrl** <https://github.com/amishra-stats/robregcc>

**RemoteRef** HEAD

**RemoteSha** 0ea849d8c93729d0ba491de02835282b99fd68e7

## Contents

classo . . . . .	2
classo_path . . . . .	3

coef_cc . . . . .	4
cpsec_nsp . . . . .	6
cpsec_sp . . . . .	8
getscsfun . . . . .	9
getscsfun.sp . . . . .	10
plot_cv . . . . .	11
plot_path . . . . .	12
plot_resid . . . . .	14
residuals . . . . .	16
robregcc_nsp . . . . .	18
robregcc_option . . . . .	20
robregcc_sim . . . . .	22
robregcc_sim2 . . . . .	24
robregcc_sp . . . . .	26
simulate_robregcc . . . . .	28
simulate_robregcc_nsp . . . . .	29
simulate_robregcc_sp . . . . .	30

<b>Index</b>	<b>31</b>
--------------	-----------

---

classo	<i>Estimate parameters of linear regression model with compositional covariates using method suggested by Pixu shi.</i>
--------	---

---

### Description

The model uses scaled lasoo approach for model selection.

### Usage

```
classo(Xt, y, C, we = NULL, type = 1, control = list())
```

### Arguments

Xt	CLR transformed predictor matrix.
y	model response vector
C	sub-compositional matrix
we	specify weight of model parameter
type	1/2 for l1 / l2 loss in the model
control	a list of internal parameters controlling the model fitting

### Value

beta	model parameter estimate
------	--------------------------

**References**

Shi, P., Zhang, A. and Li, H., 2016. *Regression analysis for microbiome compositional data. The Annals of Applied Statistics*, 10(2), pp.1019-1040.

**Examples**

```
library(robregcc)
library(magrittr)

data(simulate_robregcc)
X <- simulate_robregcc$X;
y <- simulate_robregcc$y
C <- simulate_robregcc$C
n <- nrow(X); p <- ncol(X); k <- nrow(C)

# Predictor transformation due to compositional constraint:
Xt <- cbind(1,X)      # accounting for intercept in predictor
C <- cbind(0,C)       # accounting for intercept in constraint
bw <- c(0,rep(1,p))   # weight matrix to not penalize intercept

# Non-robust regression, [Pixu Shi 2016]
control <- robregcc_option(maxiter = 5000, tol = 1e-7, lminfac = 1e-12)
fit.nr <- classo(Xt, y, C, we = bw, type = 1, control = control)
```

---

classo\_path

---

*Compute solution path of constrained lasso.*


---

**Description**

The model uses scaled lasoo approach for model selection.

**Usage**

```
classo_path(Xt, y, C, we = NULL, control = list())
```

**Arguments**

Xt	CLR transformed predictor matrix.
y	model response vector
C	sub-compositional matrix
we	specify weight of model parameter
control	a list of internal parameters controlling the model fitting

**Value**

betapath	solution path estimate
beta	model parameter estimate

**Examples**

```

library(robregcc)
library(magrittr)

data(simulate_robregcc)
X <- simulate_robregcc$X;
y <- simulate_robregcc$y
C <- simulate_robregcc$C
n <- nrow(X); p <- ncol(X); k <- nrow(C)

#
Xt <- cbind(1,X)           # accounting for intercept in predictor
C <- cbind(0,C)           # accounting for intercept in constraint
bw <- c(0,rep(1,p))       # weight matrix to not penalize intercept

# Non-robust regression
# control <- robregcc_option(maxiter = 5000, tol = 1e-7, lminfac = 1e-12)
# fit.path <- classo_path(Xt, y, C, we = bw, control = control)

```

---

coef_cc	<i>Extract coefficients estimate from the sparse version of the robregcc fitted object.</i>
---------	---

---

**Description**

S3 methods extracting estimated coefficients for objects generated by robregcc. Robust coefficient estimate.

**Usage**

```
coef_cc(object, type = 0, s = 0)
```

**Arguments**

object	Object generated by robregcc.
type	0/1 residual estimate before/after sanity check
s	0/1 no/yes 1se rule

**Value**

coefficient estimate



```

# Robust regression using lasso penalty [Huber equivalent]
fit.soft <- robregcc_sp(Xt,y,C, cindex = 1,
                      control = control, penalty.index = 2,
                      alpha = 0.95)

# Robust regression using hard thresholding penalty
control$lmaxfac = 1e2           # Parameter for constructing sequence of lambda
control$lminfac = 1e-3         # Parameter for constructing sequence of lambda
control$sigmafac = 2#1.345
fit.hard <- robregcc_sp(Xt,y,C, beta.init = fit.init$betaf,
                      gamma.init = fit.init$residuals,
                      cindex = 1,
                      control = control, penalty.index = 3,
                      alpha = 0.95)

coef_cc(fit.ada)
coef_cc(fit.soft)
coef_cc(fit.hard)

```

---

cpssc\_nsp

*Principal sensitivity component analysis with compositional covariates in non-sparse setting.*


---

### Description

Produce model and its residual estimate based in PCS analysis.

### Usage

```
cpssc_nsp(X0, y0, alp = 0.4, cfac = 2, b1 = 0.25, cc1 = 2.937,
          C = NULL, control = list())
```

### Arguments

X0	CLR transformed predictor matrix.
y0	model response vector
alp	(0,0.5) fraction of data sample to be removed to generate subsample
cfac	initial value of shift parameter for weight construction/initialization
b1	tukey bisquare function parameter producing desired breakdown point
cc1	tukey bisquare function parameter producing desired breakdown point
C	sub-compositional matrix
control	a list of internal parameters controlling the model fitting

**Value**

betaf	TModel parameter estimate
residuals	residual estimate

**References**

Mishra, A., Mueller, C.,(2019) *Robust regression with compositional covariates. In prepration.*  
arXiv:1909.04990.

**Examples**

```

library(robregcc)
library(magrittr)

data(simulate_robregcc_nsp)
X <- simulate_robregcc_nsp$X;
y <- simulate_robregcc_nsp$y
C <- simulate_robregcc_nsp$C
n <- nrow(X); p <- ncol(X); k <- nrow(C)

# Predictor transformation due to compositional constraint:
# Equivalent to performing centered log-ratio transform
Xt <- svd(t(C))$u %>% tcrossprod() %>% subtract(diag(p),..) %>% crossprod(t(X),..)
#
Xm <- colMeans(Xt)
Xt <- scale(Xt,Xm,FALSE) # centering of predictors
mean.y <- mean(y)
y <- y - mean.y # centering of response
Xt <- cbind(1,Xt) # accounting for intercept in predictor
C <- cbind(0,C) # accounting for intercept in constraint
bw <- c(0,rep(1,p)) # weight matrix to not penalize intercept

example_seed <- 2*p+1
set.seed(example_seed)

# Breakdown point for tukey Bisquare loss function
b1 = 0.5 # 50% breakdown point
cc1 = 1.567 # corresponding model parameter
# b1 = 0.25; cc1 = 2.937

# Initialization [PSC analysis for compositional data]
control <- robregcc_option(maxiter=3000,tol = 1e-6)
fit.init <- cpsc_nsp(Xt, y,alp=0.4,cfac=2,b1 = b1, cc1 = cc1,C,control)

```

---

cpsc_sp	<i>Principal sensitivity component analysis with compositional covariates in sparse setting.</i>
---------	--

---

### Description

Produce model and its residual estimate based on PCS analysis.

### Usage

```
cpsc_sp(
  X0,
  y0,
  alp = 0.4,
  cfac = 2,
  b1 = 0.25,
  cc1 = 2.937,
  C = NULL,
  we,
  type,
  control = list()
)
```

### Arguments

X0	CLR transformed predictor matrix.
y0	model response vector
alp	(0,0.5) fraction of data sample to be removed to generate subsample
cfac	initial value of shift parameter for weight construction/initialization
b1	tukey bisquare function parameter producing desired breakdown point
cc1	tukey bisquare function parameter producing desired breakdown point
C	sub-compositional matrix
we	penalization index for model parameters beta
type	1/2 for l1 / l2 loss in the model
control	a list of internal parameters controlling the model fitting

### Value

betaf	TModel parameter estimate
residuals	residual estimate

### References

Mishra, A., Mueller, C.,(2019) *Robust regression with compositional covariates. In prepration.* arXiv:1909.04990.

**Examples**

```

library(robregcc)
library(magrittr)

data(simulate_robregcc)
X <- simulate_robregcc$X;
y <- simulate_robregcc$y
C <- simulate_robregcc$C
n <- nrow(X); p <- ncol(X); k <- nrow(C)

Xt <- cbind(1,X) # include intercept in predictor
C <- cbind(0,C) # include intercept in constraint
bw <- c(0,rep(1,p)) # weights not penalize intercept

example_seed <- 2*p+1
set.seed(example_seed)

# Breakdown point for tukey Bisquare loss function
b1 = 0.5 # 50% breakdown point
cc1 = 1.567 # corresponding model parameter
b1 = 0.25; cc1 = 2.937

# Initialization [PSC analysis for compositional data]
control <- robregcc_option(maxiter = 1000,
  tol = 1e-4,lminfac = 1e-7)
# fit.init <- cpsc_sp(Xt, y,alp = 0.4, cfac = 2, b1 = b1, cc1 = cc1,C,bw,1,control)

```

---

getscsfun

---

*Subfunction for principal sensitive component analysis:*


---

**Description**

Subfunction PCS non-sparse.

**Usage**

```
getscsfun(Xa, ya, alp0 = 0.4, b1 = 0.25, cc1 = 2.937)
```

**Arguments**

Xa	CLR transformed predictor matrix.
ya	model response vector
alp0	(0,0.5) fraction of data sample to be removed to generate subsample
b1	tukey bisquare function parameter producing desired breakdown point
cc1	tukey bisquare function parameter producing desired breakdown point

**Value**

beta	Model parameter estimate
scale	scale estimate

**References**

Mishra, A., Mueller, C.,(2018) *Robust regression with compositional covariates. In prepration.*

**Examples**

```
### specify examples here to be shown in the package:
print("aditya")
```

---

getscsfun.sp	<i>Subfunction for principal sensitive component analysis (sparsity):</i>
--------------	---

---

**Description**

Subfunction PCS sparse.

**Usage**

```
getscsfun.sp(Xa, ya, alp0 = 0.4, b1 = 0.25, cc1 = 2.937, C = NULL,
  we, type, control = list())
```

**Arguments**

Xa	CLR transformed predictor matrix.
ya	model response vector
alp0	(0,0.5) fraction of data sample to be removed to generate subsample
b1	tukey bisquare function parameter producing desired breakdown point
cc1	tukey bisquare function parameter producing desired breakdown point
C	sub-compositional matrix
we	penalization index for model parameters beta
type	1/2 for l1 / l2 loss in the model
control	a list of internal parameters controlling the model fitting

**Value**

beta	Model parameter estimate
scale	scale estimate

**References**

Mishra, A., Mueller, C.,(2018) *Robust regression with compositional covariates. In prepration.*

**Examples**

```
### specify examples here to be shown in the package:
print("aditya")
```

---

plot_cv	<i>Plot cross-validation error plot</i>
---------	---

---

**Description**

S3 methods plotting crossvalidation error using the object obtained from robregcc.

**Usage**

```
plot_cv(object)
```

**Arguments**

object                    robregcc fitted onject

**Value**

generate cv error plot

**Examples**

```
library(magrittr)
library(robregcc)

data(simulate_robregcc)
X <- simulate_robregcc$X;
y <- simulate_robregcc$y
C <- simulate_robregcc$C
n <- nrow(X); p <- ncol(X); k <- nrow(C)

Xt <- cbind(1,X)                    # accounting for intercept in predictor
C <- cbind(0,C)                    # accounting for intercept in constraint
bw <- c(0,rep(1,p))                # weight matrix to not penalize intercept

example_seed <- 2*p+1
set.seed(example_seed)

# Breakdown point for tukey Bisquare loss function
b1 = 0.5                            # 50% breakdown point
cc1 = 1.567                        # corresponding model parameter
b1 = 0.25; cc1 = 2.937

# Initialization [PSC analysis for compositional data]
control <- robregcc_option(maxiter=1000,tol = 1e-4,lminfac = 1e-7)
fit.init <- cpsc_sp(Xt, y,alp = 0.4, cfac = 2, b1 = b1,
```

```

cc1 = cc1,C,bw,1,control)

## Robust model fitting

# control parameters
control <- robregcc_option()
beta.wt <- fit.init$betaR # Set weight for model parameter beta
beta.wt[1] <- 0
control$gamma = 1 # gamma for constructing weighted penalty
control$spb = 40/p # fraction of maximum non-zero model parameter beta
control$outMiter = 1000 # Outer loop iteration
control$inMiter = 3000 # Inner loop iteration
control$nlam = 50 # Number of tuning parameter lambda to be explored
control$lmaxfac = 1 # Parameter for constructing sequence of lambda
control$lminfac = 1e-8 # Parameter for constructing sequence of lambda
control$tol = 1e-20; # tolerance parameter for converging [inner loop]
control$out.tol = 1e-16 # tolerance parameter for convergence [outer loop]
control$kfolds = 10 # number of fold of crossvalidation
control$sigmafac = 2#1.345
# Robust regression using adaptive lasso penalty
fit.ada <- robregcc_sp(Xt,y,C,
                      beta.init = beta.wt, cindex = 1,
                      gamma.init = fit.init$residuals,
                      control = control,
                      penalty.index = 1, alpha = 0.95)

# Robust regression using lasso penalty [Huber equivalent]
fit.soft <- robregcc_sp(Xt,y,C, cindex = 1,
                       control = control, penalty.index = 2,
                       alpha = 0.95)

# Robust regression using hard thresholding penalty
control$lmaxfac = 1e2 # Parameter for constructing sequence of lambda
control$lminfac = 1e-3 # Parameter for constructing sequence of lambda
control$sigmafac = 2#1.345
fit.hard <- robregcc_sp(Xt,y,C, beta.init = fit.init$betaf,
                       gamma.init = fit.init$residuals,
                       cindex = 1,
                       control = control, penalty.index = 3,
                       alpha = 0.95)

plot_cv(fit.ada)
plot_cv(fit.soft)
plot_cv(fit.hard)

```

**Description**

S3 methods plotting solution path of model parameter and mean shift using the object obtained from robregcc.

**Usage**

```
plot_path(object, ptype = 0)
```

**Arguments**

object	Object generated by robregcc.
ptype	path type 0/1 for Gamma/Beta path respectively

**Value**

plot solution path

**Examples**

```
library(magrittr)
library(robregcc)

data(simulate_robregcc)
X <- simulate_robregcc$X;
y <- simulate_robregcc$y
C <- simulate_robregcc$C
n <- nrow(X); p <- ncol(X); k <- nrow(C)

Xt <- cbind(1,X)           # accounting for intercept in predictor
C <- cbind(0,C)           # accounting for intercept in constraint
bw <- c(0,rep(1,p))       # weight matrix to not penalize intercept

example_seed <- 2*p+1
set.seed(example_seed)

# Breakdown point for tukey Bisquare loss function
b1 = 0.5                   # 50% breakdown point
cc1 = 1.567                # corresponding model parameter
b1 = 0.25; cc1 = 2.937

# Initialization [PSC analysis for compositional data]
control <- robregcc_option(maxiter=1000,tol = 1e-4,lminfac = 1e-7)
fit.init <- cpsc_sp(Xt, y,alp = 0.4, cfac = 2, b1 = b1,
cc1 = cc1,C,bw,1,control)

## Robust model fitting

# control parameters
control <- robregcc_option()
beta.wt <- fit.init$betaR  # Set weight for model parameter beta
beta.wt[1] <- 0
```

```

control$gamma = 1           # gamma for constructing weighted penalty
control$spb = 40/p         # fraction of maximum non-zero model parameter beta
control$outMiter = 1000    # Outer loop iteration
control$inMiter = 3000    # Inner loop iteration
control$nlam = 50         # Number of tuning parameter lambda to be explored
control$lmaxfac = 1       # Parameter for constructing sequence of lambda
control$lminfac = 1e-8    # Parameter for constructing sequence of lambda
control$tol = 1e-20;      # tolerance parameter for converging [inner loop]
control$out.tol = 1e-16   # tolerance parameter for convergence [outer loop]
control$kfold = 10        # number of fold of crossvalidation
control$sigmafac = 2#1.345
# Robust regression using adaptive lasso penalty
fit.ada <- robregcc_sp(Xt,y,C,
                      beta.init = beta.wt, cindex = 1,
                      gamma.init = fit.init$residuals,
                      control = control,
                      penalty.index = 1, alpha = 0.95)

# Robust regression using lasso penalty [Huber equivalent]
fit.soft <- robregcc_sp(Xt,y,C, cindex = 1,
                       control = control, penalty.index = 2,
                       alpha = 0.95)

# Robust regression using hard thresholding penalty
control$lmaxfac = 1e2     # Parameter for constructing sequence of lambda
control$lminfac = 1e-3    # Parameter for constructing sequence of lambda
control$sigmafac = 2#1.345
fit.hard <- robregcc_sp(Xt,y,C, beta.init = fit.init$betaf,
                       gamma.init = fit.init$residuals,
                       cindex = 1,
                       control = control, penalty.index = 3,
                       alpha = 0.95)

plot_path(fit.ada)
plot_path(fit.soft)
plot_path(fit.hard)

```

---

plot\_resid

*Plot residuals estimate from robregcc object*


---

### Description

S3 methods extracting residuals from the objects generated by robregcc.

### Usage

```
plot_resid(object, type = 0, s = 0)
```

**Arguments**

object            Object generated by robregcc.  
 type            0/1 residual estimate before/after sanity check  
 s                0/1 no/yes lse rule

**Value**

plot estimated residual

**Examples**

```
library(magrittr)
library(robregcc)

data(simulate_robregcc)
X <- simulate_robregcc$X;
y <- simulate_robregcc$y
C <- simulate_robregcc$C
n <- nrow(X); p <- ncol(X); k <- nrow(C)

Xt <- cbind(1,X)                            # accounting for intercept in predictor
C <- cbind(0,C)                            # accounting for intercept in constraint
bw <- c(0,rep(1,p))                        # weight matrix to not penalize intercept

example_seed <- 2*p+1
set.seed(example_seed)

# Breakdown point for tukey Bisquare loss function
b1 = 0.5                                    # 50% breakdown point
cc1 = 1.567                                # corresponding model parameter
b1 = 0.25; cc1 = 2.937

# Initialization [PSC analysis for compositional data]
control <- robregcc_option(maxiter=1000,tol = 1e-4,lminfac = 1e-7)
fit.init <- cpsc_sp(Xt, y,alp = 0.4, cfac = 2, b1 = b1,
cc1 = cc1,C,bw,1,control)

## Robust model fitting

# control parameters
control <- robregcc_option()
beta.wt <- fit.init$betaR    # Set weight for model parameter beta
beta.wt[1] <- 0
control$gamma = 1                         # gamma for constructing weighted penalty
control$spb = 40/p                        # fraction of maximum non-zero model parameter beta
control$outMiter = 1000                   # Outer loop iteration
control$inMiter = 3000                   # Inner loop iteration
control$nlam = 50                         # Number of tuning parameter lambda to be explored
control$lmaxfac = 1                       # Parameter for constructing sequence of lambda
control$lminfac = 1e-8                   # Parameter for constructing sequence of lambda
```

```

control$tol = 1e-20;          # tolerance parameter for converging [inner loop]
control$out.tol = 1e-16      # tolerance parameter for convergence [outer loop]
control$kfolds = 10         # number of fold of crossvalidation
control$sigmafac = 2#1.345
# Robust regression using adaptive lasso penalty
fit.ada <- robregcc_sp(Xt,y,C,
                      beta.init = beta.wt, cindex = 1,
                      gamma.init = fit.init$residuals,
                      control = control,
                      penalty.index = 1, alpha = 0.95)

# Robust regression using lasso penalty [Huber equivalent]
fit.soft <- robregcc_sp(Xt,y,C, cindex = 1,
                       control = control, penalty.index = 2,
                       alpha = 0.95)

# Robust regression using hard thresholding penalty
control$lmaxfac = 1e2        # Parameter for constructing sequence of lambda
control$lminfac = 1e-3      # Parameter for constructing sequence of lambda
control$sigmafac = 2#1.345
fit.hard <- robregcc_sp(Xt,y,C, beta.init = fit.init$betaf,
                       gamma.init = fit.init$residuals,
                       cindex = 1,
                       control = control, penalty.index = 3,
                       alpha = 0.95)

plot_resid(fit.ada)
plot_resid(fit.soft)
plot_resid(fit.hard)

```

---

residuals	<i>Extract residuals estimate from the sparse version of the robregcc fitted object.</i>
-----------	--

---

### Description

Robust residuals estimate

### Usage

```

## S3 method for class 'robregcc'
residuals(object, ...)

```

**Arguments**

object           robregcc fitted object  
 ...             Other arguments for future usage.

**Value**

residuals estimate

**Examples**

```
library(magrittr)
library(robregcc)

data(simulate_robregcc)
X <- simulate_robregcc$X;
y <- simulate_robregcc$y
C <- simulate_robregcc$C
n <- nrow(X); p <- ncol(X); k <- nrow(C)

Xt <- cbind(1,X)                           # accounting for intercept in predictor
C <- cbind(0,C)                            # accounting for intercept in constraint
bw <- c(0,rep(1,p))                        # weight matrix to not penalize intercept

example_seed <- 2*p+1
set.seed(example_seed)

# Breakdown point for tukey Bisquare loss function
b1 = 0.5                                  # 50% breakdown point
cc1 = 1.567                                # corresponding model parameter
b1 = 0.25; cc1 = 2.937

# Initialization [PSC analysis for compositional data]
control <- robregcc_option(maxiter=1000,tol = 1e-4,lminfac = 1e-7)
fit.init <- cpsc_sp(Xt, y,alp = 0.4, cfac = 2, b1 = b1,
cc1 = cc1,C,bw,1,control)

## Robust model fitting

# control parameters
control <- robregcc_option()
beta.wt <- fit.init$betaR   # Set weight for model parameter beta
beta.wt[1] <- 0
control$gamma = 1                         # gamma for constructing weighted penalty
control$spb = 40/p                        # fraction of maximum non-zero model parameter beta
control$outMiter = 1000                   # Outer loop iteration
control$inMiter = 3000                    # Inner loop iteration
control$nlam = 50                         # Number of tuning parameter lambda to be explored
control$lmaxfac = 1                        # Parameter for constructing sequence of lambda
control$lminfac = 1e-8                    # Parameter for constructing sequence of lambda
control$tol = 1e-20;                      # tolerance parameter for converging [inner loop]
```

```

control$out.tol = 1e-16      # tolerance parameter for convergence [outer loop]
control$kfolds = 10        # number of fold of crossvalidation
control$sigmafac = 2#1.345
# Robust regression using adaptive lasso penalty
fit.ada <- robregcc_sp(Xt,y,C,
                      beta.init = beta.wt, cindex = 1,
                      gamma.init = fit.init$residuals,
                      control = control,
                      penalty.index = 1, alpha = 0.95)

# Robust regression using lasso penalty [Huber equivalent]
fit.soft <- robregcc_sp(Xt,y,C, cindex = 1,
                       control = control, penalty.index = 2,
                       alpha = 0.95)

# Robust regression using hard thresholding penalty
control$lmaxfac = 1e2      # Parameter for constructing sequence of lambda
control$lminfac = 1e-3    # Parameter for constructing sequence of lambda
control$sigmafac = 2#1.345
fit.hard <- robregcc_sp(Xt,y,C, beta.init = fit.init$betaf,
                       gamma.init = fit.init$residuals,
                       cindex = 1,
                       control = control, penalty.index = 3,
                       alpha = 0.95)

residuals(fit.ada)
residuals(fit.soft)
residuals(fit.hard)

```

---

robregcc_nsp	<i>Robust model estimation approach for regression with compositional covariates.</i>
--------------	---

---

### Description

Generate solution path for range of lambda in case of where model parameter beta is not assumed to be sparse(nsp).

### Usage

```

robregcc_nsp(X, y, C, intercept = FALSE, gamma.wt = NULL,
             control = list(), penalty.index = 1, verbose = TRUE)

```

**Arguments**

X	CLR transformed predictor matrix.
y	model response vector
C	sub-compositional constraint matrix
intercept	true/false to include intercept term in the model
gamma.wt	initial value of shift parameter for weight construction/initialization
control	a list of internal parameters controlling the model fitting
penalty.index	1, 2, 3 corresponding to adaptive, soft and hard penalty
verbose	TRUE/FALSE for showing progress of the cross validation

**Value**

Method	Type of penalty used
betapath	model parameter estimate along solution path
gammapath	shift parameter estimate along solution path
lambpath	sequence of fitted lambda)
X	predictors
y	response

**References**

Mishra, A., Mueller, C.,(2019) *Robust regression with compositional covariates. In prepration.* arXiv:1909.04990.

**Examples**

```
library(robregcc)
library(magrittr)

data(simulate_robregcc_nsp)
X <- simulate_robregcc_nsp$X;
y <- simulate_robregcc_nsp$y
C <- simulate_robregcc_nsp$C
n <- nrow(X); p <- ncol(X); k <- nrow(C)

# Predictor transformation due to compositional constraint:
# Equivalent to performing centered log-ratio transform
Xt <- svd(t(C))$u %>% tcrossprod() %>% subtract(diag(p),.) %>% crossprod(t(X),.)
#
Xm <- colMeans(Xt)
Xt <- scale(Xt,Xm,FALSE) # centering of predictors
mean.y <- mean(y)
y <- y - mean.y # centering of response
Xt <- cbind(1,Xt) # accounting for intercept in predictor
C <- cbind(0,C) # accounting for intercept in constraint
bw <- c(0,rep(1,p)) # weight matrix to not penalize intercept
```

```

example_seed <- 2*p+1
set.seed(example_seed)

# Breakdown point for tukey Bisquare loss function
b1 = 0.5 # 50% breakdown point
cc1 = 1.567 # corresponding model parameter
# b1 = 0.25; cc1 = 2.937

# Initialization [PSC analysis for compositional data]
control <- robregcc_option(maxiter=3000,tol = 1e-6)
fit.init <- cpsc_nsp(Xt, y,alp=0.4,cfac=2,b1 = b1, cc1 = cc1,C,control)

# Robust procedure
# control parameters
control <- robregcc_option()
control$tol <- 1e-30
control$nlam = 25;
control$lminfac = 1e-5;
control$outMiter = 10000
control$gamma <- 2
# Robust regression using adaptive elastic net penalty [case III, Table 1]
fit.ada <- robregcc_nsp(Xt,y, C, intercept = FALSE,
                      gamma.wt = fit.init$residuals,
                      control = control, penalty.index = 1)

# Robust regression using elastic net penalty [case II, Table 1]
control$lminfac = 1e-1;
fit.soft <- robregcc_nsp(Xt,y,C,intercept = FALSE, gamma.wt = NULL,
                      control = control, penalty.index = 2)

# Robust regression using hard-ridge penalty [case I, Table 1]
control$tol <- 1e-30
control$nlam = 25;
control$lminfac = 1e-1;
control$outMiter = 10000
fit.hard <- robregcc_nsp(Xt,y,C, intercept = FALSE,
                      gamma.wt = fit.init$residuals,
                      control = control, penalty.index = 3)

```

---

robregcc\_option

*Control parameter for model estimation:*


---

### Description

The model approach use scaled lasoo approach for model selection.

**Usage**

```
robregcc_option(
  maxiter = 10000,
  tol = 1e-10,
  nlam = 100,
  out.tol = 1e-08,
  lminfac = 1e-08,
  lmaxfac = 10,
  mu = 1,
  nu = 1.05,
  sp = 0.3,
  gamma = 2,
  outMiter = 3000,
  inMiter = 500,
  kmaxS = 500,
  tolS = 1e-04,
  nlamx = 20,
  nlamy = 20,
  spb = 0.3,
  spy = 0.3,
  lminfacX = 1e-06,
  lminfacY = 0.01,
  kfold = 10,
  fullpath = 0,
  sigmafac = 2
)
```

**Arguments**

maxiter	maximum number of iteration for convergence
tol	tolerance value set for convergence
nlam	number of lambda to be generated to obtain solution path
out.tol	tolerance value set for convergence of outer loop
lminfac	a multiplier of determining lambda_min as a fraction of lambda_max
lmaxfac	a multiplier of lambda_max
mu	penalty parameter used in enforcing orthogonality
nu	penalty parameter used in enforcing orthogonality (incremental rate of mu)
sp	maximum proportion of nonzero elements in shift parameter
gamma	adaptive penalty weight exponential factor
outMiter	maximum number of outer loop iteration
inMiter	maximum number of inner loop iteration
kmaxS	maximum number of iteration for fast S estimator for convergence
tolS	tolerance value set for convergence in case of fast S estimator
nlamx	number of x lambda

n $\lambda$ my	number of y lambda
spb	sparsity in beta
spy	sparsity in shift gamma
lminfacX	a multiplier of determining lambda_min as a fraction of lambda_max for sparsity in X
lminfacY	a multiplier of determining lambda_min as a fraction of lambda_max for sparsity in shift parameter
kfold	number of folds for crossvalidation
fullpath	1/0 to get full path yes/no
sigmafac	multiplying factor for the range of standard deviation

**Value**

a list of controlling parameter.

**Examples**

```
# default options
library(robregcc)
control_default = robregcc_option()
# manual options
control_manual <- robregcc_option(maxiter=1000,tol = 1e-4,lminfac = 1e-7)
```

---

robregcc_sim	<i>Simulation data</i>
--------------	------------------------

---

**Description**

Simulate data for the robust regression with compositional covariates

**Usage**

```
robregcc_sim(n, betacc, beta0, 0, Sigma, levgl, snr, shft, m, C, out = list())
```

**Arguments**

n	sample size
betacc	model parameter satisfying compositional covariates
beta0	intercept
0	number of outlier
Sigma	covariance matrix of simulated predictors
levgl	1/0 whether to include leveraged observation or not
snr	noise to signal ratio
shft	multiplying factor to model variance for creating outlier
m	test sample size
C	subcompositional matrix
out	list for obtaining output with simulated data structure

**Value**

a list containing simulated output.

**References**

Mishra, A., Mueller, C.,(2019) *Robust regression with compositional covariates. In prepration.* arXiv:1909.04990.

**Examples**

```
## Simulation example:
library(robregcc)
library(magrittr)

## n: sample size
## p: number of predictors
## o: fraction of observations as outliers
## L: {0,1} => leveraged {no, yes},
## s: multiplicative factor
## ngrp: number of subgroup in the model
## snr: noise to signal ratio for computing true std_err

## Define parameters to simulate example
p <- 80          # number of predictors
n <- 300         # number of sample
O <- 0.10*n     # number of outlier
L <- 1
s <- 8
ngrp <- 4       # number of sub-composition
snr <- 3        # Signal to noise ratio
example_seed <- 2*p+1 # example seed
set.seed(example_seed)
# Simulate subcomposition matrix
C1 <- matrix(0,ngrp,23)
tind <- c(0,10,16,20,23)
for (ii in 1:ngrp)
  C1[ii,(tind[ii] + 1):tind[ii + 1]] <- 1
C <- matrix(0,ngrp,p)
C[,1:ncol(C1)] <- C1
# model parameter beta
beta0 <- 0.5
beta <- c(1, -0.8, 0.4, 0, 0, -0.6, 0, 0, 0, 0, -1.5, 0, 1.2, 0, 0, 0.3)
beta <- c(beta,rep(0,p - length(beta)))
# Simulate response and predictor, i.e., X, y
Sigma <- 1:p %>% outer(.,.,'-') %>% abs(); Sigma <- 0.5^Sigma
data.case <- vector("list",1)
set.seed(example_seed)
data.case <- robregcc_sim(n,beta,beta0, O = 0,
  Sigma,levg = L, snr,shft = s,0, C,out = data.case)
data.case$C <- C
# We have saved a copy of simulated data in the package
# with name simulate_robregcc
```

```
# simulate_robregcc = data.case;
# save(simulate_robregcc, file = 'data/simulate_robregcc.rda')

X <- data.case$X          # predictor matrix
y <- data.case$y         # model response
```

---

robregcc\_sim2

*Simulation data with mis-specified model parameters*


---

### Description

Simulate data for the robust regression with compositional covariates with mis-specified model parameters

### Usage

```
robregcc_sim2(
  n,
  betacc,
  beta0,
  betacm,
  O,
  Sigma,
  levg,
  snr,
  m,
  C,
  out = list()
)
```

### Arguments

n	sample size
betacc	model parameter satisfying compositional covariates
beta0	intercept
betacm	model parameter satisfying compositional covariates mis-specified
O	number of outlier
Sigma	covariance matrix of simulated predictors
levg	1/0 whether to include leveraged observation or not
snr	noise to signal ratio
m	test sample size
C	subcompositional matrix
out	list for obtaining output with simulated data structure

**Value**

a list containing simulated output.

**References**

Mishra, A., Mueller, C.,(2019) *Robust regression with compositional covariates. In prepration.* arXiv:1909.04990.

**Examples**

```
## Simulation example:
library(robregcc)
library(magrittr)

## n: sample size
## p: number of predictors
## o: fraction of observations as outliers
## L: {0,1} => leveraged {no, yes},
## s: multiplicative factor
## ngrp: number of subgroup in the model
## snr: noise to signal ratio for computing true std_err

## Define parameters to simulate example
p <- 80          # number of predictors
n <- 300         # number of sample
O <- 0.10*n     # number of outlier
L <- 1
s <- 8
ngrp <- 4       # number of sub-composition
snr <- 3        # Signal to noise ratio
example_seed <- 2*p+1 # example seed
set.seed(example_seed)
# Simulate subcomposition matrix
C1 <- matrix(0,ngrp,23)
tind <- c(0,10,16,20,23)
for (ii in 1:ngrp)
  C1[ii,(tind[ii] + 1):tind[ii + 1]] <- 1
C <- matrix(0,ngrp,p)
C[,1:ncol(C1)] <- C1
# model parameter beta
beta0 <- 0.5
beta <- c(1, -0.8, 0.4, 0, 0, -0.6, 0, 0, 0, 0, -1.5, 0, 1.2, 0, 0, 0.3)
beta <- c(beta,rep(0,p - length(beta)))
# mis-specified model parameter
betam <- c(-1.8, -0.0, 0.0, 0, 0.9, 0.9, 0, 0, 0, 0, -1.5, 0, 1.2, 0, 0, 0.3)
betam <- c(betam,rep(0,p - length(betam)))
# Simulate response and predictor, i.e., X, y
Sigma <- 1:p %>% outer(.,.-') %>% abs(); Sigma <- 0.5^Sigma
data.case <- vector("list",1)
set.seed(example_seed)
data.case <- robregcc_sim2(n,beta,beta0,betam, O = 0,Sigma,levg = L,
snr,0, C,out = data.case)
```

```

data.case$C <- C
# We have saved a copy of simulated data in the package
# with name simulate_robregcc
# simulate_robregcc = data.case;
# save(simulate_robregcc, file = 'data/simulate_robregcc.rda')

X <- data.case$X          # predictor matrix
y <- data.case$y          # model response

```

---

robregcc_sp	<i>Robust model estimation approach for regression with compositional covariates.</i>
-------------	---

---

### Description

Fit regression model with compositional covariates for a range of tuning parameter lambda. Model parameters is assumed to be sparse.

### Usage

```

robregcc_sp(
  X,
  y,
  C,
  beta.init = NULL,
  gamma.init = NULL,
  cindex = 1,
  control = list(),
  penalty.index = 3,
  alpha = 1,
  verbose = TRUE
)

```

### Arguments

X	predictor matrix
y	phenotype/response vector
C	conformable sub-compositional matrix
beta.init	initial value of model parameter beta
gamma.init	inital value of shift parameter gamma
cindex	index of control (not penalized) variable in the model
control	a list of internal parameters controlling the model fitting
penalty.index	a vector of length 2 specifying type of penalty for model parameter and shift parameter respectively. 1, 2, 3 corresponding to adaptive, soft and hard penalty
alpha	elastic net penalty
verbose	TRUE/FALSE for showing progress of the cross validation

**Value**

Method	Type of penalty used
betapath	model parameter estimate along solution path
gammapath	shift parameter estimate along solution path
lamapath	sequence of fitted lambda)
k0	scaling factor
cver	error from k fold cross validation
selInd	selected index from minimum and 1se rule cross validation error
beta0	beta estimate corresponding to selected index
gamma0	mean shift estimate corresponding to selected index
residual0	residual estimate corresponding to selected index
inlier0	inlier index corresponding to selected index
betaE	Post selection estimate corresponding to selected index
residualE	post selection residual corresponding to selected index
inlierE	post selection inlier index corresponding to selected index

**References**

Mishra, A., Mueller, C.,(2019) *Robust regression with compositional covariates. In prepration.* arXiv:1909.04990.

**Examples**

```

library(magrittr)
library(robregcc)

data(simulate_robregcc)
X <- simulate_robregcc$X;
y <- simulate_robregcc$y
C <- simulate_robregcc$C
n <- nrow(X); p <- ncol(X); k <- nrow(C)

Xt <- cbind(1,X)           # accounting for intercept in predictor
C <- cbind(0,C)           # accounting for intercept in constraint
bw <- c(0,rep(1,p))      # weight matrix to not penalize intercept

example_seed <- 2*p+1
set.seed(example_seed)

# Breakdown point for tukey Bisquare loss function
b1 = 0.5                  # 50% breakdown point
cc1 = 1.567               # corresponding model parameter
b1 = 0.25; cc1 = 2.937

# Initialization [PSC analysis for compositional data]
control <- robregcc_option(maxiter=1000,tol = 1e-4,lminfac = 1e-7)

```

```

fit.init <- cpsc_sp(Xt, y, alp = 0.4, cfac = 2, b1 = b1,
cc1 = cc1, C, bw, 1, control)

## Robust model fitting

# control parameters
control <- robregcc_option()
beta.wt <- fit.init$betaR # Set weight for model parameter beta
beta.wt[1] <- 0
control$gamma = 1 # gamma for constructing weighted penalty
control$spb = 40/p # fraction of maximum non-zero model parameter beta
control$outMiter = 1000 # Outer loop iteration
control$inMiter = 3000 # Inner loop iteration
control$nlam = 50 # Number of tuning parameter lambda to be explored
control$lmaxfac = 1 # Parameter for constructing sequence of lambda
control$lminfac = 1e-8 # Parameter for constructing sequence of lambda
control$tol = 1e-20; # tolerance parameter for converging [inner loop]
control$out.tol = 1e-16 # tolerance parameter for convergence [outer loop]
control$kfold = 10 # number of fold of crossvalidation
control$sigmafac = 2#1.345
# Robust regression using adaptive lasso penalty
fit.ada <- robregcc_sp(Xt,y,C,
beta.init = beta.wt, cindex = 1,
gamma.init = fit.init$residuals,
control = control,
penalty.index = 1, alpha = 0.95)

# Robust regression using lasso penalty [Huber equivalent]
fit.soft <- robregcc_sp(Xt,y,C, cindex = 1,
control = control, penalty.index = 2,
alpha = 0.95)

# Robust regression using hard thresholding penalty
control$lmaxfac = 1e2 # Parameter for constructing sequence of lambda
control$lminfac = 1e-3 # Parameter for constructing sequence of lambda
control$sigmafac = 2#1.345
fit.hard <- robregcc_sp(Xt,y,C, beta.init = fit.init$betaf,
gamma.init = fit.init$residuals,
cindex = 1,
control = control, penalty.index = 3,
alpha = 0.95)

```

---

simulate\_robregcc

*Simulated data for testing functions in the robregcc package (sparse setting).*

---

### Description

A list of response (y), predictors (X) and sub-cpmposition matrix (C).

**Usage**

```
data(simulate_robregcc)
```

**Format**

A list with three components:

**X** Compositional predictors.

**y** Outcome with outliers.

**C** Sub-composition matrix.

**Details**

Vector **y**, response with a certain percentage of observations as outliers.

Matrix **X**, Compositional predictors.

**Source**

Simulated data

**Examples**

```
library(robregcc)
data(simulate_robregcc)
X <- simulate_robregcc$X;
y <- simulate_robregcc$y
C <- simulate_robregcc$C
n <- nrow(X); p <- ncol(X); k <- nrow(C)
```

---

`simulate_robregcc_nsp` *Simulated data for testing functions in the robregcc package (non-sparse setting).*

---

**Description**

A list of response (**y**), predictors (**X**) and sub-composition matrix (**C**).

**Usage**

```
data(simulate_robregcc_nsp)
```

**Format**

A list with three components:

**X** Compositional predictors.

**y** Outcome with outliers.

**C** Sub-composition matrix.

**Details**

Vector  $y$ , response with a certain percentage of observations as outliers.

Matrix  $X$ , Compositional predictors.

**Examples**

```
data(simulate_robregcc_nsp)
X <- simulate_robregcc_nsp$X;
y <- simulate_robregcc_nsp$y
C <- simulate_robregcc_nsp$C
n <- nrow(X); p <- ncol(X); k <- nrow(C)
```

---

simulate\_robregcc\_sp *Simulated data for testing functions in the robregcc package (sparse setting).*

---

**Description**

A list of response ( $y$ ), predictors ( $X$ ) and sub-composition matrix ( $C$ ).

**Usage**

```
data(simulate_robregcc_sp)
```

**Format**

A list with three components:

**X** Compositional predictors.

**y** Outcome with outliers.

**C** Sub-composition matrix.

**Details**

Vector  $y$ , response with a certain percentage of observations as outliers.

Matrix  $X$ , Compositional predictors.

**Examples**

```
data(simulate_robregcc_sp)
X <- simulate_robregcc_sp$X;
y <- simulate_robregcc_sp$y
C <- simulate_robregcc_sp$C
n <- nrow(X); p <- ncol(X); k <- nrow(C)
```

# Index

## \* datasets

- simulate\_robregcc, 28
- simulate\_robregcc\_nsp, 29
- simulate\_robregcc\_sp, 30

- classo, 2
- classo\_path, 3
- coef\_cc, 4
- cpssc\_nsp, 6
- cpssc\_sp, 8

- getscsfun, 9
- getscsfun.sp, 10

- plot\_cv, 11
- plot\_path, 12
- plot\_resid, 14

- residuals, 16
- robregcc\_nsp, 18
- robregcc\_option, 20
- robregcc\_sim, 22
- robregcc\_sim2, 24
- robregcc\_sp, 26

- simulate\_robregcc, 28
- simulate\_robregcc\_nsp, 29
- simulate\_robregcc\_sp, 30