Package: HDANOVA (via r-universe)

October 18, 2026

Type Package

Title High-Dimensional Analysis of Variance

Version 0.8.6

Date 2026-04-22

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, MASS, mixlm (>= 1.4.3), pls, pracma, progress, Rcpp, reformulas, RSpectra

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, testthat (>= 3.0.0), vegan

Config/testthat/edition 3

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

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

Config/pak/sysreqs cmake make libicu-dev

Repository https://khliland.r-universe.dev

Date/Publication 2026-06-01 07:26:00 UTC

RemoteUrl https://github.com/khliland/hdanova

RemoteRef HEAD

RemoteSha 20e3ed3e9f98d5b1f5a044541a9414229171b3d6

Contents

apca . . . 3
apls . . . 4
asca . . . 6
asca_plots . . . 9
asca_results . . . 11
biplot.asca . . . 13
block.data.frame . . . 14
caldana . . . 15
candies . . . 16
dummycode . . . 16
extended.model.frame . . . 17
extract_estimates . . . 18
hdanova . . . 19
limmpca . . . 22
logLik.hdanova . . . 24
model.frame.asca . . . 25
msca . . . 26
pcanova . . . 28
pcanova_plots . . . 29
pcanova_results . . . 30
permanova . . . 31
permutation . . . 32
pls . . . 33
prc . . . 34
predict.hdanova . . . 35
predict_asca_family . . . 36
rotation . . . 38
sca . . . 39
signflip . . . 40
timeplot . . . 41
update_without_factor . . . 42

Index **43**

apca *ANOVA Principal Component Analysis - APCA*

Description

APCA function for fitting ANOVA Principal Component Analysis models.

Usage

```
apca(
  formula,
  data,
  add_error = TRUE,
  contrasts = "contr.sum",
  permute = FALSE,
  perm.type = c("approximate", "exact"),
  ...
)
```

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".
permute	Number of permutations to perform (default = 1000).
perm.type	Type of permutation to perform, either "approximate" or "exact" (default = "approximate").
...	Additional parameters for the hdanova 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: hdanova. 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")
```

---

apls *Analysis of Variance Partial Least Squares - APLS*

Description

This is a quite general and flexible implementation of APLS.

Usage

```
apls(
  formula,
  data,
  add_error = TRUE,
  contrasts = "contr.sum",
  permute = FALSE,
  perm.type = c("approximate", "exact"),
  ...
)
```

Arguments

formula	Model formula accepting a single response (block) and predictors. See Details for more information.
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".
permute	Number of permutations to perform (default = 1000).
perm.type	Type of permutation to perform, either "approximate" or "exact" (default = "approximate").
...	Additional arguments to hdanova.

Details

APLS 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 PLS is performed on the corresponding least squares estimates, i.e., `Y = X1 B1 + X2 B2 + ... + E = T1 P1' + T2 P2' + ... + E`. For balanced designs, the PLS components are equivalent to PCA components, i.e., APLS and APCA are equivalent. This version of APLS encompasses variants of LiMM-PLS, generalized APLS and covariates APLS.

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 `apls` 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: `hdanova`. Extraction of results and plotting: `asca_results`, `asca_plots`, `pcanova_results` and `pcanova_plots`

Examples

```
# Load candies data
data(candies)

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

# APLS model with interaction
mod <- apls(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)

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

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

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

# Load Caldana data
data(caldana)

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

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

---

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",
  permute = FALSE,
  perm.type = c("approximate", "exact"),
  ...
)
```

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".
permute	Number of permutations to perform (default = 1000).
perm.type	Type of permutation to perform, either "approximate" or "exact" (default = "approximate").
...	Additional arguments to hdanova.

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 = X1 B1 + X2 B2 + ... + E = T1 P1' + T2 P2' + ... + 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: `hdanova`. 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)

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

# ASCA model with design-relevant compressed response
# using 5 partial least squares components
mod.pca <- asca(assessment ~ candy + assessor, data=candies, pls.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)

# Unbalanced data: compare native vs SS-type-aligned effects
drop_idx <- which(candies$candy == levels(candies$candy)[1] &
                  candies$assessor == levels(candies$assessor)[1])
candies.unbal <- candies[-drop_idx[seq_len(min(2, length(drop_idx)))], ]

mod.native <- asca(assessment ~ candy * assessor, data = candies.unbal)
mod.sstype <- asca(assessment ~ candy * assessor, data = candies.unbal,
                   respect_SStype = TRUE)

par.old <- par(mfrow = c(2,1), mar = c(4,4,2,1))
scoreplot(mod.native,
          main = "Top: native (regression-based LS)")
scoreplot(mod.sstype,
          main = "Bottom: respect_SStype = TRUE")
```

```
par(par.old)

perm.native <- permutation(mod.native, permute = 1000, respect_SStype = FALSE)
perm.sstype <- permutation(mod.sstype, permute = 1000, respect_SStype = TRUE)
summary(perm.native)
summary(perm.sstype)
```

---

asca_plots *ASCA Plot Methods*

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, ...)

rotationplot(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 chosing plot level (default = "all").
pch.scores	integer plotting symbol.
pch.projections	integer plotting symbol.
gr.col	integer vector of colours for groups.
projections	Include backprojections in score plot (default = TRUE).
spider	Draw lines between group centers and backprojections (default = FALSE).
ellipsoids	character "confidence" or "data" ellipsoids for balanced fixed effect models.
confidence	numeric vector of ellipsoid confidences, default = c(0.4, 0.68, 0.95).
xlim	numeric x limits.
ylim	numeric y limits.
xlab	character x label.
ylab	character y label.
legendpos	character position of legend.
main	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: hdanova. Extraction of results and plotting: asca_results, asca_plots, pcanova_results and pcanova_plots

Examples

```
# For end-to-end examples using scoreplot()/loadingplot(), see
# ?asca, ?apca, ?apls, ?limmpca and ?msca.
```

---

asca_results *ASCA Result Methods*

Description

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

Usage

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

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

## S3 method for class 'summary.hdanova'
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: hdanova. Extraction of results and plotting: asca_results, asca_plots, pcanova_results and pcanova_plots

Examples

```
# For end-to-end examples using summary(), scores(), loadings(), and
# projections(), see ?asca and ?apca.
```

`biplot.asca` *Biplot for ASCA models*

Description

Biplot for ASCA models

Usage

```
## S3 method for class 'asca'
biplot(
  x,
  factor = 1,
  comps = 1:2,
  xlim = NULL,
  ylim = NULL,
  col = "darkgray",
  expand = 1,
  labels,
  legendpos,
  ...
)
```

Arguments

`x`	`asca` object.
`factor`	Factor number or name.
`comps`	`integer` vector of selected components.
`xlim`	`numeric` vector of length 2 for x-axis limits of the loadings.
`ylim`	`numeric` vector of length 2 for y-axis limits of the loadings.
`col`	`vector` of colours for score axes and loading axes and points/texts.
`expand`	`numeric` expansion for the scores, defaulting to 1.
`labels`	optional. If `"names"`, row names are used as labels. If `"numbers"`, row numbers are used as labels. (Can also be a vector of labels.)
`legendpos`	`character` position of legend.
`...`	Additional arguments to `plot` and `scoreplot`.

Value

No return, only a plot.

Examples

```
# Load candies data
data(candies)

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

# Biplot
biplot(mod)

# Biplot with named loadings
biplot(mod, labels="names")
```

`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

`X`	Either a single `data.frame` to index or a `list` of matrices/data.frames
`block_inds`	Named `list` of indexes if `X` is a single `data.frame`, otherwise `NULL`.
`to.matrix`	`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: `hdanova`. 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 *Sensory assessment of candies.*

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 *Dummy-coding of a single vector*

Description

Flexible dummy-coding allowing for all R's built-in types of contrasts and optional dropping of a factor level to reduce rank defficiency 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: `hdanova`. 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)
```

`extract_estimates` *Extract estimates for a given factor combination*

Description

Extracts and sums the LS estimates for a given factor combination from an object of class hdanova. If `add_residuals` is `TRUE`, the residuals are added to the LS estimates. If `substract` is `TRUE`, the returned matrix is the data with chosen estimates subtracted.

Usage

```
extract_estimates(object, factors, subtract = FALSE, add_residuals = FALSE)
```

Arguments

`object`	`asca` object.
`factors`	`vector` of factor names or numbers.
`subtract`	`logical` subtract the estimates from the data (default = FALSE).
`add_residuals`	`logical` add residuals to the estimates (default = FALSE).

Value

A matrix of the extracted estimates.

See Also

Model fitting and related outputs: `hdanova`, `asca_results` and `predict.hdanova`.

Examples

```
# Load candies data
data(candies)

# Basic HDANOVA model with two factors and interaction
mod <- hdanova(assessment ~ candy * assessor, data=candies)

# Extract estimates for the interaction
inter <- extract_estimates(mod, c("candy:assessor"))

# Visualize the interaction effect
image(t(inter), main="Interaction effect", xlab="Attribute", ylab="Sample")
```

hdanova *High-Dimensional Analysis of Variance*

Description

This function provides a high-dimensional analysis of variance (HDANOVA) method which can be used alone or as part of a larger analysis, e.g., ASCA, APCA, LiMM-PCA, MSCA or PC-ANOVA. It can be called directly or through the convenience functions `asca`, `apca`, `limmpca`, `msca` and `pcanova`.

Usage

```
hdanova(
  formula,
  data,
  subset,
  weights,
  na.action,
  family,
  scale = FALSE,
  add_error = FALSE,
  aug_error = "denominator",
  pca.in = FALSE,
  pls.in = FALSE,
  contrasts = "contr.sum",
  unrestricted = FALSE,
  SStype = "II",
  respect_SStype = FALSE,
  REML = NULL,
  REML_ssq_method = c("exact_refit", "wald", "ls"),
  equal_baseline = FALSE,
  use_ED = FALSE
)
```

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.
`scale`	Scaling of the response matrix. Defaults to `FALSE` (no scaling). For alternatives, see Details.

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).
pca.in	Compress response before ASCA (number of components).
pls.in	Compress response before ASCA using PLS (number of components).
contrasts	Effect coding: "contr.sum" (default), "contr.weighted" (not for lme4 models), "contr.reference", "contr.treatment". Can also be specified shorthand without "contr.", e.g., "sum", or as a named vector for different coding per factor.
unrestricted	Use unrestricted ANOVA decomposition (default = FALSE).
SStype	Type of sum-of-squares: "I" = sequential, "II" (default) = last term, obeying marginality, "III" = last term, not obeying marginality.
respect_SStype	Logical; if FALSE (default), keep regression-based effect matrices in LS. If TRUE, expose SS-type-aligned effect matrices in LS while retaining regression matrices in LS_regression. This setting also propagates to permutation() when respect_SStype = NULL is used there.
REML	Parameter to mixlm: NULL (default) = sum-of-squares, TRUE = REML, FALSE = ML. If supplied without any r() terms in the formula, it is ignored with a warning and the model is fitted as fixed-effects.
REML_ssq_method	Method for REML mixed-model SSQ decomposition: "exact_refit" (default), "wald", or "ls". This is only used when REML is TRUE or FALSE for mixed models with r() terms.
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.
use_ED	Use "effective dimensions" for score rescaling in LiMM-PCA.

Details

Scaling of the response matrix can be done by setting the scale parameter. If scale=TRUE, each column is scaled by its standard deviation (autoscaling). A numeric value can be provided to scale the columns by specific quantities. If scale is a character string, the first element is interpreted as a factor name and the second element is interpreted as a factor level, whose samples the standard deviations are calculated from (reference group scaling).

SStype and respect_SStype (fixed and MoM mixed models): SStype controls how each effect's contribution is isolated from the others. Type I (sequential) assigns variance in the order terms appear in the formula; Type II (default) tests each term against all others that do not contain it, respecting marginality; Type III tests each term against the full model, ignoring marginality. For balanced designs the three types give identical results; differences arise in unbalanced or non-orthogonal designs.

By default (respect_SStype = FALSE) the effect matrices in LS are regression projections — each effect's columns are projected independently from the full model coefficient matrix. This is consistent with the legacy ASCA workflow. When respect_SStype = TRUE the LS matrices are rederived from QR contrasts that match the chosen SStype, so the Frobenius norm of LS[[eff]]

equals the corresponding ANOVA sum of squares. The regression matrices are still available in `LS_regression`. For balanced, fully-crossed designs the two sets are numerically identical; for unbalanced designs they will differ, particularly for Type II and III.

REML SSQ strategies (mixed models with `r()` terms): When `REML = TRUE` or `REML = FALSE`, the model is fitted with lme4 and the classical ANOVA table is replaced by one of three decomposition strategies controlled by `REML_ssq_method`:

- `"exact_refit"` (default): For each effect a reduced model is re-fitted (REML or ML as specified) and the SSQ is the difference in log-likelihoods scaled to the sum-of-squares metric. Concretely, if ℓ_f and ℓ_r are the log-likelihoods of the full and reduced models, the statistic is

$$\mathrm{SSQ} = 2(\ell_f - \ell_r).$$

  This is the most principled approach but requires one additional model fit per effect, making it slower for large models.

- `"wald"`: Uses the Wald chi-square statistic from the full model divided by the residual variance to obtain an approximate SSQ. For a fixed effect with coefficient vector $\hat{\boldsymbol{\beta}}_j$ and covariance $\mathbf{V}_j = \mathrm{Var}(\hat{\boldsymbol{\beta}}_j)$,

$$\mathrm{SSQ} \approx \hat{\boldsymbol{\beta}}_j^\top \mathbf{V}_j^{-1} \hat{\boldsymbol{\beta}}_j \cdot \hat{\sigma}^2.$$

  No additional model fits are required; fast but less accurate for small samples or near-singular random effects.

- `"ls"`: Projects the REML/ML coefficient estimates through the fixed-effect design matrix to recover a least-squares-style SSQ,

$$\mathrm{SSQ} = \|\mathbf{X}_j \hat{\boldsymbol{\beta}}_j\|_F^2,$$

  where $\mathbf{X}_j$ contains the columns of the design matrix for effect j. Computationally cheap and numerically stable but ignores the mixed-model covariance structure; best treated as an approximation.

Permutation statistics vs. fitted-model SSQ: Permutation testing (`permutation`) always uses QR-based projection statistics computed on the fixed-effect design matrix, regardless of `REML_ssq_method`. When `respect_SStype = FALSE` (default) the permutation statistic is a regression-projection norm; when `respect_SStype = TRUE` it is an SS-type-aligned QR contrast norm. Neither is identical to the fitted-model REML SSQ values reported in `object$ssq`, because REML/ML decompositions account for the random-effect covariance structure whereas permutation statistics do not. The p-values are still valid under their respective null hypotheses; the SSQ values should not be compared across the two sources.

Value

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

Examples

```
# Load candies data
data(candies)
```

```
# Basic HDANOVA model with two factors
mod <- hdanova(assessment ~ candy + assessor, data=candies)
summary(mod)
```

---

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",
  permute = FALSE,
  perm.type = c("approximate", "exact"),
  SStype = "III",
  ...
)
```

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".
permute	Number of permutations to perform (default = 1000).

perm.type	Type of permutation to perform, either "approximate" or "exact" (default = "approximate").
SStype	Type of sum-of-squares: "I" = sequential, "II" = last term, obeying marginality, "III" (default) = last term, not obeying marginality.
...	Additional arguments to hdanova.

Details

The Sum of Squares for the model is dependent on the SStype of the model. For SStype = "I" and SStype = "II" the SSQ is based on LLR (possibly inflating large contributions), while it is directly estimated from the model for SStype = "III". SStype = "III" is the default for LiMM-PCA and should be combined with sum coding. Sum of Squares for the random effects are based on the variance components.

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: hdanova. Extraction of results and plotting: asca_results, asca_plots, pcanova_results and pcanova_plots Prediction on new data: predict_asca_family.

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)
```

`logLik.hdanova` *Log-Likelihood for HDANOVA Objects*

Description

Extract log-likelihood from fitted HDANOVA objects. For fixed-effects and MoM ('r()' with `REML = NULL`) models, the log-likelihood is computed from the residual sum of squares. For REML/ML mixed models ('r()' with `REML = TRUE` or `FALSE`), the log-likelihood is extracted directly from the stored lme4 model objects.

Usage

```
## S3 method for class 'hdanova'
logLik(object, ...)
```

Arguments

`object`	A fitted `hdanova` object.
`...`	Reserved for generic compatibility.

Details

The log-likelihood is computed as follows:

Fixed-effects and MoM models: The multivariate Gaussian log-likelihood is

$$\ell_j = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma_j^2) - \frac{1}{2\sigma_j^2}\sum_i e_{ij}^2$$

where e_{ij} is the residual for observation i and response j, and $\sigma_j^2 = SSE_j/n$ is the ML variance estimate for response column j. The total log-likelihood is $\ell = \sum_j \ell_j$.

REML/ML mixed models: The log-likelihood is the sum of the individual response-specific log-likelihoods from the fitted lme4 models,

$$\ell = \sum_j \ell_j$$

where ℓ_j is the log-likelihood of the lme4::lmerMod fit for response j.

Value

An object of class `logLik` with the computed log-likelihood value, degrees of freedom ('df'), and the number of observations ('nobs').

See Also

Model fitting: `hdanova`. Information criteria: `AIC` and `BIC`.

Examples

```
data(candies)
mod <- hdanova(assessment ~ candy + assessor, data = candies)
ll <- logLik(mod)
print(ll)

## Not run:
# For mixed models:
mod_reml <- hdanova(assessment ~ candy + r(assessor), data = candies, REML = TRUE)
ll_reml <- logLik(mod_reml)
print(ll_reml)

## End(Not run)
```

`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

`formula`	The `asca` object.
`...`	Not implemented
`object`	The `asca` object.

Value

A `data.frame` or `matrix` object.

See Also

Core model constructors: `asca`, `apca`, `apls`, `limmpca`, `msca` and `pcanova`.

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",
  permute = FALSE,
  perm.type = c("approximate", "exact"),
  ...
)
```

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".
`permute`	Number of permutations to perform (default = 1000).
`perm.type`	Type of permutation to perform, either "approximate" or "exact" (default = "approximate").
`...`	Additional arguments to hdanova.

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: hdanova. 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)
```

pcanova *Principal Components Analysis of Variance Simultaneous Component Analysis - PC-ANOVA*

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 hdanova 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: hdanova. 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.glht(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: hdanova. Extraction of results and plotting: asca_results, asca_plots, pcanova_results and pcanova_plots

Examples

```
# For end-to-end examples using scoreplot()/loadingplot() on PC-ANOVA
# objects, see ?pcanova.
```

---

pcanova_results *PC-ANOVA Result Methods*

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: hdanova. Extraction of results and plotting: asca_results, asca_plots, pcanova_results and pcanova_plots

Examples

```
# For examples using print()/summary() for PC-ANOVA objects, see
# ?pcanova.
```

---

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.

See Also

Related wrappers and model families: prc, asca, apca, apls, limmpca, msca and pcanova.

Examples

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

---

permutation *Permutation for SS-Type-Aligned HDANOVA*

Description

Permutation testing for hdanova() objects using SS-type-aligned QR effect matrices. This function is intended as an opt-in alternative to the legacy regression-based permutation() workflow.

Usage

```
permutation(
  object,
  permute = 1000,
  perm.type = c("approximate", "exact"),
  respect_SStype = NULL,
  unique.digits = 12,
  unique.frac = 0.95,
  exhaustive.warn = TRUE
)
```

Arguments

object	A hdanova object.
permute	Number of permutations to perform (default = 1000).
perm.type	Type of permutation to perform, either "approximate" or "exact" (default = "approximate").

`respect_SStype` Logical or `NULL`. If `NULL` (default), use the `hdanova` object setting (`object$more$respect_SStype`). If `FALSE`, follow the legacy regression-based permutation logic. If `TRUE`, use SS-type-aligned QR contrasts. For REML/ML mixed models, this may yield permutation statistics that differ from fitted-model REML SSQ decompositions selected by `REML_ssq_method`.

`unique.digits` Number of digits used when rounding permutation SSQ values before checking uniqueness (default = 12). Set to `NULL` to disable this warning.

`unique.frac` Minimum fraction of unique rounded SSQ values required to avoid warning (default = 0.95). Set to `NULL` to disable this warning.

`exhaustive.warn` Logical; if `TRUE` (default), warn when exact permutation uses exhaustive enumeration with fewer permutations than requested.

Details

The permutation statistics are computed from SS-type-aligned QR reduced/full model contrasts rather than the legacy regression-based LS matrices. Fixed models and mixed MoM models are supported. Approximate permutation uses a relaxed global shuffle of observations; exact permutation uses permissible block-restricted shuffles. For REML/ML mixed models with `respect_SStype = TRUE`, a warning is issued to highlight that permutation statistics and fitted-model REML SSQ decompositions are based on different computational definitions.

Value

An updated `hdanova` object with `permute` results.

`pls` *Partial Least Squares (PLS) for HDANOVA*

Description

This function performs Partial Least Squares (PLS) on a `hdanova`.

Usage

```
pls(object, ...)

## Default S3 method:
pls(object, ...)
```

Arguments

`object` A `hdanova` object.

`...` Additional arguments (not used).

Details

For residuals, PCA is performed instead of PLS as there is no natural response.

Value

An updated hdanova object with PLS results.

See Also

Main wrapper: apls. Related decomposition: sca. Plotting and summaries: asca_plots and asca_results.

Examples

```
# Load candies data
data(candies)

# Basic HDANOVA model with two factors
mod <- hdanova(assessment ~ candy + assessor, data=candies)
mod <- pls(mod)
scoreplot(mod)
```

---

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: hdanova. 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)
```

---

predict.hdanova *Predict for HDANOVA Objects*

Description

Reconstructs an HDANOVA-style object on newdata without refitting by reusing stored coefficients and projection objects from the fitted model. This implementation supports fixed-effects, mixed MoM (r() with REML = NULL), and REML/ML mixed workflows (r() with REML = TRUE or FALSE).

Usage

```
## S3 method for class 'hdanova'
predict(object, newdata, ...)
```

Arguments

object	A fitted hdanova object.
newdata	A data frame containing the variables used in the original model formula.
...	Reserved for generic compatibility; runtime overrides are not supported.

Value

An hdanova-family object computed on newdata.

See Also

ASCA-family prediction wrappers: predict_asca_family. Model constructors: hdanova, asca, apca, apls, limmpca, msca and pcanova.

Examples

```
data(candies)
# Train/test split (every third sample to test)
test_idx  <- seq(3, nrow(candies), by = 3)
train_idx <- setdiff(1:nrow(candies),test_idx)
candies_train <- candies[train_idx, ]
candies_test  <- candies[test_idx, ]

# Fixed-effects model prediction
mod <- hdanova(assessment ~ candy + assessor, data = candies_train)
pred <- predict(mod, newdata = candies_test)

var_idx <- seq_len(ncol(mod$LS$candy))
old.par <- par(mfrow = c(1,2), mar = c(4,4,2,1), mgp = c(2,0.7,0))
image(x = var_idx, y = seq_along(train_idx), z = t(mod$LS$candy),
      xaxt = "n", yaxt = "n", main = "Original candy LS",
      xlab = "Variable index", ylab = "Train sample index")
axis(1, at = var_idx, labels = var_idx)
axis(2, at = seq_along(train_idx), labels = train_idx)
image(x = var_idx, y = seq_along(test_idx), z = t(pred$LS$candy),
      xaxt = "n", yaxt = "n", main = "Predicted candy LS",
      xlab = "Variable index", ylab = "Test sample index")
axis(1, at = var_idx, labels = var_idx)
axis(2, at = seq_along(test_idx), labels = test_idx)
par(old.par)

# Mixed MoM model prediction (r() with REML = NULL)
mod_mom <- hdanova(assessment ~ candy + r(assessor), data = candies_train)
pred_mom <- predict(mod_mom, newdata = candies_test)
cat("Mixed MoM model prediction successful.\n")
cat("SSQ names:", paste(names(pred_mom$ssq), collapse = "|"), "\n")
cat("dfDenom:", paste(pred_mom$dfDenom, collapse = "|"), "\n")

# REML mixed model prediction (r() with REML = TRUE)
mod_reml <- hdanova(assessment ~ candy + r(assessor), data = candies_train, REML = TRUE)
pred_reml <- predict(mod_reml, newdata = candies_test)
cat("REML mixed model prediction successful.\n")
cat("SSQ names:", paste(names(pred_reml$ssq), collapse = "|"), "\n")
cat("dfDenom:", paste(pred_reml$dfDenom, collapse = "|"), "\n")
```

---

predict_asca_family *Predict Methods for ASCA-family Objects*

Description

Reconstructs the HDANOVA base on `newdata` via `predict.hdanova()` and then applies the class-specific decomposition step: `sca()` for ASCA/APCA/MSCA and `pls()` for APLS. By default, decomposition is done by projection onto training component spaces. A refit mode is also available.

Usage

```
## S3 method for class 'asca'
predict(object, newdata, decomposition = c("project", "refit"), ...)

## S3 method for class 'apca'
predict(object, newdata, decomposition = c("project", "refit"), ...)

## S3 method for class 'msca'
predict(object, newdata, decomposition = c("project", "refit"), ...)

## S3 method for class 'apls'
predict(object, newdata, decomposition = c("project", "refit"), ...)

## S3 method for class 'limmpca'
predict(object, newdata, decomposition = c("project", "refit"), ...)
```

Arguments

`object`	A fitted `asca`, `apca`, `msca`, `apls`, or `limmpca` object.
`newdata`	A data frame containing variables from the original model formula.
`decomposition`	Decomposition mode: `"project"` (default) projects onto training component spaces; `"refit"` recomputes decomposition on predicted LS matrices.
`...`	Reserved for compatibility; forwarded to `predict.hdanova()`.

Value

A predicted object of the same high-level class as `object`.

See Also

Base prediction engine: `predict.hdanova`. Related model constructors: `asca`, `apca`, `apls`, `msca` and `limmpca`.

Examples

```
data(candies)
test_idx  <- seq(3, nrow(candies), by = 3)
train_idx <- setdiff(seq_len(nrow(candies)), test_idx)
candies_train <- candies[train_idx, ]
candies_test  <- candies[test_idx, ]

mod_asca <- asca(assessment ~ candy * assessor, data = candies_train)
pred_asca <- predict(mod_asca, newdata = candies_test)
scoreplot(mod_asca, factor="candy", legend=TRUE)
with(pred_asca$projected, points(candy[,1], candy[,2], pch="x", cex=0.8,
                                 col=as.numeric(pred_asca$model.frame$candy)))

pred_asca_refit <- predict(mod_asca, newdata = candies_test, decomposition = "refit")
```

```
mod_apca <- apca(assessment ~ candy + assessor, data = candies_train)
pred_apca <- predict(mod_apca, newdata = candies_test)

mod_msca <- msca(assessment ~ candy, data = candies_train)
pred_msca <- predict(mod_msca, newdata = candies_test)

mod_apls <- apls(assessment ~ candy + assessor, data = candies_train)
pred_apls <- predict(mod_apls, newdata = candies_test)

mod_limmpca <- limmpca(assessment ~ candy + r(assessor),
                       data = candies_train, pca.in = 3)
pred_limmpca <- predict(mod_limmpca, newdata = candies_test)
```

`rotation` *Rotation test for HDANOVA*

Description

Rotation testing for HDANOVA. This function performs random orthogonal rotations of exchangeable residual units for each approved effect and adds the resulting null distributions to the `hdanova` object.

Usage

```
rotation(
  object,
  rotate = 1000,
  unique.digits = 12,
  unique.frac = 0.95,
  block.type = c("denominator", "global")
)
```

Arguments

`object`	A `hdanova` object.
`rotate`	Number of random rotations to perform (default = 1000).
`unique.digits`	Number of digits used when rounding rotation SSQ values before checking uniqueness (default = 12). Set to `NULL` to disable this warning.
`unique.frac`	Minimum fraction of unique rounded SSQ values required to avoid warning (default = 0.95). Set to `NULL` to disable this warning.
`block.type`	Rotation blocking strategy. `"denominator"` (default) rotates within denominator-compatible exchangeable blocks. `"global"` rotates across all observations.

Value

An updated `hdanova` object with rotation-test results stored in `object$permute` for compatibility with existing summary and plotting tools.

See Also

Base fitting: hdanova. Permutation alternative: permutation. Plot helper: rotationplot.

Examples

```
# Load candies data
data(candies)

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

# Rotation test
modRot <- rotation(mod)
summary(modRot)

# Plot null distribution for "candy" effect
rotationplot(modRot, factor="candy")
```

---

sca *Simultaneous Component Analysis*

Description

This function performs Simultaneous Component Analysis (SCA) on a hdanova object.

Usage

```
sca(object)
```

Arguments

object	A hdanova object.

Value

An updated hdanova object with SCA results.

See Also

Model constructors and wrappers: hdanova, asca, apca, limmpca and msca. Plotting and summaries: asca_plots and asca_results.

Examples

```
# Load candies data
data(candies)

# Basic HDANOVA model with two factors
mod <- hdanova(assessment ~ candy + assessor, data=candies)
mod <- sca(mod)
scoreplot(mod)
```

---

signflip *Flip signs of a component/factor combination in a SCA/PCA object*

Description

This function flips the sign of a selected component in a selected factor of an asca object. This affects both scores, loadings and projected data.

Usage

```
signflip(object, factor, comp)
```

Arguments

object	asca object.
factor	integer/character for selecting a model factor.
comp	integer for selected component.

Value

An asca object with the sign of the selected component flipped.

Examples

```
# Load candies data
data(candies)

# Basic HDANOVA model with two factors
mod <- hdanova(assessment ~ candy + assessor, data=candies)
mod <- sca(mod)
old.par <- par(mfrow=c(1,2), mar=c(4,4,1,1))
scoreplot(mod, factor="candy")
loadingplot(mod, factor="candy")
par(old.par)

# Flip the sign of the first component of the candy factor
mod <- signflip(mod, factor="candy", comp=1)
old.par <- par(mfrow=c(1,2), mar=c(4,4,1,1))
```

```
scoreplot(mod, factor="candy")
loadingplot(mod, factor="candy")
par(old.par)
```

`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

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

Value

Nothing

Examples

```
data("caldana")
mod.comb <- asca(compounds ~ time + comb(light + time:light), 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, *31, 32*
AIC, *24*
apca, 3, *3*, *5*, *7*, *11, 12*, *14*, *17*, *19*, *23*, *25*, *27, 28*, *30–32*, *35*, *37*, *39*
apls, 4, *25*, *32*, *34, 35*, *37*
asca, *3*, *5*, 6, *7*, *9–12*, *14*, *17*, *19*, *23*, *25*, *27, 28*, *30–32*, *35*, *37*, *39*
asca_plots, *3*, *5*, *7*, 9, *11, 12*, *14*, *17*, *21*, *23*, *27, 28*, *30, 31*, *34, 35*, *39*
asca_results, *3*, *5*, *7*, 11, *11, 12*, *14*, *17, 18*, *21*, *23*, *27, 28*, *30, 31*, *34, 35*, *39*

BIC, *24*
biplot.asca, 13
block.data.frame, 14

caldana, 15
candies, 16

data.frame, *17*
dummycode, 16

extended.model.frame, 17
extract_estimates, 18

hdanova, *3–5*, *7*, *11, 12*, *14*, *17, 18*, 19, *23, 24*, *26–28*, *30, 31*, *35*, *39*

limmpca, *3*, *5*, *7*, *11, 12*, *14*, *17*, *19*, 22, *23*, *25*, *27, 28*, *30–32*, *35*, *37*, *39*
loadingplot.asca *(*asca_plots*)*, 9
loadingplot.pcanova *(*pcanova_plots*)*, 29
loadings.asca *(*asca_results*)*, 11
logLik.hdanova, 24

model.frame, *17*
model.frame.asca, 25
model.matrix.asca *(*model.frame.asca*)*, 25
msca, *3*, *5*, *7*, *11, 12*, *14*, *17*, *19*, *23*, *25*, 26, *27, 28*, *30–32*, *35*, *37*, *39*

pcanova, *3*, *5*, *7*, *11, 12*, *14*, *17*, *19*, *23*, *25*, *27*, 28, *28–32*, *35*
pcanova_plots, *3*, *5*, *7*, *11, 12*, *14*, *17*, *23*, *27, 28*, 29, *30, 31*, *35*
pcanova_results, *3*, *5*, *7*, *11, 12*, *14*, *17*, *23*, *27, 28*, 30, *30, 31*, *35*
permanova, *3*, *5*, *7*, *11, 12*, *14*, *17*, *23*, *27, 28*, *30*, 31, *31*, *35*
permutation, *21*, 32, *39*
permutationplot *(*asca_plots*)*, 9
pls, 33
prc, *3*, *5*, *7*, *11, 12*, *14*, *17*, *23*, *27, 28*, *30–32*, 34, *34, 35*
predict.apca *(*predict_asca_family*)*, 36
predict.apls *(*predict_asca_family*)*, 36
predict.asca *(*predict_asca_family*)*, 36
predict.hdanova, *18*, 35, *37*
predict.limmpca *(*predict_asca_family*)*, 36
predict.msca *(*predict_asca_family*)*, 36
predict_asca_family, *23*, *35*, 36
print.asca *(*asca_results*)*, 11
print.hdanova *(*asca_results*)*, 11
print.pcanova *(*pcanova_results*)*, 30
print.summary.asca *(*asca_results*)*, 11
print.summary.hdanova *(*asca_results*)*, 11
print.summary.pcanova *(*pcanova_results*)*, 30
projections *(*asca_results*)*, 11
projections.pcanova *(*pcanova_results*)*, 30

rotation, 38
rotationplot, *39*
rotationplot *(*asca_plots*)*, 9

sca, *34*, 39
scoreplot.asca *(*asca_plots*)*, 9
scoreplot.pcanova *(*pcanova_plots*)*, 29
scores.asca *(*asca_results*)*, 11

signflip, 40
summary.asca (asca_results), 11
summary.hdanova (asca_results), 11
summary.pcanova (pcanova_results), 30

timeplot, 41

update_without_factor, 42