

# Package ‘rqcanon’

May 9, 2026

**Type** Package

**Title** Canonical Quantile Regression

**Version** 0.1.0

**Description** A quantile regression method for multivariate data to find linear combinations of explanatory and response variables generalizing canonical correlation. The package consists of functions, `rqcan()` for fitting the coefficients, and `summary.rqcan()`, which calls a bootstrap function. For details, see the help files for `rqcan()` and `summary.rqcan()`, and the reference: Portnoy (2022) <[doi:10.1016/j.jmva.2022.105071](https://doi.org/10.1016/j.jmva.2022.105071)>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Imports** quantreg

**RoxygenNote** 7.3.1

**Depends** R (>= 2.10)

**NeedsCompilation** no

**Author** Stephen Portnoy [aut, cre],  
Isa Mostachetti [com],  
Daniel Taylor-Rodriguez [rev]

**Maintainer** Stephen Portnoy <[sportnoy@illinois.edu](mailto:sportnoy@illinois.edu)>

**Repository** CRAN

**Date/Publication** 2024-07-27 16:20:02 UTC

## Contents

<code>boot.can</code> . . . . .	2
<code>example_data</code> . . . . .	3
<code>rqcan</code> . . . . .	4
<code>rqcan1</code> . . . . .	5
<code>summary.rqcan</code> . . . . .	6

<b>Index</b>	<b>8</b>
--------------	----------

boot.can

*Bootstrap***Description**

Internal function to carry out the bootstrap ; It is a sub-module of summary.rqcan; not intended for general use.

The parameters may be passed by summary.rqcan (see below)

**Usage**

```
boot.can(
  a,
  Rep = 200,
  method = "Andrews",
  msub = 0.9,
  seed,
  nsing = 5,
  prb = FALSE
)
```

**Arguments**

a	output from rqcan
Rep	number of bootstrap replications (default=200)
method	"Andrews" (default) or ""xy"
msub	parameter defining the size of the bootstrap subsample for developmental work only: see the boot.can function
seed	a starting seed (default: missing: no new seed set)
nsing	number of consecutive singular replicatioions to ignore (default = 5)
prb	if TRUE (default = FALSE), print every time 10 percent of the bootstrap samples are done

**Details**

See help(summary.rqcan) ; If errors occur or modification is wanted, see the routine boot.can

**Value**

Returns list(As, Bs, sdc): As (Bs) are N by dim(alpha) (dim(beta)) arrays of all bootstrap alphas and beta values; sdc = sqrt(m/n): SD adjustment for m-choose-n bootstrap, or 1 for "xy" bootstrap

---

example_data	<i>Psychological data</i>
--------------	---------------------------

---

**Description**

A dataset from UCLA Statistical Methods and Data Analytics about investigating the associations between psychological measures and academic achievement measures.

**Usage**

```
example_data
```

**Format**

```
## 'example_data' A data frame with 600 rows and 8 columns:
```

**locus\_of\_control** Psychological Locus of Control

**self\_concept** Psychological Self Concept

**motivation** Psychological Motivation

**read** Academic Reading

**write** Academic Writing

**math** Academic Math

**science** Academic Science

**female** Binary flag for 1 being female.

**Source**

```
<https://stats.idre.ucla.edu/stat/data/mmreg.csv>
```

---

 rqcan

*Canonical Quantile Regression.*


---

### Description

Given multivariate data matrices  $X$  (explanatory variables) and  $Y$  (response variables), the function fits coefficients of the  $Y$ -variables that are best fit by a quantile regression on  $X$ . These are analogous to the coefficients given by a classical canonical correlations analysis, but replace the implicit L2 norm by an L1 norm. See: "Method" and "Reference" below.

This is a simple S3 class for display formatting purposes.

### Usage

```
rqcan(
  X,
  Y,
  tau = 0.5,
  a.pos = 1,
  ap = rep(1, na),
  na = ncol(Y),
  wts = rep(1, nrow(X))
)
```

### Arguments

<code>X</code>	input design matrix of explanatory variables (without intercept)
<code>Y</code>	input matrix of response variables ( $nrow(Y) = nrow(X)$ )
<code>tau</code>	desired quantile, default = .5
<code>a.pos</code>	for first component: non-empty vector of indices of $Y$ -variables ( $Y$ -columns) whose alpha coefficients are constrained to be positive (to provide the direction of increasing responses); default = 1
<code>ap</code>	for subsequent components $j = 2, 3, \dots$ , <code>na</code> : vector whose $(j-1)$ -th element is the $Y$ -variable (column) index whose alpha coefficients are constrained to be positive; default = <code>rep(1,na-1)</code>
<code>na</code>	number of components desired ( $1 \leq na \leq ncol(Y)$ )
<code>wts</code>	used only for use with the bootstrap methods. If weighting is desired for the sample observations, <code>rqcan</code> will multiply the columns of $X$ and $Y$ by the vector <code>wts</code> ; but the bootstrap methods will apply to the unweighted data, and so will be incorrect; default = <code>rep(1,nrow(X))</code> (unweighted analysis)

### Details

Finds orthogonal alpha coefficients and corresponding best-fitting beta coefficients to minimize  $\sum |x_i' \beta - y_i' \alpha|$  subject to  $\sum |\alpha| = 1$  (where  $x_i$  and  $y_i$  are the  $i$ -th rows of  $X$  and  $Y$ ). The intercept is included ( $X$  should not include intercept). Need  $ncol(Y) > 1$ . For first

component: if  $\text{length}(a.\text{pos}) < \text{ncol}(Y)$ ,  $\text{sum}(\text{lalpha}) = 1$  is constrained by going through all sign choices ( $s_j = \text{sign}(\alpha_j)$ ) and setting  $Y1_j = s_j Y_j$  ( $j$  not in  $a.\text{pos}$ ). A constrained regression quantile fit is applied from `quantreg: rq.fit.fnc(cbind(1,X,Y1),y0=0,R,r,tau)`. where  $(R,r)$  constrains all  $\alpha_j \geq 0$  and  $\text{sum}(\alpha_j) \geq 1$  ( $\text{sum} = 1$  at min). Note: `rq.fit.fnc` solves by generating a sequence of quadratic approximations. The matrix defining one quadratic problem may be singular (and stop the computation) even if the input design matrices are of full rank. If a singularity stop occurs, jittering the data (see `jitter()`) sometimes helps. For the subsequent  $j$ -th component, only the index given by `ap(j-1)` is constrained to be positive. Alpha coefficients for subsequent components are constrained to be orthogonal to previous alpha coefficients.

### Value

object of class "rqcan"; a list of matrices of the alpha and beta coefficients: the  $j$ -th row of each matrix is the coefficients for the  $j$ -th component; input data and the constraint matrices  $R$  and  $r$  are also returned in the list

### Fields

`list` A list.

### References

S. Portnoy, 2022. Canonical quantile regression, *J. Multivar. Anal.*, 192, 105071.

### See Also

See [summary.rqcan](#) for a description of the summary function.

### Examples

```
X <- as.matrix(example_data[,1:3])
Y <- as.matrix(example_data[,4:7])

a <- rqcan(X,Y,tau=.75,a.pos=2)
summary(a)
```

---

rqcan1

*First Component*

---

### Description

Internal function to find the first component

It is not intended for general use, but the documentation may be helpful if errors occur or if one wishes to modify the algorithms

### Usage

```
rqcan1(X, Y, tau = 0.5, a.pos = 1, wts = rep(1, nrow(X)))
```

**Arguments**

X	input X-matrix
Y	input Y-matrix
tau	probability for quantile (default = .5)
a.pos	indices of Y-variable whose coefficient is constrained to be positive (default = 1)
wts	case weights (default = rep(1,nrow(X)) )

**Details**

The function finds the leading pair of indices. Notes: an intercept is added (X should not include 1st col = 1) ; ncol(Y) should be > 1 ; length(a.pos) should be at least 1 to specify coef signs (if tau = .5 and a.pos = NULL, coef and -coef give the same solution) ; for length(a.pos) < ncol(Y), the constraint  $\sum(\alpha_j) = 1$  is set by setting  $Y1_j = s_j Y_j$  (j !in a.pos) where  $s_j = \text{sgn}(\alpha_j)$  ; all sign choices are used and then constrained `rq.fit.fnc( cbind(1,X,Y1),y0=0,R,r,tau)` is applied (R,r) constrains all  $\alpha_j \geq 0$  and  $\sum(\alpha_j) \geq 1$  (makes sum = 1)

**Value**

Returns list(a,X,Y,a.pos,R,r,rho1): a = output from `rq.fit.fnc(XY,y0,R,r,tau)` ; X,Y,a.pos = input data ; R,r = constraint matrices for `rq.fit.fnc` ; rho1 = `rq` objective fct. ; if `rq.fit.fnc` generates a singular matrix, returns "sing"

---

summary.rqcan

---

*Summary of rqcan function results.*


---

**Description**

Uses one of two bootstrap methods to provide Standard Error and confidence intervals for the alpha and beta coefficients for all components.

**Usage**

```
## S3 method for class 'rqcan'
summary(object, pr = TRUE, ci = 1, fact = 1, ...)
```

**Arguments**

object	rqcan object returned by rqcan
pr	print tables if TRUE (default)
ci	type of 95 ci=1: use (adjusted) .025 and .975 percentiles of bootstrap distribution ci=2: use normal approx with adjusted SE based on interquartile range ci may be a vector to provide more than one type of interval; default=1
fact	a factor to adjust conf ints for components 2:na; used only for development
...	parameters that are sent to the bootstrap function: Rep: number of bootstrap replications (default=200) method: "Andrews" (default) or ""xy" msub: parameter defining the size of the bootstrap subsample for developmental work only: see the boot.can function seed: a starting seed (default: missing: no new seed set) nsing: number of consecutive singular replicatios to ignore (default = 5) prb: if TRUE (default = FALSE), print every time 10 percent of the bootstrap samples are done

**Details**

The Portnoy reference showed that a subsample bootstrap (as described by Andrews) gives consistent estimates of SE's and confidence intervals. The subsample size is  $m = \text{ceiling}(\min(n, \max(\log(n)*(px+py+1), n^{m_{\text{sub}}})) )$  (where  $n = \text{nrow}(X)$ ,  $px = \text{ncol}(X)$ ,  $py = \text{ncol}(Y)$ ),  $m_{\text{sub}}$  is as above). Some simulations and examples suggest that this is OK. The usual "xy" bootstrap (sampling rows independently with replacement) can be specified. It seems to give similar confidence intervals to "Andrews", but the SE estimates may be wrong; and no form of consistency has been proven. Note: as noted in `help(rqcan)`, the `quantreg` function `rq.fit.fnc` may generate singular matrices even if the input design matrix is of full rank. In simulation examples, this can happen for some bootstrap replications (perhaps less than 1/1000 times). When this occurs, a new bootstrap replication is drawn. If more than `nsing` consecutive singularities are produced, the bootstrap function returns with those replications that it has already found (a number less than `Rep`), with a warning. If a singularity warning occurs, using "xy", or changing the seed or "jittering" the data (see `jitter()`) sometimes helps.

**Value**

Returns `list(As,Bs,sdc)`: `As` and `Bs` are matrices with `Rep` rows giving alpha beta coefficients for each bootstrap replication; and `sdc` is a standard error adjustment based on the subsample bootstrap:  $\text{sdc} = \sqrt{1 - m/n}$ .

# Index

## \* datasets

example\_data, 3

boot.can, 2

example\_data, 3

rqcan, 4

rqcan1, 5

summary.rqcan, 5, 6