Package 'kanova'

May 8, 2026

Version 0.3-20

Date 2025-08-19

Title Quasi Analysis of Variance for K-Functions

Author Rolf Turner [aut, cre] (ORCID: <https://orcid.org/0000-0001-5521-5218>)

Maintainer Rolf Turner <rolfturner@posteo.net>

Description One-way and two-way analysis of variance for replicated point patterns, grouped by one or two classification factors, on the basis of the corresponding K-functions.

Imports spatstat.geom, spatstat.explore, spatstat.random

Suggests R.rsp, Devore7

VignetteBuilder R.rsp

LazyData true

Depends R (>= 3.2.2)

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2025-08-19 08:10:02 UTC

Contents

kanova *Quasi analysis of variance for K-functions*

Description

One-way and two-way quasi analysis of variance for replicated point patterns, grouped by one or two classification factors. The analysis is based on the values of a summary function. This summary function may be specified by the user. It is usually one of the four standard summary functions, most often the K-function, which is the default.

Usage

```
kanova(fmla,data,expo=0,rsteps=128,r=NULL,sumFnNm=NULL,
       warnSFN=TRUE,test=TRUE,bylevel=FALSE,
       permtype=c("stdres","data"),nperm=99,brief=TRUE,verb=TRUE,
       keepdata=FALSE,divByVar=TRUE)
```

Arguments

`fmla` A formula specifying the test to be conducted. (See **Details**.) There can be at most two main effect predictors (possibly with interaction between them). The left hand side of `fmla` may be omitted. If so, it is taken to be the name of the first column of `data`.

`data` A hyperframe (see `hyperframe()`) containing the data to be analysed. If the left hand side of `fmla` is supplied, then `data` must have a column with a name which matches this left hand side. If no such column exists, an error is thrown. The response column may be a list of point patterns, or it may be a list of numeric vectors (all of which must have the same length).

These numeric vectors are notionally values of a diagnostic or summary function, applied to point patterns, and then evaluated at the attribute `"r"` (see below) of `data`. However that the entries in the list need not actually *be* summary functions. They are simply numeric vectors and *not* objects of class `"fv"` as returned by a summary function. In particular they may be scalars, which allows the `kanova()` function to be applied to analysis of variance of scalars.

Argument `data` may also have one or two columns with names matching those of the main effect predictors as specified by `fmla`. If such columns cannot be found, then objects with these names are sought in the "parent frame" (see `parent.frame()`.) If the appropriate objects still cannot be found, then an error is thrown. These columns (or objects) are internally coerced to factors.

If the response consists of numeric vectors, and if these vectors are *not* scalars, then `data` must have an attribute `"r"` which specifies the "argument" of the putative summary functions in the list. This attribute should be a numeric vector and must be of the same length as the numeric vectors in the list. If the response functions are scalars, then no `"r"` attribute is needed, and, if present, is ignored.

Finally, `data` may have a columns named `"wts"` ("weights") whose entries are positive scalars. Notionally they may be thought of as a power of the numbers of

points in the corresponding point patterns. If there is no column named "wts" then the weights are all taken to equal 1.

If the response column consists of a list of point patterns then any column named "wts" is ignored, and the weights are indeed a power of the numbers of points in the corresponding point patterns. The impact of using these weights (when expo, see below, is strictly positive) is to diminish the influence of quantities corresponding to patterns having few points, and conversely to emphasise the influence of quantities corresponding to patterns having many points.

expo — Non-negative numeric scalar. Ignored unless the response is a list of point patterns. Statistics in kanova are in general calculated using *weighted* means and in this context the weights are the counts of points in the patterns, raised to the power expo. Unweighteb means are used when expo is equal to 0, the default. If expo is set equal to 1 then the weights are simply the pattern counts.

rsteps — Integer scalar. Ignored if argument r (see below) is supplied (i.e. is not NULL), or if the response consists of a list of numeric vectors (rather than a list of point patterns). The argument rsteps specifies the number of (equal) steps between values of the vector r at which the summary function is evaluated. The values of r are equispaced on the interval from 0 to rtop, the latter being calculated internally. The value of rtop depends on the observation windows of the patterns in the response and on their intensities. It also depends on the summary function being used.

r — Numeric vector. Ignored unless the response consists of a list of point patterns. Note that if the response consists of a list of vectors, representing notional diagnostic functions, then argument data must have an attribute r to provide the relevant r object. In this setting an error is thrown if the required attribute is not present (unless the response vectors are in fact scalars).

The argument r specifies the values (distances) at which the summary/diagnostic function (see below) is evaluated. Errors may be thrown if the first entry of r is not 0, or if the values of r are not sufficiently finely spaced. Generally users should not specify r. unless they have a sound understanding of what they are doing.

sumFnNm — Character string naming the summary/diagnostic function to be used. If this is not one of the "standard four", i.e. "Kest", "Fest", "Gest", or "Jest",and if warnSFN (see below) is TRUE, then a warning is issued. In this case **there may be problems**; the code is not robust in this respect. If sumFnNm is not specified (left NULL) then it defaults to "Kest". Users should apply summary functions other than Kest() only if they have a sound understanding of what they are doing. See **Notes**.

warnSFN — Logical scalar. Should a warning be issued if sumFnNm is not one of the "standard four"?

test — Logical scalar. Should a Monte Carlo test of the null hypothesis be carried out?

bylevel — Logical scalar. Should a test of the model y ~ A + B be carried out as b one-variable tests of the significance of A, within each level of B? That is, should we perform b tests specified by y ~ A, data=sdata[[i]], i = 1, ..., b where b is the number of levels of the factor B, and sdata[[i]] consists of those data corresponding to the i-th level of factor B?

If bylevel is TRUE, then the object returned by kanova() is an object of class "multi.kanova", which is a list of length b, each entry of which is an object of class "kanova".

permtype — Character string specifying what sort of permutations should be done to produce the Monte Carlo test statistics. Ignored if test is FALSE.

If permtype is "stdres" then the Monte Carlo data are formed by permuting the residuals, from the *saturated* model, which have been standardised by dividing them by their (estimated) standard deviations. After permutation, the residuals are "unstandardised" by multiplying them by the appropriate standard deviation. The results are then added back to the fitted values from the null model.

If permtype is "data" then the Monte Carlo data are formed by permuting the original data sets.

In the two-way setting, when the test is for the main effect A, then the data (if permtype is "data") are permuted *within* the levels of B.

If fmla specifies an interaction between the main effect predictors, then permtype cannot be "data" and *must* be "stdres", otherwise an error is thrown.

nperm — The number of permutations to be used to determine the Monte Carlo p-value. Ignored if test is FALSE

brief — Logical scalar. Should the object returned by this function be "brief"? See **Value**.

verb — I.e. "verbose". Logical scalar. Should rudimentary "progress reports" be printed out (in the course of conducting the permutation test for "significance" of the test statistic)? Such "reports" consists simply of indications of how many permutations have been effected so far. Ignored if test is FALSE.

keepdata — Logical scalar. Should a copy of the data, to which the model has been fitted, be included as a component of of the object returned by kanova()? See **Value** for more detail.

divByVar — Logical scalar. Should the components of the test statistic be divided by the variances of the underlying un-squared values? (See **Details**.)

Details

formulae:

The formulae used in the fmla argument may take the form y ~ A, y ~ A + B, or y ~ A * B. These "look like" those used in "ordinary" analysis of variance, but in the second instance the interpretation is different. The formula y ~ A + B does not actually fit the additive A + B model. It effects a test of the "significance" of A, "allowing for B" (see below). It is thereby obvious that y ~ A + B is not equivalent to y ~ B + A (whereas in "ordinary" analysis of variance) they *are* equivalent).

The formula y ~ A * B tests the model with interaction against the additive model, as in ordinary analysis of variance. The additive model is not often meaningful in the current context, so such a test for interaction is probably not meaningful either. The test is included in the code basically for the sake of completeness, and to allow for the possibility that a user may encounter a circumstance in which the additive model actually is meaningful.

allowing for a second effect:

This concept is pertinent only when the formula in question is of the form y ~ A + B. In this

setting, if permtype is "data", the factor B is "allowed for" by permuting the data *within* the levels of B. If permtype is "stdres", the standardised residuals are permuted. These are residuals from the saturated model, which includes a B effect, thereby "allowing for" B.

integration:

The value of the test statistic is obtained as a sum of numerical integrals of certain sums of squares. The integrals are computed via a rough trapezoid rule. The integration is carried out over the value of r, the argument of the summary functions that are being analysed. If the response consists of numeric vectors of length 1, i.e. of scalars, then no integration is in fact performed, and the corresponding (downweighted) sum of squares is returned. You may, if you like, think of this as integrating with respect to a measure which has a point mass of 1 at a conceptual single value of r.

the divByVar **argument:**

The divByVar argument exists essentially to allow the package developers to conduct certain simulation experiments. In normal circumstances the user would not set the value of this argument (i.e. would let the value of this argument retain its default value of TRUE).

If the argument divByVar is TRUE, then the sums of squares, from which the test statistic is formed, are are divided by the estimated variance of the quantity being squared. This procedure is analogous to the studentisation procedure used by Hahn, 2012. If divByVar is FALSE then no such division takes place and the sums of squares involved are "raw".

The quantities that are involved in the sums of squares are formed from certain "fitted values" which are weighted means of the observations (observed values of summary functions). The variance referred to is formed as an expression involving weighted means of squares of the residuals. These residuals are of course equal to the observations minus the fitted values.

more detail:

More detail about the test statistic, the fitted values, the residuals and the estimated variance, can be found in the vignette "testStat".

Value

If bylevel is TRUE then the object returned is of class "multi.kanova" which is a list of length equal to then number of levels of the second predictor in the model. Each entry in this list is an object of class "kanova". If keepdata is TRUE then the object in question (of class "multi.kanova") has an attribute "data". This is equal to the data argument, possibly augmented by the value of the second predictor in the model if this predictor was not found in the original data and was located in the parent frame.

Note that the "kanova" components of a "multi.kanova" object *never* themselves have a list entry named "data", irrespective of the value of keepdata. If keepdata is TRUE, then the appropriate data object is returned as an *attribute* of the "multi.kanova" object.

If test is TRUE then the "multi.kanova" object returned has an attribute named "oapv" ("overall p-value"). This is calculated as 1 - (1 - pvmin)^b where pvmin is the minimum of the b p-values obtained when each of the levels of the second predictor is tested individually for an A effect. Note that

$$F(y) = 1 - (1 - y)^b$$

is the pdf of the minimum of b independent observations that are uniformly distributed on [0,1]. Alternatively one may think of the expression for oapv as arising from the Sidak adjustment to the minimum p-value to allow for multiple comparisons.

If bylevel is FALSE then the object returned is of class "kanova", and is described as follows:

If brief is TRUE, then the object in question is a list with components:

Effectname	Character string naming the effect being tested for.
stat	Numerical scalar equal to the value of the test statistic calculated from the original data.
pvalue	The Monte Carlo p-value of the test calculated by comparing stat with test statistics formed from simulated data, generated by permutation, which satisfy the null hypothesis.

If brief is FALSE then the "kanova" object in question has additional components:

nperm	The nperm argument.
permtype	The permtype argument.
Tstar	The vector of nperm values of the test statistic calculated from the simulated data sets.
fmla	The fmla argument.
sumFnNm	The sumFnNm argument.
data	The data argument. This may possibly have been augmented by any predictor values which were not found in the original data and were located in the parent frame.

Components pvalue, nperm and Tstar are present in an object of class "kanova" only if test is TRUE. Component data is present only if keepdata is TRUE.

Warning

When keepdata is TRUE, the way that the data is "kept" depends on the value of bylevel. See **Value**.

Notes

- Simulation experiments have given some evidence that Fest() and Gest() and Jest() lead to tests that have lower power than that obtained than tests obtained by using Kest(). The power obtained seems to be substantially lower in the case of Fest() and Gest(), somewhat lower in the case of Jest().

  Consequently, users are advised to eschew the use of Fest(), Gest() and Jest() (despite their ready availability) and to stick with the default summary function Kest(). Users should ignore this advice *only* if they have a sound reason for doing so and a sound understanding of the consequences.

- Only one-way and two-way (quasi) analyses of variance are accommodated. If you feel inclined to ask why there is no provision for higher order analysis of variance, just look at the code and the answer should be obvious. It *might* be possible to implement higher order quasi analysis of variance of summary functions, but this is unlikely to have any practical use. Writing the code would, for me, be a nightmare!

Author(s)

Rolf Turner <rolfturner@posteo.net>

References

Diggle, Peter J., Mateu, Jorge and Clough, Helen E. (2000) A comparison between parametric and non-parametric approaches to the analysis of replicated spatial point patterns, *Advances in Applied Probability* **32**, pp. 331 – 343.

Diggle, P. J., Lange, N. and Benes, F. (1991) Analysis of variance for replicated spatial point patterns in clinical neuroanatomy, *Journal of the American Statistical Association*, **86**, pp. 618 – 625.

Hahn, Ute (2012) A studentized permutation test for the comparison of spatial point patterns, *Journal of the American Statistical Association*, **107**, pp. 754 – 764, DOI: 10.1080/01621459.2012.688463.

See Also

studpermu.test()

Examples

```
# The following is inappropriate since there is a second
# classification factor.
set.seed(104)
s1 <- kanova(patterns ~ Pos,data=stomata,permtype="d",nperm=9)

# Here we are testing for a Layer effect allowing for a Pos effect.
set.seed(7)
s2a <- kanova(patterns ~ Layer + Pos, data=stomata,permtype="d",nperm=9)
s2b <- kanova(patterns ~ Layer + Pos, data=stomata,permtype="s",nperm=9)

# Here we are testing for a Pos effect allowing for a Layer effect.
set.seed(78)
s3a <- kanova(patterns ~ Pos + Layer, data=stomata,nperm=9)
# permtype defaults to "stdres".
## Not run:  # Takes too long.
    set.seed(24)
    s3b <- kanova(patterns ~ Pos + Layer, data=stomata,nperm=999)
    # Get a p-value of 0.001

## End(Not run)

# Here we are testing for a Pos effect by testing for such an
# effect within each level of Layer.
set.seed(770)
s3c <- kanova(patterns ~ Pos + Layer, bylevel=TRUE,data=stomata,nperm=9)
# attr(s3c,"oapv") is 0.2172 --- not significant.

# Here, we are testing for a Layer by Pos interaction.  Unlikely to
# be meaningful.
set.seed(2)
s4 <- kanova(patterns ~ Layer * Pos, data=stomata,nperm=9) # permtype must be "s"
```

```
# Artificial data.
## Not run:  # Takes too long.
if(requireNamespace("spatstat.geom")) {
    set.seed(3)
    r    <- seq(0,25,length=129)
    rsp  <- lapply(1:144,function(k){pi*r^2 + runif(129,-0.1,0.1)})
    rsp  <- lapply(rsp,function(x){pmax(0,x)})
    fctr1 <- factor(rep(1:4,12,each=3))
    fctr2 <- factor(rep(1:3,48))
    wts  <- sample(50:100,144,replace=TRUE)
    X    <- spatstat.geom::hyperframe(rsp=rsp,fctr1=fctr1,fctr2=fctr2,wts=wts)
    attr(X,"r") <- r
    set.seed(118)
# Testing for a fctr1 effect, allowing for a fctr2 effect.
    s5a  <- kanova(rsp ~ fctr1 + fctr2, data=X,brief=FALSE,nperm=9)
# Testing for an interaction; meaningful in this artificial data
# context.
    s5b  <- kanova(rsp ~ fctr1*fctr2,data=X,brief=FALSE,nperm=9)
}

## End(Not run)

# Scalar data.
## Not run:  # Takes too long.
if(requireNamespace("Devore7")) {
    X   <- spatstat.geom::as.hyperframe(Devore7::xmp11.10)
    s6a <- kanova(Tempr ~ Period*Strain,data=X,nperm=999)
    s6b <- kanova(Tempr ~ Period+Strain,data=X,nperm=999)
    s6c <- kanova(Tempr ~ Strain+Period,data=X,nperm=999)
    chk <- lm(Tempr ~ Period*Strain,data=X)
# Executing anova(chk) reveals p-values that are
# at least roughly similar to those in s6a, s6b, and s6c.
}

## End(Not run)
```

`ripVar` *The Ripley Variance of K-functions.*

Description

Calculates the variance of the K-function of a Poisson point pattern, according to Ripley's formula (as taken from equation (3) in Hahn 2012).

Usage

```
ripVar(X, r)
```

Arguments

X	A point pattern (object of class "ppp"). The variance formula is valid only if X arises from a Poisson process.
r	A numeric vector of non-negative values at which the K-function for X is to be evaluated.

Details

The vector r would normally have entries in increasing order and would have a first entry equal to 0. It may be wise to construct r as Kest(X)$r, but this is not required.

Value

A number vector, of length equal to length(r) whose entries are the variances of K(r) where $K(r)$ is equal to as.function(Kest(X)).

Author(s)

Rolf Turner <rolfturner@posteo.net>

References

Hahn, Ute (2012) A studentized permutation test for the comparison of spatial point patterns, *Journal of the American Statistical Association*, **107**, pp. 754 – 764, DOI: 10.1080/01621459.2012.688463.

See Also

Kest()

Examples

```
if(requireNamespace("spatstat.random")) {
    X    <- spatstat.random::rpoispp(100)
    vKX1 <- ripVar(X,r=0.05*(1:5))
    if(requireNamespace("spatstat.explore")) {
        r    <- spatstat.explore::Kest(X)$r
        vKX2 <- ripVar(X,r=r)
        plot(r,vKX2,type="l")
        points(0.05*(1:5),vKX1)
    }
}
```

`stomata` *Stomata patterns*

Description

Point patterns of stomata at 18 locations in each of 12 leaves of *Michelia cavaleriei var. platypetala.*

Usage

```
stomata
```

Format

The object `stomata` is a hyperframe with 216 rows. It has a column named `patterns` containing the point patterns of stomata locations, a column named `Leaf` which is a factor with levels 1 to 12 identifying the leaf from which the pattern was obtained, a column named `Layer` which is a factor with levels 1 to 6 identifying a location within each "position" (see "`Pos`"), and a column named `Pos` which is a factor with levels 1 to 3, position 1 being closest to the central stem of the leaf and position 3 being closest to the outer edge of the leaf (and farthest from the central stem).

Details

Each pattern was observed in a rectangular window of dimension 1200×900 microns.

Source

The data were kindly supplied by Prof. Peijian Shi of the College of Biology and the Environment, Nanjing Forestry University, Nanjing, P.R. China.

References

Peijian Shi, Yabing Jiao, Peter J. Diggle, Rolf Turner, Rong Wang and Ülo Niinemets 2021. Spatial relationship between stomata of a Magnoliaceae species at the areole level. *Annals of Botany* **128**, pp. 875–885. DOI https://doi.org/10.1093/aob/mcab106

Examples

```
plot(stomata[1,1,drop=TRUE])
```

Index