Package 'cctest'

August 27, 2024

Version 1.0.0

Title Canonical Correlations and Tests of Independence

Description A simple interface for multivariate correlation analysis that unifies various classical statistical procedures including t-tests, tests in univariate and multivariate linear models, parametric and nonparametric tests for correlation, Kruskal-Wallis tests, standard non-exact versions of Wilcoxon rank-sum and signed rank tests, chi-squared tests of independence, score tests of particular hypotheses in generalized linear models, canonical correlation analysis and linear discriminant analysis.

Author Robert Schlicht [aut, cre]

Maintainer Robert Schlicht <robert.schlicht@tu-dresden.de>

License EUPL (>= 1.1)

Imports stats

NeedsCompilation no

Repository CRAN

Date/Publication 2024-08-27 11:40:06 UTC

Contents

`cctest` *Tests of Independence Based on Canonical Correlations*

Description

`cctest` estimates canonical correlations between two sets of variables, possibly after removing effects of a third set of variables, and performs a classical multivariate test of (conditional) independence based on Pillai's statistic.

Usage

```
cctest(formula, data = NULL, df = formula[-2L], ..., tol = 1e-07)
```

Arguments

`formula`	A `formula` object of the form `Y ~ X ~ A`, where `Y` represents dependent variables, `X` represents a second set of dependent variables or explanatory variables not present under the null hypothesis, and `A` represents explanatory variables that remain under the null hypothesis. The operators (like `+`) and expansion rules defined for the `model` part of a `formula` object here apply to all three parts alike. Typically, `A` includes at least the constant `1` to specify a model with intercepts; unlike `lm`, the function never adds this automatically.
`data`	An optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model.
`df`	An optional `formula` object of the form `~ A0`, where `A0` is a replacement of `A` for the degrees of freedom computation. If not specified, this is the same as `A`.
`...`	Additional optional arguments passed to `model.frame`. In particular, `subset` specifies which rows of data to include, `na.action` how to handle missing values (e.g., `na.exclude`), and `weights` is a vector of any nonnegative numbers that specify how many identical observations each row represents.
`tol`	The tolerance in the QR decomposition for detecting linear dependencies (i.e., collinearities) of the variables.

Details

`cctest` unifies various classical statistical procedures that involve the same underlying computations, including t-tests, tests in univariate and multivariate linear models, parametric and nonparametric tests for correlation, Kruskal–Wallis tests, standard non-exact versions of Wilcoxon rank-sum and signed rank tests, chi-squared tests of independence, score tests of particular hypotheses in generalized linear models, canonical correlation analysis and linear discriminant analysis (see Examples).

Specifically, for the matrices with ranks r_x and r_y obtained from X and Y by subtracting from each column its orthogonal projection on the column space of A, the function computes factorizations $\tilde{X}U$ and $\tilde{Y}V$ with $\tilde{X}$ and $\tilde{Y}$ having r_x and r_y columns, respectively, such that both $\tilde{X}^\top\tilde{X} = rI$ and $\tilde{Y}^\top\tilde{Y} = rI$, and $\tilde{X}^\top\tilde{Y} = rD$ is a rectangular diagonal matrix with decreasing diagonal elements. The scaling factor r, which should be nonzero, is the dimension of the orthogonal complement of the column space of A_0.

The function realizes this variant of the singular value decomposition (cf. Greenacre 1984, p. 344) by first computing preliminary QR factorizations of the stated form (taking $r = 1$) without the requirement on D, and then, in a second step, modifying these based on a standard singular value decomposition of that matrix. The main work is done in a rotated coordinate system where the column space of A aligns with the coordinate axes. The basic approach and the rank detection algorithm are inspired by the implementations in `cancor` and in `lm`, respectively.

The diagonal elements of D, or singular values, are the estimated *canonical correlations* (Hotelling 1936) of the variables represented by X and Y if these follow a linear model $(X \;\; Y) = A(\alpha \;\; \beta) + (\delta \;\; \epsilon)$ with known A, unknown $(\alpha \;\; \beta)$ and error terms $(\delta \;\; \epsilon)$ that have uncorrelated rows with

expectation zero and an identical unknown covariance matrix. In the most common case, where `A` is given as a constant 1, these are the sample canonical correlations (i.e., based on simple centering) most often presented in the literature for full column ranks r_x and r_y. They are always decreasing and between 0 and 1.

In the case of the linear model with independent normally distributed rows and $A_0 = A$, the ranks r_x and r_y equal, with probability 1, the ranks of the covariance matrices of the rows of X and Y, respectively, or r, whichever is smaller. Under the hypothesis of independence of X and Y, given those ranks, the joint distribution of the s squared singular values, where s is the smaller of the two ranks, is then known and in the case $r \geq r_x + r_y$ has a probability density (Hsu 1939, Anderson 2003, Anderson 2007) given by

$$\rho(t_1, ..., t_s) \propto \prod_{i=1}^{s} t_i^{(|r_x - r_y| - 1)/2} (1 - t_i)^{(r - r_x - r_y - 1)/2} \prod_{i<j} (t_i - t_j),$$

$1 \geq t_1 \geq \cdots \geq t_s \geq 0$. For $s = 1$ this reduces to the well-known case of a single beta distributed R^2 or equivalently an F distributed $\frac{R^2/(r_x+r_y-1)}{(1-R^2)/(r-r_x-r_y+1)}$, with the divisors in the numerator and denominator representing the degrees of freedom, or twice the parameters of the beta distribution.

Pillai's statistic is the sum of squares of the canonical correlations, which equals, even without the requirement on D, the squared Frobenius norm of that matrix (or trace of $D^\top D$). Replacing the distribution of that statistic divided by s (i.e., of the mean of squares) with beta or gamma distributions with first or shape parameter $r_x r_y/2$ and expectation $r_x r_y/(rs)$ leads to the F and chi-squared approximations that the p-values returned by `cctest` are based on.

The F or beta approximation (Pillai 1954, p. 99, p. 44) is usually used with $A_0 = A$ and then is exact if $s = 1$. The chi-squared approximation represents Rao's (1948) score test (with a test statistic that is r times Pillai's statistic) in the model obtained after removing (or conditioning on) the orthogonal projections on the column space of A_0 provided that is a subset of the column space of A.

Value

A list with class `htest` containing the following components:

`x`, `y`	matrices $\tilde{X}$ and $\tilde{Y}$ of new transformed variables
`xinv`, `yinv`	matrices U and V representing the inverse coordinate transformations
`estimate`	vector of canonical correlations, i.e., the diagonal elements of D
`statistic`	vector of p-values based on Pillai's statistic and classical chi-squared and F approximations
`method`	the name of the function
`data.name`	a character string representation of `formula`

Note

The handling of `weights` differs from that in `lm` unless the nonzero weights are scaled so as to have a mean of 1. Also, to facilitate predictions for rows with zero weights (see Examples and the code marked as optional), the square roots of the weights, used internally for scaling the data, are always computed as nonzero numbers, even for zero weights, where they are so small that their square is still numerically zero and hence without effect on the correlation analysis. An `offset`, if included in `A` or `...`, is subtracted from all columns in `X` and `Y`.

Author(s)

Robert Schlicht

References

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika* 28, 321–377. doi:10.1093/biomet/28.34.321, doi:10.2307/2333955

Hsu, P.L. (1939). On the distribution of roots of certain determinantal equations. *Annals of Eugenics* 9, 250–258. doi:10.1111/j.14691809.1939.tb02212.x

Rao, C.R. (1948). Large sample tests of statistical hypotheses concerning several parameters with applications to problems of estimation. *Mathematical Proceedings of the Cambridge Philosophical Society* 44, 50–57. doi:10.1017/S0305004100023987

Pillai, K.C.S. (1954). *On some distribution problems in multivariate analysis* (Institute of Statistics mimeo series 88). North Carolina State University, Dept. of Statistics.

Greenacre, M.J. (1984). *Theory and application of correspondence analysis*. Academic Press.

Anderson, T.W. (2003). *An introduction to multivariate statistical analysis*, 3rd edition, Ch. 12–13. Wiley.

Anderson, T.W. (2007). Multiple discoveries: distribution of roots of determinantal equations. *Journal of Statistical Planning and Inference* 137, 3240–3248. doi:10.1016/j.jspi.2007.03.008

See Also

Functions `cancor`, `anova.mlm` in package `stats` and implementations of canonical correlation analysis in other packages such as `MVar`, `candisc` (both including tests based on Wilks' statistic), `yacca`, `CCA`.

Examples

```
## Artificial observations in 5-by-5 meter quadrats in a forest for
## comparing cctest analyses with equivalent "stats" methods:
set.seed(0)
dat <- within(data.frame(row.names=1:150), {
  plot <- sample(factor(c("a","b")), 150, TRUE)         # plot a or b
  x    <- as.integer(runif(150,1,31) + 81*(plot=="b"))  # x position on grid
  y    <- as.integer(runif(150,1,31) + 61*(plot=="b"))  # y position on grid
  ori  <- sample(factor(c("E","N","S","W")), 150, TRUE) # orientation of slope
  elev <- runif(150,605,645) + 5*(plot=="b")            # elevation (in meters)
  h    <- rnorm(150, 125-.17*elev, 3.5)                 # tree height (in meters)
  h5   <- rnorm(150, h, 2)                              # tree height 5 years earlier
  h10  <- rnorm(150, h5, 2)                             # tree height 10 years earlier
  c15  <- as.integer(rnorm(150, h10, 2) > 20)           # 0-1 coded, 15 years earlier
  sapl <- rnbinom(150, 2.6, mu=.02*elev)                # number of saplings
})
dat[1:8,]

## t-tests:
cctest(h~plot~1, dat)
  t.test(h~plot, dat, var.equal=TRUE)
```

```
  summary(lm(h~plot, dat))
cctest(I(h-20)~1~0, dat)
  t.test(dat$h, mu=20)
  t.test(h~1, dat, mu=20)
cctest(I(h-h5)~1~0, dat)
  t.test(dat$h, dat$h5, paired=TRUE)
  t.test(Pair(h,h5)~1, dat)

## Test for correlation:
cctest(h~elev~1, dat)
  cor.test(~h+elev, dat)

## One-way analysis of variance:
cctest(h~ori~1, dat)
  anova(lm(h~ori, dat))

## F-tests in linear models:
cctest(h~ori~1+elev, dat)
  anova(lm(h~1+elev, dat), lm(h~ori+elev, dat))
cctest(h~h10~0, dat, subset=1:5)
  anova(lm(h~0,dat,subset=1:5), lm(h~0+h10,dat,subset=1:5))

## Test in multivariate linear model based on Pillai's statistic:
cctest(h+h5+h10~x+y~1+elev, dat)
  anova(lm(cbind(h,h5,h10)~elev, dat),
    lm(cbind(h,h5,h10)~elev+x+y, dat))

## Test based on Spearman's rank correlation coefficient:
cctest(rank(h)~rank(elev)~1, dat)
  cor.test(~h+elev, dat, method="spearman", exact=FALSE)

## Kruskal-Wallis and Wilcoxon rank-sum tests:
cctest(rank(h)~ori~1, dat)
  kruskal.test(h~ori, dat)
cctest(rank(h)~plot~1, dat)
  wilcox.test(h~plot, dat, exact=FALSE, correct=FALSE)

## Wilcoxon signed rank test:
cctest(rank(abs(h-h5))~sign(h-h5)~0, subset(dat, h-h5 != 0))
  wilcox.test(h-h5 ~ 1, dat, exact=FALSE, correct=FALSE)

## Chi-squared test of independence:
cctest(ori~plot~1, dat, ~0)
cctest(ori~plot~1, xtabs(~ori+plot,dat), ~0, weights=Freq)
  summary(xtabs(~ori+plot, dat, drop.unused.levels=TRUE))
  chisq.test(dat$ori, dat$plot, correct=FALSE)

## Score test in logistic regression (logit model, ...~1 only):
cctest(c15~x+y~1, dat, ~0)
  anova(glm(c15~1, binomial, dat, epsilon=1e-12),
    glm(c15~1+x+y, binomial, dat), test="Rao")

## Score test in multinomial logit model (...~1 only):
```

```
cctest(ori~x+y~1, dat, ~0)
  with(list(d=dat, e=expand.grid(stringsAsFactors=FALSE,
    i=row.names(dat), j=levels(dat$ori))
  ), anova(
    glm(d[i,"ori"]==j ~ j+d[i,"x"]+d[i,"y"], poisson, e, epsilon=1e-12),
    glm(d[i,"ori"]==j ~ j*(d[i,"x"]+d[i,"y"]), poisson, e), test="Rao"
  ))

## Absolute values of (partial) correlation coefficients:
cctest(h~elev~1, dat)$est
  cor(dat$h, dat$elev)
cctest(h~elev~1+x+y, dat)$est
  cov2cor(estVar(lm(cbind(h,elev)~1+x+y, dat)))
cctest(h~x+y+elev~1, dat)$est^2
  summary(lm(h~1+x+y+elev, dat))$r.squared

## Canonical correlations:
cctest(h+h5+h10~x+y~1, dat)$est
  cancor(dat[c("x","y")],dat[c("h","h5","h10")])$cor

## Linear discriminant analysis:
with(list(
  cc = cctest(h+h5+h10~ori~1, dat, ~ori)
), cc$y / sqrt(1-cc$est^2)[col(cc$y)])[1:7,]
  #predict(MASS::lda(ori~h+h5+h10,dat))$x[1:7,]

## Correspondence analysis:
cctest(ori~plot~1, xtabs(~ori+plot,dat), ~0, weights=Freq)[1:2]
  #MASS::corresp(~plot+ori, dat, nf=2)

## Prediction in multivariate linear model:
with(list(
  cc = cctest(h+h5+h10~1+x+y~0, dat, weights=plot=="a")
), cc$x %*% diag(cc$est,ncol(cc$x),ncol(cc$y)) %*% cc$yinv)[1:7,]
  predict(lm(cbind(h,h5,h10)~1+x+y, dat, subset=plot=="a"), dat)[1:7,]

## Not run:
## Handling of additional arguments and edge cases:
cctest(h~h10~offset(h5), dat)
  anova(lm(h~0+offset(h5), dat), lm(h~0+I(h10-h5)+offset(h5), dat))
cctest(h~x~1, dat, weights=sapl/mean(sapl[sapl!=0]))
  anova(lm(h~1, dat, weights=sapl),
    lm(h~1+x, dat, weights=sapl))
cctest(sqrt(h-17)~elev~1, dat[1:5,], na.action=na.exclude)[1:2]
  scale(resid(lm(cbind(elev,sqrt(h-17))~1, dat[1:5,],
    na.action=na.exclude)), FALSE)
cctest(ori:I(sum(Freq)/Freq)~I(0*Freq)~offset(Freq^0), xtabs(~ori,dat),
    weights=Freq^2/sum(Freq)/c(.4,.1,.2,.3), na.action=na.fail)
  chisq.test(xtabs(~ori,dat), p=c(.4,.1,.2,.3))
cctest(c15~h~1, dat,     tol=0.999*sqrt(1-cctest(h~1~0,dat)$est^2))
  summary(lm(c15~h, dat, tol=0.999*sqrt(1-cctest(h~1~0,dat)$est^2)))
cctest(c15~h~1, dat,     tol=1.001*sqrt(1-cctest(h~1~0,dat)$est^2))
  summary(lm(c15~h, dat, tol=1.001*sqrt(1-cctest(h~1~0,dat)$est^2)))
```

```
cctest(c(1)~c(0)~c(0))
  anova(lm(1~0),lm(1~0))
cctest(0~0~0, dat, na.action=na.fail)
  NaN
cctest(1~0~1, dat)
  anova(lm(h^0~1, dat), lm(h^0~0+1, dat))
cctest(1~1~0, dat)
  anova(lm(h^0~0, dat), lm(h^0~1, dat))
## End(Not run)
```

Index