

Package ‘HDANOVA’

January 21, 2025

Type Package

Title High-Dimensional Analysis of Variance

Version 0.8.2

Date 2025-01-20

Description

Functions and datasets to support Smilde, Marini, Westerhuis and Liland (2025, ISBN: 978-1-394-21121-0)

``Analysis of Variance for High-Dimensional Data - Applications in Life, Food and Chemical Sciences".

This implements and imports a collection of methods for HD-ANOVA data analysis with common interfaces, result- and plotting functions, multiple real data sets and four vignettes covering a range different applications.

Depends R (>= 3.5.0)

Imports car, lme4, mixlm (>= 1.4.2), pls, pracma, progress, RSpecra

Suggests knitr, vegan

License GPL (>= 2)

URL <https://khliland.github.io/HDANOVA/>,
<https://github.com/khliland/HDANOVA/>

BugReports <https://github.com/khliland/HDANOVA/issues/>

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Kristian Hovde Liland [aut, cre]
(<<https://orcid.org/0000-0001-6468-9423>>)

Maintainer Kristian Hovde Liland <kristian.liland@nmbu.no>

Repository CRAN

Date/Publication 2025-01-21 13:30:07 UTC

Contents

apca	2
asca	3
asca_fit	5
asca_plots	7
asca_results	9
block.data.frame	10
caldana	11
candies	12
dummycode	12
extended.model.frame	13
limmpca	14
model.frame.asca	15
msca	16
pcanova	17
pcanova_plots	19
pcanova_results	20
permanova	21
prc	22
timeplot	23
update_without_factor	24
Index	25

apca	<i>ANOVA Principal Component Analysis - APCA</i>
------	--

Description

APCA function for fitting ANOVA Principal Component Analysis models.

Usage

```
apca(formula, data, add_error = TRUE, contrasts = "contr.sum", ...)
```

Arguments

formula	Model formula accepting a single response (block) and predictors.
data	The data set to analyse.
add_error	Add error to LS means (default = TRUE).
contrasts	Effect coding: "sum" (default = sum-coding), "weighted", "reference", "treatment".
...	Additional parameters for the asca_fit function.

Value

An object of class `apca`, inheriting from the general `asca` class. Further arguments and plots can be found in the [asca](#) documentation.

References

Harrington, P.d.B., Vieira, N.E., Espinoza, J., Nien, J.K., Romero, R., and Yergey, A.L. (2005) Analysis of variance–principal component analysis: A soft tool for proteomic discovery. *Analytica chimica acta*, 544 (1-2), 118–127.

See Also

Main methods: [asca](#), [apca](#), [limmpca](#), [msca](#), [pcanova](#), [prc](#) and [permanova](#). Workhorse function underpinning most methods: [asca_fit](#). Extraction of results and plotting: [asca_results](#), [asca_plots](#), [pcanova_results](#) and [pcanova_plots](#)

Examples

```
data(candies)
ap <- apca(assessment ~ candy, data=candies)
scoreplot(ap)

# Numeric effects
candies$num <- eff <- 1:165
mod <- apca(assessment ~ candy + assessor + num, data=candies)
summary(mod)
scoreplot(mod, factor=3, gr.col=rgb(eff/max(eff), 1-eff/max(eff),0), pch.scores="x")
```

 asca

Analysis of Variance Simultaneous Component Analysis - ASCA

Description

This is a quite general and flexible implementation of ASCA.

Usage

```
asca(formula, data, contrasts = "contr.sum", ...)
```

Arguments

<code>formula</code>	Model formula accepting a single response (block) and predictors. See Details for more information.
<code>data</code>	The data set to analyse.
<code>contrasts</code>	Effect coding: "sum" (default = sum-coding), "weighted", "reference", "treatment".
<code>...</code>	Additional arguments to asca_fit .

Details

ASCA is a method which decomposes a multivariate response according to one or more design variables. ANOVA is used to split variation into contributions from factors, and PCA is performed on the corresponding least squares estimates, i.e., $Y = X_1 B_1 + X_2 B_2 + \dots + E = T_1 P_1' + T_2 P_2' + \dots + E$. This version of ASCA encompasses variants of LiMM-PCA, generalized ASCA and covariates ASCA. It includes confidence ellipsoids for the balanced crossed-effect ASCA.

The formula interface is extended with the function `r()` to indicate random effects and `comb()` to indicate effects that should be combined. See Examples for use cases.

Value

An `asca` object containing loadings, scores, explained variances, etc. The object has associated plotting ([asca_plots](#)) and result ([asca_results](#)) functions.

References

- Smilde, A., Jansen, J., Hoefsloot, H., Lamers, R., Van Der Greef, J., and Timmerman, M. (2005). ANOVA-Simultaneous Component Analysis (ASCA): A new tool for analyzing designed metabolomics data. *Bioinformatics*, 21(13), 3043–3048.
- Liland, K.H., Smilde, A., Marini, F., and Næs, T. (2018). Confidence ellipsoids for ASCA models based on multivariate regression theory. *Journal of Chemometrics*, 32(e2990), 1–13.
- Martin, M. and Govaerts, B. (2020). LiMM-PCA: Combining ASCA+ and linear mixed models to analyse high-dimensional designed data. *Journal of Chemometrics*, 34(6), e3232.

See Also

Main methods: [asca](#), [apca](#), [limmpca](#), [msca](#), [pcanova](#), [prc](#) and [permanova](#). Workhorse function underpinning most methods: [asca_fit](#). Extraction of results and plotting: [asca_results](#), [asca_plots](#), [pcanova_results](#) and [pcanova_plots](#)

Examples

```
# Load candies data
data(candies)

# Basic ASCA model with two factors
mod <- asca(assessment ~ candy + assessor, data=candies)
print(mod)

# ASCA model with interaction
mod <- asca(assessment ~ candy * assessor, data=candies)
print(mod)

# Result plotting for first factor
loadingplot(mod, scatter=TRUE, labels="names")
scoreplot(mod)
# No backprojection
scoreplot(mod, projections=FALSE)
# Spider plot
```

```
scoreplot(mod, spider=TRUE, projections=FALSE)

# ASCA model with compressed response using 5 principal components
mod.pca <- asca(assessment ~ candy + assessor, data=candies, pca.in=5)

# Mixed Model ASCA, random assessor
mod.mix <- asca(assessment ~ candy + r(assessor), data=candies)
scoreplot(mod.mix)

# Mixed Model ASCA, REML estimation
mod.mix <- asca(assessment ~ candy + r(assessor), data=candies, REML=TRUE)
scoreplot(mod.mix)

# Load Caldana data
data(caldana)

# Combining effects in ASCA
mod.comb <- asca(compounds ~ time + comb(light + time:light), data=caldana)
summary(mod.comb)
timeplot(mod.comb, factor="light", time="time", comb=2)

# Permutation testing
mod.perm <- asca(assessment ~ candy * assessor, data=candies, permute=TRUE)
summary(mod.perm)
```

asca_fit

ASCA Fitting Workhorse Function

Description

This function is called by all ASCA related methods in this package. It is documented so that one can have access to a richer set of parameters from the various methods or call this function directly. The latter should be done with care as there are many possibilities and not all have been used in publications or tested thoroughly.

Usage

```
asca_fit(
  formula,
  data,
  subset,
  weights,
  na.action,
  family,
  permute = FALSE,
  perm.type = c("approximate", "exact"),
  unrestricted = FALSE,
  add_error = FALSE,
```

```

    aug_error = "denominator",
    use_ED = FALSE,
    pca.in = FALSE,
    contrasts = "contr.sum",
    coding,
    equal_baseline = FALSE,
    SStype = "II",
    REML = NULL
  )

```

Arguments

formula	Model formula accepting a single response (block) and predictors. See Details for more information.
data	The data set to analyse.
subset	Expression for subsetting the data before modelling.
weights	Optional object weights.
na.action	How to handle NAs (no action implemented).
family	Error distributions and link function for Generalized Linear Models.
permute	Perform approximate permutation testing, default = FALSE (numeric or TRUE = 1000 permutations).
perm.type	Type of permutation: "approximate" (default) or "exact".
unrestricted	Use unrestricted ANOVA decomposition (default = FALSE).
add_error	Add error to LS means, e.g., for APCA.
aug_error	Augment score matrices in backprojection. Default = "denominator" (of F test), "residual" (force error term), numeric value (alpha-value in LiMM-PCA).
use_ED	Use "effective dimensions" for score rescaling in LiMM-PCA.
pca.in	Compress response before ASCA (number of components).
contrasts	Effect coding: "sum" (default = sum-coding), "weighted", "reference", "treatment".
coding	Defunct. Use 'contrasts' instead.
equal_baseline	Experimental: Set to TRUE to let interactions, where a main effect is missing, e.g., a nested model, be handled with the same baseline as a cross effect model. If TRUE the corresponding interactions will be put in quotation marks and included in the model.frame.
SStype	Type of sum-of-squares: "I" = sequential, "II" (default) = last term, obeying marginality, "III" = last term, not obeying marginality.
REML	Parameter to mixlm: NULL (default) = sum-of-squares, TRUE = REML, FALSE = ML.

Value

An asca object containing loadings, scores, explained variances, etc. The object has associated plotting ([asca_plots](#)) and result ([asca_results](#)) functions.

Description

Various plotting procedures for [asca](#) objects.

Usage

```
## S3 method for class 'asca'
loadingplot(object, factor = 1, comps = 1:2, ...)

## S3 method for class 'asca'
scoreplot(
  object,
  factor = 1,
  comps = 1:2,
  within_level = "all",
  pch.scores = 19,
  pch.projections = 1,
  gr.col = NULL,
  projections = TRUE,
  spider = FALSE,
  ellipsoids,
  confidence,
  xlim,
  ylim,
  xlab,
  ylab,
  legendpos,
  ...
)

permutationplot(object, factor = 1, xlim, xlab = "SSQ", main, ...)
```

Arguments

object	asca object.
factor	integer/character for selecting a model factor. If factor <= 0 or "global", the PCA of the input is used (negativ factor to include factor level colouring with global PCA).
comps	integer vector of selected components.
...	additional arguments to underlying methods.
within_level	MSCA parameter for choosing plot level (default = "all").
pch.scores	integer plotting symbol.

<code>pch.projections</code>	integer plotting symbol.
<code>gr.col</code>	integer vector of colours for groups.
<code>projections</code>	Include backprojections in score plot (default = TRUE).
<code>spider</code>	Draw lines between group centers and backprojections (default = FALSE).
<code>ellipsoids</code>	character "confidence" or "data" ellipsoids for balanced fixed effect models.
<code>confidence</code>	numeric vector of ellipsoid confidences, default = c(0.4, 0.68, 0.95).
<code>xlim</code>	numeric x limits.
<code>ylim</code>	numeric y limits.
<code>xlab</code>	character x label.
<code>ylab</code>	character y label.
<code>legendpos</code>	character position of legend.
<code>main</code>	Plot title.

Details

Usage of the functions are shown using generics in the examples in [asca](#). Plot routines are available as `scoreplot.asca` and `loadingplot.asca`.

Value

The plotting routines have no return.

References

- Smilde, A., Jansen, J., Hoefsloot, H., Lamers, R., Van Der Greef, J., and Timmerman, M. (2005). ANOVA-Simultaneous Component Analysis (ASCA): A new tool for analyzing designed metabolomics data. *Bioinformatics*, 21(13), 3043–3048.
- Liland, K.H., Smilde, A., Marini, F., and Næs, T. (2018). Confidence ellipsoids for ASCA models based on multivariate regression theory. *Journal of Chemometrics*, 32(e2990), 1–13.
- Martin, M. and Govaerts, B. (2020). LiMM-PCA: Combining ASCA+ and linear mixed models to analyse high-dimensional designed data. *Journal of Chemometrics*, 34(6), e3232.

See Also

Main methods: [asca](#), [apca](#), [limmpca](#), [msca](#), [pcanova](#), [prc](#) and [permanova](#). Workhorse function underpinning most methods: [asca_fit](#). Extraction of results and plotting: [asca_results](#), [asca_plots](#), [pcanova_results](#) and [pcanova_plots](#)

Description

Standard result computation and extraction functions for ASCA ([asca](#)).

Usage

```
## S3 method for class 'asca'
print(x, ...)

## S3 method for class 'asca'
summary(object, extended = TRUE, df = FALSE, ...)

## S3 method for class 'summary.asca'
print(x, digits = 2, ...)

## S3 method for class 'asca'
loadings(object, factor = 1, ...)

## S3 method for class 'asca'
scores(object, factor = 1, ...)

projections(object, ...)

## S3 method for class 'asca'
projections(object, factor = 1, ...)
```

Arguments

x	asca object.
...	additional arguments to underlying methods.
object	asca object.
extended	Extended output in summary (default = TRUE).
df	Show degrees of freedom in summary (default = FALSE).
digits	integer number of digits for printing.
factor	integer/character for selecting a model factor.

Details

Usage of the functions are shown using generics in the examples in [asca](#). Explained variances are available (block-wise and global) through `blockexpl` and `print.rosaexpl`. Object printing and summary are available through: `print.asca` and `summary.asca`. Scores and loadings have their own extensions of `scores()` and `loadings()` through `scores.asca` and `loadings.asca`. Special to ASCA is that scores are on a factor level basis, while back-projected samples have their own function in `projections.asca`.

Value

Returns depend on method used, e.g. `projections.asca` returns projected samples, `scores.asca` return scores, while `print` and `summary` methods return the object invisibly.

References

- Smilde, A., Jansen, J., Hoefsloot, H., Lamers, R., Van Der Greef, J., and Timmerman, M. (2005). ANOVA-Simultaneous Component Analysis (ASCA): A new tool for analyzing designed metabolomics data. *Bioinformatics*, 21(13), 3043–3048.
- Liland, K.H., Smilde, A., Marini, F., and Næs, T. (2018). Confidence ellipsoids for ASCA models based on multivariate regression theory. *Journal of Chemometrics*, 32(e2990), 1–13.
- Martin, M. and Govaerts, B. (2020). LiMM-PCA: Combining ASCA+ and linear mixed models to analyse high-dimensional designed data. *Journal of Chemometrics*, 34(6), e3232.

See Also

Main methods: [asca](#), [apca](#), [limmpca](#), [msca](#), [pcanova](#), [prc](#) and [permanova](#). Workhorse function underpinning most methods: [asca_fit](#). Extraction of results and plotting: [asca_results](#), [asca_plots](#), [pcanova_results](#) and [pcanova_plots](#)

`block.data.frame`

Block-wise indexable data.frame

Description

This is a convenience function for making `data.frames` that are easily indexed on a block-wise basis.

Usage

```
block.data.frame(X, block_inds = NULL, to.matrix = TRUE)
```

Arguments

<code>X</code>	Either a single <code>data.frame</code> to index or a list of matrices/ <code>data.frames</code>
<code>block_inds</code>	Named list of indexes if <code>X</code> is a single <code>data.frame</code> , otherwise <code>NULL</code> .
<code>to.matrix</code>	logical indicating if input list elements should be converted to matrices.

Value

A `data.frame` which can be indexed block-wise.

See Also

Main methods: [asca](#), [apca](#), [limmpca](#), [msca](#), [pcanova](#), [prc](#) and [permanova](#). Workhorse function underpinning most methods: [asca_fit](#). Extraction of results and plotting: [asca_results](#), [asca_plots](#), [pcanova_results](#) and [pcanova_plots](#)

Examples

```
# Random data
M <- matrix(rnorm(200), nrow = 10)
# .. with dimnames
dimnames(M) <- list(LETTERS[1:10], as.character(1:20))

# A named list for indexing
inds <- list(B1 = 1:10, B2 = 11:20)

X <- block.data.frame(M, inds)
str(X)
```

caldana

Arabidopsis thaliana growth experiment

Description

A dataset containing 67 metabolites from plants grown under different light and temperature conditions. This subset of the data contains only the light effect and time effect for limited conditions, while the full data also contains gene expressions.

Usage

```
data(caldana)
```

Format

A data.frame having 140 rows and 3 variables:

light Light levels

time Time of measurement

compound Metabolic compounds

References

Caldana C, Degenkolbe T, Cuadros-Inostroza A, Klie S, Sulpice R, Leisse A, et al. High-density kinetic analysis of the metabolomic and transcriptomic response of Arabidopsis to eight environmental conditions. *Plant J.* 2011;67(5):869-884.

candies	<i>Sensory assessment of candies.</i>
---------	---------------------------------------

Description

A dataset containing 9 sensory attributes for 5 candies assessed by 11 trained assessors.

Usage

```
data(candies)
```

Format

A data.frame having 165 rows and 3 variables:

assessment Matrix of sensory attributes

assessor Factor of assessors

candy Factor of candies

References

Luciano G, Næs T. Interpreting sensory data by combining principal component analysis and analysis of variance. *Food Qual Prefer.* 2009;20(3):167-175.

dummycode	<i>Dummy-coding of a single vector</i>
-----------	--

Description

Flexible dummy-coding allowing for all R's built-in types of contrasts and optional dropping of a factor level to reduce rank deficiency probability.

Usage

```
dummycode(Y, contrast = "contr.sum", drop = TRUE)
```

Arguments

Y vector to dummy code.

contrast Contrast type, default = "contr.sum".

drop logical indicating if one level should be dropped (default = TRUE).

Value

matrix made by dummy-coding the input vector.

Examples

```
vec <- c("a", "a", "b", "b", "c", "c")
dummycode(vec)
```

extended.model.frame *Extracting the Extended Model Frame from a Formula or Fit*

Description

This function attempts to apply [model.frame](#) and extend the result with columns of interactions.

Usage

```
extended.model.frame(formula, data, ..., sep = ".")
```

Arguments

formula	a model formula or terms object or an R object.
data	a data.frame, list or environment (see model.frame).
...	further arguments to pass to model.frame .
sep	separator in contraction of names for interactions (default = ".").

Value

A [data.frame](#) that includes everything a [model.frame](#) does plus interaction terms.

See Also

Main methods: [asca](#), [apca](#), [limmpca](#), [msca](#), [pcanova](#), [prc](#) and [permanova](#). Workhorse function underpinning most methods: [asca_fit](#). Extraction of results and plotting: [asca_results](#), [asca_plots](#), [pcanova_results](#) and [pcanova_plots](#)

Examples

```
dat <- data.frame(Y = c(1,2,3,4,5,6),
                 X = factor(LETTERS[c(1,1,2,2,3,3)]),
                 W = factor(letters[c(1,2,1,2,1,2)]))
extended.model.frame(Y ~ X*W, dat)
```

limmpca

*Linear Mixed Model PCA***Description**

This function mimics parts of the LiMM-PCA framework, combining ASCA+ and linear mixed models to analyse high-dimensional designed data. The default is to use REML estimation and scaling of the backprojected errors. See examples for alternatives.

Usage

```
limmpca(
  formula,
  data,
  pca.in = 5,
  aug_error = 0.05,
  use_ED = FALSE,
  REML = TRUE,
  contrasts = "contr.sum",
  ...
)
```

Arguments

formula	Model formula accepting a single response (block) and predictors. See Details for more information.
data	The data set to analyse.
pca.in	Compress response before ASCA (number of components), default = 5.
aug_error	Error term of model ("denominator", "residual", numeric alpha-value). The latter implies the first with a scaling factor.
use_ED	Use Effective Dimensions instead of degrees of freedom when scaling.
REML	Use restricted maximum likelihood estimation. Alternatives: TRUE (default), FALSE (ML), NULL (least squares).
contrasts	Effect coding: "sum" (default = sum-coding), "weighted", "reference", "treatment".
...	Additional arguments to asca_fit .

Value

An object of class `limmpca`, inheriting from the general `asca` class.

References

- Martin, M. and Govaerts, B. (2020). LiMM-PCA: Combining ASCA+ and linear mixed models to analyse high-dimensional designed data. *Journal of Chemometrics*, 34(6), e3232.

See Also

Main methods: [asca](#), [apca](#), [limmpca](#), [msca](#), [pcanova](#), [prc](#) and [permanova](#). Workhorse function underpinning most methods: [asca_fit](#). Extraction of results and plotting: [asca_results](#), [asca_plots](#), [pcanova_results](#) and [pcanova_plots](#)

Examples

```
# Load candies data
data(candies)

# Default LiMM-PCA model with two factors and interaction, 5 PCA components
mod <- limmpca(assessment ~ candy*r(assessor), data=candies)
summary(mod)
scoreplot(mod, factor = "candy")

# LiMM-PCA with least squares estimation and 8 PCA components
modLS <- limmpca(assessment ~ candy*r(assessor), data=candies, REML=NULL, pca.in=8)
summary(modLS)
scoreplot(modLS, factor = "candy")

# Load Caldana data
data(caldana)

# Combining effects in LiMM-PCA (assuming light is a random factor)
mod.comb <- limmpca(compounds ~ time + comb(r(light) + r(time:light)), data=caldana, pca.in=8)
summary(mod.comb)
```

model.frame.asca

Model Frame and Model Matrix for ASCA-like Models

Description

Extraction functions to retrieve the `model.frame` and `model.matrix` of an `asca` object.

Usage

```
## S3 method for class 'asca'
model.frame(formula, ...)

## S3 method for class 'asca'
model.matrix(object, ...)
```

Arguments

<code>formula</code>	The <code>asca</code> object.
<code>...</code>	Not implemented
<code>object</code>	The <code>asca</code> object.

Value

A data.frame or matrix object.

Examples

```
# Load candies data
data(candies)

# Basic ASCA model with two factors
mod <- asca(assessment ~ candy + assessor, data=candies)

# Extract model frame and model matrix
mf <- model.frame(mod)
head(mf)
mm <- model.matrix(mod)
par.old <- par(mar=c(3,3,3,1), mgp=c(1,0.7,0))
image(t(mm[seq(165,1,-1),]), main="Model Matrix", xlab="dummy values", ylab="samples",
      axes=FALSE)
par(par.old)
```

msca

Multilevel Simultaneous Component Analysis - MSCA

Description

This MSCA implementation assumes a single factor to be used as between-individuals factor.

Usage

```
msca(formula, data, contrasts = "contr.sum", ...)
```

Arguments

formula	Model formula accepting a single response (block) and predictors. See Details for more information.
data	The data set to analyse.
contrasts	Effect coding: "sum" (default = sum-coding), "weighted", "reference", "treatment".
...	Additional arguments to asca_fit .

Value

An asca object containing loadings, scores, explained variances, etc. The object has associated plotting ([asca_plots](#)) and result ([asca_results](#)) functions.

References

- Smilde, A., Jansen, J., Hoefsloot, H., Lamers, R., Van Der Greef, J., and Timmerman, M. (2005). ANOVA-Simultaneous Component Analysis (ASCA): A new tool for analyzing designed metabolomics data. *Bioinformatics*, 21(13), 3043–3048.
- Liland, K.H., Smilde, A., Marini, F., and Næs, T. (2018). Confidence ellipsoids for ASCA models based on multivariate regression theory. *Journal of Chemometrics*, 32(e2990), 1–13.

See Also

Main methods: [asca](#), [apca](#), [limmpca](#), [msca](#), [pcanova](#), [prc](#) and [permanova](#). Workhorse function underpinning most methods: [asca_fit](#). Extraction of results and plotting: [asca_results](#), [asca_plots](#), [pcanova_results](#) and [pcanova_plots](#)

Examples

```
# Load candies data
data(candies)

# Basic MSCA model with a single factor
mod <- msca(assessment ~ candy, data=candies)
print(mod)
summary(mod)

# Result plotting for first factor
loadingplot(mod, scatter=TRUE, labels="names")
scoreplot(mod)

# Within scores
scoreplot(mod, factor="within")

# Within scores per factor level
par.old <- par(mfrow=c(3,2), mar=c(4,4,2,1), mgp=c(2,0.7,0))
for(i in 1:length(mod$scores.within))
  scoreplot(mod, factor="within", within_level=i,
            main=paste0("Level: ", names(mod$scores.within)[i]),
            panel.first=abline(v=0,h=0,col="gray",lty=2))
par(par.old)

# Permutation testing
mod.perm <- asca(assessment ~ candy * assessor, data=candies, permute=TRUE)
summary(mod.perm)
```

Description

This is a quite general and flexible implementation of PC-ANOVA.

Usage

```
pcanova(formula, data, ncomp = 0.9, contrasts = "contr.sum", ...)
```

Arguments

formula	Model formula accepting a single response (block) and predictor names separated by + signs.
data	The data set to analyse.
ncomp	The number of components to retain, proportion of variation or default = minimum cross-validation error.
contrasts	Effect coding: "sum" (default = sum-coding), "weighted", "reference", "treatment".
...	Additional parameters for the <code>asca_fit</code> function.

Details

PC-ANOVA works in the opposite order of ASCA. First the response matrix is decomposed using ANOVA. Then the components are analysed using ANOVA with respect to a design or grouping in the data. The latter can be ordinary fixed effects modelling or mixed models.

Value

A `pcanova` object containing loadings, scores, explained variances, etc. The object has associated plotting (`pcanova_plots`) and result (`pcanova_results`) functions.

References

Luciano G, Næs T. Interpreting sensory data by combining principal component analysis and analysis of variance. *Food Qual Prefer.* 2009;20(3):167-175.

See Also

Main methods: [asca](#), [apca](#), [limmpca](#), [msca](#), [pcanova](#), [prc](#) and [permanova](#). Workhorse function underpinning most methods: [asca_fit](#). Extraction of results and plotting: [asca_results](#), [asca_plots](#), [pcanova_results](#) and [pcanova_plots](#)

Examples

```
# Load candies data
data(candies)

# Basic PC-ANOVA model with two factors, cross-validated opt. of #components
mod <- pcanova(assessment ~ candy + assessor, data = candies)
print(mod)
```

```

# PC-ANOVA model with interaction, minimum 90% explained variance
mod <- pcanova(assessment ~ candy * assessor, data = candies, ncomp = 0.9)
print(mod)
summary(mod)

# Tukey group letters for 'candy' per component
lapply(mod$models, function(x)
  mixlm::cld(mixlm::simple.glt(x,
    effect = "candy")))

# Result plotting
loadingplot(mod, scatter=TRUE, labels="names")
scoreplot(mod)

# Mixed Model PC-ANOVA, random assessor
mod.mix <- pcanova(assessment ~ candy + r(assessor), data=candies, ncomp = 0.9)
scoreplot(mod.mix)
# Fixed effects
summary(mod.mix)

```

pcanova_plots

PC-ANOVA Result Methods

Description

Various plotting procedures for [pcanova](#) objects.

Usage

```

## S3 method for class 'pcanova'
scoreplot(object, factor = 1, comps = 1:2, col = "factor", ...)

```

Arguments

object	pcanova object.
factor	integer/character for selecting a model factor.
comps	integer vector of selected components.
col	character for selecting a factor to use for colouring (default = first factor) or ordinary colour specifications.
...	additional arguments to underlying methods.

Details

Usage of the functions are shown using generics in the examples in [pcanova](#). Plot routines are available as `scoreplot.pcanova` and `loadingplot.pcanova`.

Value

The plotting routines have no return.

References

Luciano G, Næs T. Interpreting sensory data by combining principal component analysis and analysis of variance. *Food Qual Prefer.* 2009;20(3):167-175.

See Also

Main methods: [asca](#), [apca](#), [limmpca](#), [msca](#), [pcanova](#), [prc](#) and [permanova](#). Workhorse function underpinning most methods: [asca_fit](#). Extraction of results and plotting: [asca_results](#), [asca_plots](#), [pcanova_results](#) and [pcanova_plots](#)

pcanova_results	<i>PC-ANOVA Result Methods</i>
-----------------	--------------------------------

Description

Standard result computation and extraction functions for ASCA ([pcanova](#)).

Usage

```
## S3 method for class 'pcanova'
summary(object, ...)

## S3 method for class 'summary.pcanova'
print(x, digits = 2, ...)

## S3 method for class 'pcanova'
print(x, ...)

## S3 method for class 'pcanova'
summary(object, ...)
```

Arguments

object	pcanova object.
...	additional arguments to underlying methods.
x	pcanova object.
digits	integer number of digits for printing.

Details

Usage of the functions are shown using generics in the examples in [pcanova](#). Explained variances are available (block-wise and global) through `blockexpl` and `print.rosaexpl`. Object printing and summary are available through: `print.pcanova` and `summary.pcanova`. Scores and loadings have their own extensions of `scores()` and `loadings()` through `scores.pcanova` and `loadings.pcanova`. Special to ASCA is that scores are on a factor level basis, while back-projected samples have their own function in `projections.pcanova`.

Value

Returns depend on method used, e.g. `projections.pcanova` returns projected samples, `scores.pcanova` return scores, while `print` and `summary` methods return the object invisibly.

References

Luciano G, Næs T. Interpreting sensory data by combining principal component analysis and analysis of variance. *Food Qual Prefer.* 2009;20(3):167-175.

See Also

Main methods: [asca](#), [apca](#), [limmpca](#), [msca](#), [pcanova](#), [prc](#) and [permanova](#). Workhorse function underpinning most methods: [asca_fit](#). Extraction of results and plotting: [asca_results](#), [asca_plots](#), [pcanova_results](#) and [pcanova_plots](#)

permanova

Permutation Based MANOVA - PERMANOVA

Description

Wrapper for the [adonis2](#) function to allow ordinary formula input.

Usage

```
permanova(formula, data, ...)
```

Arguments

formula	Model formula accepting a single response matrix and predictors. See details in adonis2 .
data	The data set to analyse.
...	Additional arguments to adonis2 .

Value

An ANOVA table with permutation-based p-values.

Examples

```
data(caldana)
(pr <- permanova(compounds ~ light * time, caldana))
```

prc

Principal Response Curves

Description

Wrapper for the [prc](#) function to allow for formula input.

Usage

```
prc(formula, data, ...)
```

Arguments

formula	Model formula accepting a single response (block) and predictors. If no predictor is called 'time', time is assumed to be the second predictor.
data	The data set to analyse.
...	Additional arguments to prc .

Value

An object of class `prc`.

See Also

Main methods: [asca](#), [apca](#), [limmpca](#), [msca](#), [pcanova](#), [prc](#) and [permanova](#). Workhorse function underpinning most methods: [asca_fit](#). Extraction of results and plotting: [asca_results](#), [asca_plots](#), [pcanova_results](#) and [pcanova_plots](#)

Examples

```
data(caldana)
(pr <- prc(compounds ~ light * time, caldana))
summary(pr)
```

`timeplot`*Timeplot for Combined Effects*

Description

Timeplot for Combined Effects

Usage

```
timeplot(  
  object,  
  factor,  
  time,  
  comb,  
  comp = 1,  
  ylim,  
  x_time = FALSE,  
  xlab = time,  
  ylab = paste0("Score ", comp),  
  lwd = 2,  
  ...  
)
```

Arguments

<code>object</code>	asca object.
<code>factor</code>	integer/character main factor.
<code>time</code>	integer/character time factor.
<code>comb</code>	integer/character combined effect factor.
<code>comp</code>	integer component number.
<code>ylim</code>	numeric y limits.
<code>x_time</code>	logical use time levels as non-equispaced x axis (default = FALSE).
<code>xlab</code>	character x label.
<code>ylab</code>	character y label.
<code>lwd</code>	numeric line width.
<code>...</code>	additional arguments to plot.

Value

Nothing

Examples

```
data("caldana")
mod.comb <- asca(compounds ~ time + comb(light + light:time), data=caldana)

# Default time axis
timeplot(mod.comb, factor="light", time="time", comb=2)

# Non-equispaced time axis (using time levels)
timeplot(mod.comb, factor="light", time="time", comb=2, x_time=TRUE)

# Second component
timeplot(mod.comb, factor="light", time="time", comb=2, comp=2, x_time=TRUE)
```

update_without_factor *Update a Model without Factor*

Description

Perform a model update while removing a chosen factor. Hierarchical corresponds to type "II" sum-of-squares, i.e., obeying marginality, while non-hierarchical corresponds to type "III" sum-of-squares.

Usage

```
update_without_factor(model, fac, hierarchical = TRUE)
```

Arguments

model	model object to update.
fac	character factor to remove.
hierarchical	logical obey hierarchy when removing factor (default = TRUE).

Value

An updated model object is returned. If the supplied model is of type `lmerMod` and no random effects are left, the model is automatically converted to a linear model before updating.

Index

adonis2, [21](#)
apca, [2](#), [3](#), [4](#), [8](#), [10](#), [13](#), [15](#), [17](#), [18](#), [20–22](#)
asca, [3](#), [3](#), [4](#), [7–10](#), [13](#), [15](#), [17](#), [18](#), [20–22](#)
asca_fit, [3](#), [4](#), [5](#), [8](#), [10](#), [13–18](#), [20–22](#)
asca_plots, [3](#), [4](#), [6](#), [7](#), [8](#), [10](#), [13](#), [15–18](#), [20–22](#)
asca_results, [3](#), [4](#), [6](#), [8](#), [9](#), [10](#), [13](#), [15–18](#),
[20–22](#)

block.data.frame, [10](#)

caldana, [11](#)
candies, [12](#)

data.frame, [13](#)
dummycode, [12](#)

extended.model.frame, [13](#)

limmpca, [3](#), [4](#), [8](#), [10](#), [13](#), [14](#), [15](#), [17](#), [18](#), [20–22](#)
loadingplot.asca (asca_plots), [7](#)
loadingplot.pcanova (pcanova_plots), [19](#)
loadings.asca (asca_results), [9](#)

model.frame, [13](#)
model.frame.asca, [15](#)
model.matrix.asca (model.frame.asca), [15](#)
msca, [3](#), [4](#), [8](#), [10](#), [13](#), [15](#), [16](#), [17](#), [18](#), [20–22](#)

pcanova, [3](#), [4](#), [8](#), [10](#), [13](#), [15](#), [17](#), [17](#), [18–22](#)
pcanova_plots, [3](#), [4](#), [8](#), [10](#), [13](#), [15](#), [17](#), [18](#), [19](#),
[20–22](#)
pcanova_results, [3](#), [4](#), [8](#), [10](#), [13](#), [15](#), [17](#), [18](#),
[20](#), [20](#), [21](#), [22](#)
permanova, [3](#), [4](#), [8](#), [10](#), [13](#), [15](#), [17](#), [18](#), [20](#), [21](#),
[21](#), [22](#)
permutationplot (asca_plots), [7](#)
prc, [3](#), [4](#), [8](#), [10](#), [13](#), [15](#), [17](#), [18](#), [20–22](#), [22](#)
print.asca (asca_results), [9](#)
print.pcanova (pcanova_results), [20](#)
print.summary.asca (asca_results), [9](#)
print.summary.pcanova
 (pcanova_results), [20](#)
projections (asca_results), [9](#)
projections.pcanova (pcanova_results),
[20](#)

scoreplot.asca (asca_plots), [7](#)
scoreplot.pcanova (pcanova_plots), [19](#)
scores.asca (asca_results), [9](#)
summary.asca (asca_results), [9](#)
summary.pcanova (pcanova_results), [20](#)

timeplot, [23](#)

update_without_factor, [24](#)