Package ‘nzilbb.vowels’

July 22, 2025

Type Package

Title Vowel Covariation Tools

Version 0.4.1

Description Tools to support research on vowel covariation. Methods are provided to support Principal Component Analysis workflows (as in Brand et al. (2021) <doi:10.1016/j.wocn.2021.101096> and Wilson Black et al. (2023) <doi:10.1515/lingvan-2022-0086>).

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 4.1), patchwork

RoxygenNote 7.3.2.9000

Suggests knitr, rmarkdown, testthat (>= 3.0.0), vdiffr

Config/testthat/edition 3

Imports dplyr, ggplot2, magrittr, rlang, rstudioapi, tibble, tidyr, forcats, glue, purrr, tidyselect, rsample, stringr, ggrepel, gghalves, smacof, Rdpack, lifecycle, vegan

RdMacros Rdpack

URL `https://nzilbb.github.io/nzilbb_vowels/`, `https://github.com/nzilbb/nzilbb_vowels`

BugReports `https://github.com/nzilbb/nzilbb_vowels/issues`

Config/Needs/website rmarkdown, bookdown, tidyverse, ggcorrplot, mgcv, itsadug, broom, gganimate, grateful, gifski

NeedsCompilation no

Author Joshua Wilson Black [aut, cre, cph] (ORCID: <`https://orcid.org/0000-0002-8272-5763`>), James Brand [aut] (ORCID: <`https://orcid.org/0000-0002-2853-9169`>)

Maintainer Joshua Wilson Black <`joshua@wilsonblack.nz`>

Repository CRAN

Date/Publication 2025-05-06 04:10:02 UTC

Contents

`correlation_test` *Permutation test of pairwise correlations*

Description

Permute data a given number (n) of times, collecting pairwise correlations and testing them for significance. See `plot_correlation_magnitudes()` and `plot_correlation_counts()` for plotting functions which take the output of this function.

Usage

```
correlation_test(pca_data, n = 100, cor.method = "pearson")
```

Arguments

`pca_data`	dataframe or matrix containing only continuous variables. (as accepted by the `prcomp` function.)
`n`	the number of times (integer) to permute that data. **Warning:** high values will take a long time to compute. Default: 100.
`cor.method`	method to use for correlations (default = "pearson"). Alternative is "spearman" (see `?cor.test`).

Value

object of class `correlation_test`, with attributes:

- `$permuted_correlations` A tibble of length `n` of pairs from the original data, their correlations, and the significance of each correlation (as p-values).
- `$actual_correlations` the correlations of each pair of variables in the original data and their significance (as p-values).
- `$iterations` the number of permutations carried out.
- `$cor_method` the form of correlation used.

Examples

```
# get a small sample of random intercepts.
pca_data <- onze_intercepts |>
  dplyr::select(-speaker) |>
  dplyr::slice_sample(n=10)

# apply correlation test with 5 permutations.
# actual use requires at least 100.
cor_test <- correlation_test(pca_data, n = 5, cor.method = 'pearson')
# Return summary of significant correlations
summary(cor_test)

# use spearman correlation instead.
cor_test_spear <- correlation_test(pca_data, n = 10, cor.method = 'spearman')
```

`lobanov_2` *Apply Lobanov 2.0 normalisation*

Description

`lobanov_2()` takes a data frame where the first four columns are:

1. speaker identifiers,
2. vowel identifiers,
3. first formant values in Hertz,
4. second formant values in Hertz.

It returns a dataframe with two additional columns, `F1_lob2` and `F2_lob2`, containing normalised formant values.

Usage

```
lobanov_2(vowel_data)
```

Arguments

`vowel_data`	a dataframe whose first four columns are speaker ids, vowel ids, F1 values, and F2 values.

Details

This functions applies Lobanov 2.0 normalisation presented in Brand et al. (2021). This variant of Lobanov normalisation is designed to work for datasets whether the vowel types have different token counts from one another. The Lobanov 2.0 value for a vowel is given by

$$F_{lobanov2.0_i} = \frac{F_{raw_i} - \mu(\mu_{vowel_1}, \ldots, \mu_{vowel_n})}{\sigma(\mu_{vowel_1}, \ldots, \mu_{vowel_n})}$$

where, for ease of notation, we assume all values are from a single speaker. We signify the n vowel types as vowel_1, …, vowel_2, while i indicates the formant number. We implement the function for F1 and F2.

Value

a dataframe matching the input dataframe with additional columns `F1_lob2` and `F2_lob2`, containing the lobanov normalised F1 and F2 values respectively.

References

Brand, James, Jen Hay, Lynn Clark, Kevin Watson & Márton Sóskuthy (2021): Systematic co-variation of monophthongs across speakers of New Zealand English. Journal of Phonetics. Elsevier. 88. 101096. doi:10.1016/j.wocn.2021.101096

Examples

```
normed_vowels <- lobanov_2(onze_vowels)
head(normed_vowels)
```

`mds_test` *Test optimal number of MDS dimensions.*

Description

[Experimental] Generate bootstrapped confidence intervals and permutation based null distribution for MDS analysis. Output shows how much stress is reduced by adding an additional dimension to the MDS analysis of `dissimilarity_matrix`, and bootstrapped iterations of `dissimilarity_matrix`, compared with the stress reduction expected from a matrix with no meaningful structure. This function is inspired by `pca_test()`, but is less connected with statistical literature than that function. We currently reject additional dimensions is they reduce less stress than we would expect by chance. That is, when the distribution from the boostrapped analyses sits notably lower than the permuted distribution when plotted by `plot_mds_test()`

Usage

```
mds_test(
  dissimilarity_matrix,
  n_boots = 50,
  n_perms = 50,
  test_dimensions = 5,
  principal = TRUE,
  mds_type = "ordinal",
  spline_degree = 2,
  spline_int_knots = 2,
  ...
)
```

Arguments

`dissimilarity_matrix`	Square matrix of dissimilarity scores.
`n_boots`	Number of bootstrapping iterations (default: 25).
`n_perms`	Number of permutations (default: 25).
`test_dimensions`	Number of MDS dimensions to test for stress reduction (default: 5).
`principal`	Whether to apply principal axis transform to MDS (default: TRUE)
`mds_type`	What kind of MDS to apply, see `smacof::smacofSym()` (default: 'ordinal')
`spline_degree`	How many spline degrees when `type` is 'mspline' (default: 2)
`spline_int_knots`	How many internal knots when `type` is 'mspline' (default: 2)
`...`	Arguments passed to `smacof::smacofSym()`

Value

object of class `mds_test_results`, containing:

- `$stress_reduction` a tibble containing
- `$n_boots` Number of bootstrapping iterations.
- `$n_perms` Number of permutation iterations
- `$mds_type` Type of MDS analysis (`type` argument passed to `smacof::smacofSym()`)
- `$principal` Whether principal axis transformation is applied (passed to `smacof::smacofSym()`)

Examples

```
# Apply interval MDS to `sim_matrix`, with 5 permutations and bootstraps
# testing up to 3 dimensions. In real usage, increase `n_boots` and `n_perms`
# to at least 50.
mds_test(
 smacof::sim2diss(sim_matrix, method="reverse"),
 n_boots = 5,
 n_perms = 5,
 test_dimensions = 3,
 mds_type = 'interval'
)
```

`onze_intercepts` *Speaker random intercepts from GAMMs for 100 ONZE speakers*

Description

A dataset containing the speaker intercepts extracted from GAMM models fit in Brand et al. (2021).

Usage

```
onze_intercepts
```

Format

A data frame with 100 rows and 21 variables:

speaker Anonymised speaker code (character).

F1_DRESS Speaker intercept from GAMM model of DRESS F1.

F2_DRESS Speaker intercept from GAMM model of DRESS F2.

F1_FLEECE Speaker intercept from GAMM model of FLEECE F1.

F2_FLEECE Speaker intercept from GAMM model of FLEECE F2.

F1_GOOSE Speaker intercept from GAMM model of GOOSE F1.

F2_GOOSE Speaker intercept from GAMM model of GOOSE F2.

F1_KIT Speaker intercept from GAMM model of KIT F1.

F2_KIT Speaker intercept from GAMM model of KIT F2.

F1_LOT Speaker intercept from GAMM model of LOT F1.

F2_LOT Speaker intercept from GAMM model of LOT F2.

F1_NURSE Speaker intercept from GAMM model of NURSE F1.

F2_NURSE Speaker intercept from GAMM model of NURSE F2.

F1_START Speaker intercept from GAMM model of START F1.

F2_START Speaker intercept from GAMM model of START F2.

F1_STRUT Speaker intercept from GAMM model of STRUT F1.

F2_STRUT Speaker intercept from GAMM model of STRUT F2.

F1_THOUGHT Speaker intercept from GAMM model of THOUGHT F1.

F2_THOUGHT Speaker intercept from GAMM model of THOUGHT F2.

F1_TRAP Speaker intercept from GAMM model of TRAP F1.

F2_TRAP Speaker intercept from GAMM model of TRAP F2.

Source

`https://osf.io/q4j29/`

References

Brand, James, Jen Hay, Lynn Clark, Kevin Watson & Márton Sóskuthy (2021): Systematic co-variation of monophthongs across speakers of New Zealand English. Journal of Phonetics. Elsevier. 88. 101096. doi:10.1016/j.wocn.2021.101096

`onze_intercepts_full` *Speaker random intercepts for 418 ONZE speakers*

Description

A dataset containing the speaker intercepts extracted from GAMM models fit in Brand et al. (2021).

Usage

```
onze_intercepts_full
```

Format

A data frame with 481 rows and 21 variables:

speaker Anonymised speaker code.

F1_DRESS Speaker intercept from GAMM model of DRESS F1.

F2_DRESS Speaker intercept from GAMM model of DRESS F2.

F1_FLEECE Speaker intercept from GAMM model of FLEECE F1.

F2_FLEECE Speaker intercept from GAMM model of FLEECE F2.

F1_GOOSE Speaker intercept from GAMM model of GOOSE F1.

F2_GOOSE Speaker intercept from GAMM model of GOOSE F2.

F1_KIT Speaker intercept from GAMM model of KIT F1.

F2_KIT Speaker intercept from GAMM model of KIT F2.

F1_LOT Speaker intercept from GAMM model of LOT F1.

F2_LOT Speaker intercept from GAMM model of LOT F2.

F1_NURSE Speaker intercept from GAMM model of NURSE F1.

F2_NURSE Speaker intercept from GAMM model of NURSE F2.

F1_START Speaker intercept from GAMM model of START F1.

F2_START Speaker intercept from GAMM model of START F2.

F1_STRUT Speaker intercept from GAMM model of STRUT F1.

F2_STRUT Speaker intercept from GAMM model of STRUT F2.

F1_THOUGHT Speaker intercept from GAMM model of THOUGHT F1.

F2_THOUGHT Speaker intercept from GAMM model of THOUGHT F2.

F1_TRAP Speaker intercept from GAMM model of TRAP F1.

F2_TRAP Speaker intercept from GAMM model of TRAP F2.

Source

`https://osf.io/q4j29/`

References

Brand, James, Jen Hay, Lynn Clark, Kevin Watson & Márton Sóskuthy (2021): Systematic co-variation of monophthongs across speakers of New Zealand English. Journal of Phonetics. Elsevier. 88. 101096. doi:10.1016/j.wocn.2021.101096

`onze_vowels` *Monophthong data for random sample of speakers from the ONZE corpus*

Description

A dataset containing the the first and second formants, speech rate, gender, and year of birth for 100 random speakers from the ONZE corpus. 50 speakers are sampled with birth years before 1900 and 50 sampled with birth years on or after 1900 to ensure a full span of the time period. Data is present for the following NZE monophthongs, represented by Wells lexical sets: DRESS, FLEECE, GOOSE, KIT, LOT, NURSE, START, STRUT, THOUGHT, TRAP. Data for FOOT is excluded due to low token counts.

Usage

```
onze_vowels
```

Format

A dataframe with 101572 rows and 8 variables:

speaker Anonymised speaker code (factor).

vowel Variable with Wells lexical sets for 10 NZE monophthongs. Levels: DRESS, FLEECE, GOOSE, KIT, LOT, NURSE, START, STRUT, THOUGHT, TRAP (factor).

F1_50 First formant, extracted from vowel mid-point using LaBB-CAT interface with Praat.

F2_50 Second formant, extracted from vowel mid-point using LaBB-CAT interface with Praat.

speech_rate Average speaker speech rate for whole recording.

gender Gender of speaker, two levels: "M", "F" (factor).

yob Year of birth of speaker.

word Anonymised word code (factor).

Details

This dataset is derived from the data made available in the supplementary materials of Brand et al. (2021).

Source

`https://osf.io/q4j29/`

References

Brand, James, Jen Hay, Lynn Clark, Kevin Watson & Márton Sóskuthy (2021): Systematic co-variation of monophthongs across speakers of New Zealand English. Journal of Phonetics. Elsevier. 88. 101096. doi:10.1016/j.wocn.2021.101096

`onze_vowels_full` *Monophthong data for speakers from the ONZE corpus*

Description

A dataset containing the the first and second formants, speech rate, gender, and year of birth for 481 speakers from the ONZE corpus. 50 speakers are sampled with birth years before 1900 and 50 sampled with birth years on or after 1900 to ensure a full span of the time period. Data is present for the following NZE monophthongs, represented by Wells lexical sets: DRESS, FLEECE, GOOSE, KIT, LOT, NURSE, START, STRUT, THOUGHT, TRAP. Data for FOOT is excluded due to low token counts.

Usage

```
onze_vowels_full
```

Format

A data frame with 414679 rows and 8 variables:

- **speaker** Anonymised speaker code (factor).
- **vowel** Variable with Wells lexical sets for 10 NZE monophthongs. Levels: DRESS, FLEECE, GOOSE, KIT, LOT, NURSE, START, STRUT, THOUGHT, TRAP (factor).
- **F1_50** First formant, extracted from vowel mid-point using LaBB-CAT interface with Praat.
- **F2_50** Second formant, extracted from vowel mid-point using LaBB-CAT interface with Praat.
- **speech_rate** Average speaker speech rate for whole recording.
- **gender** Gender of speaker, two levels: "M", "F" (factor).
- **yob** Year of birth of speaker.
- **word** Anonymised word code (factor).

Details

This dataset is derived from the data made available in the supplementary materials of Brand et al. (2021).

Source

`https://osf.io/q4j29/`

References

Brand, James, Jen Hay, Lynn Clark, Kevin Watson & Márton Sóskuthy (2021): Systematic co-variation of monophthongs across speakers of New Zealand English. Journal of Phonetics. Elsevier. 88. 101096. doi:10.1016/j.wocn.2021.101096

`pca_contrib_plot` *PCA contribution plots*

Description

Plot the contribution of each variable in a data set to a given Principal Component (PC). Variables are arranged by ascending contribution to the PC, where contribution is the squared loading for the variable expressed as a percentage. These plots match those given in supplementary material for Brand et al. (2021).

Usage

```
pca_contrib_plot(pca_object, pc_no = 1, cutoff = 50)
```

Arguments

`pca_object`	a pca object generated by `prcomp` or `princomp`.
`pc_no`	the PC to be visualised. Default value is 1.
`cutoff`	the cutoff value for interpretation of the PC. Determines what total percentage contribution we want from the variables we select for interpretation. The default of 50 means that we pick the variables with the highest contribution to the PC until we have accounted for 50% of the total contributions to the PC. Can be set to `NULL` in which case, no cutoff value is plotted.

Details

As with the other plotting functions in this package, the result is a `ggplot2` plot. It can be modified using `ggplot2` functions (see, e.g., `plot_correlation_magnitudes()`.

Value

`ggplot` object.

References

Brand, James, Jen Hay, Lynn Clark, Kevin Watson & Márton Sóskuthy (2021): Systematic co-variation of monophthongs across speakers of New Zealand English. Journal of Phonetics. Elsevier. 88. 101096. doi:10.1016/j.wocn.2021.101096

Examples

```
onze_pca <- prcomp(onze_intercepts |> dplyr::select(-speaker), scale = TRUE)

# Plot PC1 with a cutoff value of 60%
pca_contrib_plot(onze_pca, pc_no = 1, cutoff = 60)

# Plot PC2 with no cutoff value.
pca_contrib_plot(onze_pca, pc_no = 2, cutoff = NULL)
```

`pca_rotate_2d` *Manually rotate two PCs around the origin*

Description

It is sometimes convenient to rotate principal components to most closely align with a sensible interpretation in terms of the original variables or to compare the results of PCA applied to two distinct datasets. This function allows for manual 2D rotations of principal components.

Usage

```
pca_rotate_2d(pca_obj, angle, pcs = c(1, 2))
```

Arguments

`pca_obj`	The result of a call to `prcomp()` or `princomp()`. **NB** It does not make sense to apply this function to the output of `pca_test`.
`angle`	A number indicating the number of degrees to rotate around the origin clockwise. Negative values will rotated counterclockwise.
`pcs`	A two-element vector identifying the two PCs to rotate.

Details

NB: rotated components are not principal components. They no longer explain maximal variance. Rotated components should not be referred to as 'principal components'. The simplest approach is just to call them 'components' after describing the rotation. This function modifies objects of the class 'prcomp' and 'princomp', adding an additional 'note' which collects all the rotations which have been applied. This allows any plotting function which works with the outputs of `prcomp()` or `princomp()` to work. This may result in plots which incorrectly identify rotated components as principal components. Be careful not to include any such plot in a research output.

Value

An object matching the class of `pca_obj` with loadings, scores, and variance explained by each component modified.

Examples

```
pca_obj <- prcomp(onze_intercepts |> dplyr::select(-speaker), scale=TRUE)

# Rotate PCs 3 and 6 by 10 degrees.
rotated_pca <- pca_rotate_2d(pca_obj, 10, pcs = c(3,6))
```

`pca_rotate_procrustes` *Apply Procrustes rotation to PCA loadings and scores.*

Description

It is sometimes convenient to rotate principal components to align PCA applied to one dataset with PCA applied to another. This function allows for Procrustes rotation of Principal Components, without scaling. That is, we rotate and/or flip loadings or scores so that the PCA analyses to be rotated most closely matches the loadings (or scores) from the target PCA analysis.

Usage

```
pca_rotate_procrustes(
  to_rotate,
  target,
  max_pcs,
  rotate = "loadings",
  rotation_variables = "all"
)
```

Arguments

`to_rotate`	an object of class `princomp` or `prcomp`.
`target`	an object of class `princomp` or `prcomp`
`max_pcs`	an integer. Rotation will be applied from PC1 up to `max_pcs`.
`rotate`	a string, either "loadings" or "scores", to identify whether the loadings of `to_rotate` should be aligned with `target` or the scores (default: "loadings")
`rotation_variables`	a string, names of variables to be used in the rotation. Applied to rotation of loadings when two datasets have only partial overlap of variables. (default: "all", which uses all variables).

Details

NB: rotated components are not principal components. They no longer explain maximal variance. Rotated components should not be referred to as 'principal components'. The simplest approach is just to call them 'components' after describing the rotation. This function modifies objects of the class 'prcomp' and 'princomp', adding an additional 'note' which collects all the rotations which have been applied. This allows any plotting function which works with the outputs of `prcomp()` or `princomp()` to work. This may result in plots which incorrectly identify rotated components as principal components. Be careful not to include any such plot in a research output.

Value

an object matching the class of `to_rotate`.

Examples

```
# PCA on a subset of ONZE speakers
onze_pca <- prcomp(
  onze_intercepts |> dplyr::select(-speaker),
  scale = TRUE
 )

 # PCA on all ONZE speakers
 onze_full <- prcomp(
  onze_intercepts_full |> dplyr::select(-speaker),
  scale = TRUE
 )

 # rotate subset to match loadings of `onze_full`
 rotated_pca <- onze_pca |>
   pca_rotate_procrustes(
     onze_full, max_pcs = 5
   )
```

`pca_test` *PCA with confidence intervals and null distributions*

Description

Permute and bootstrap data fed to PCA `n` times. Bootstrapped data is used to estimate confidence bands for variance explained by each PC and for each loading. Squared loadings are multiplied by the squared eigenvalue of the relevant PC. This ranks the loadings of PCs which explain a lot of variance higher than those from PCs which explain less. This approach to PCA testing follows Carmago (2022) and Vieria (2012). This approach differs from Carmago's PCAtest package by separating data generation and plotting.

Usage

```
pca_test(
  pca_data,
  n = 100,
  scale = TRUE,
  variance_confint = 0.95,
  loadings_confint = 0.9
)
```

Arguments

`pca_data`	data fed to the `prcomp` function.
`n`	the number of times to permute and bootstrap that data. **Warning:** high values will take a long time to compute.
`scale`	whether the PCA variables should be scaled (default: TRUE).
`variance_confint`	size of confidence intervals for variance explained (default: 0.95).
`loadings_confint`	size of confidence intervals for index loadings (default: 0.9).

Details

Default confidence bands on variance explained at 0.95 (i.e. alpha of 0.05). In line with Vieria (2012), the default confidence bands on the index loadings are at 0.9.

See `plot_loadings()` and `plot_variance_explained()` for useful plotting functions.

Value

object of class `pca_test_results`, containing:

- `$variance` a tibble containing the variances explained and confidence intervals for each PC.
- `$loadings` a tibble containing the index loadings and confidence intervals for each variable and PC.

- `$raw_data` a tibble containing the variance explained and loadings for each bootstrapped and permuted analysis.
- `$variance_confint` confidence intervals applied to variance explained.
- `$loadings_confint` confidence interval applied to loadings.
- `$n` the number of iterations of both permutation and bootstrapping.

References

Camargo, Arley (2022), PCAtest: testing the statistical significance of Principal Component Analysis in R. *PeerJ* 10. e12967. doi:10.7717/peerj.12967

Vieira, Vasco (2012): Permutation tests to estimate significances on Principal Components Analysis. *Computational Ecology and Software* 2. 103–123.

Examples

```
onze_pca <- pca_test(
  onze_intercepts |> dplyr::select(-speaker),
  n = 10,
  scale = TRUE
)
summary(onze_pca)
```

`pc_flip` *Flip PC loadings*

Description

The sign of the loadings and scores generated by PCA is arbitrary. Sometimes it is convenient to flip them so that all positive loadings/scores become negative (and vice versa). Sometimes one direction leads to a more natural interpretation. It is also useful when comparing the results of PCA across multiple data sets. This function will flip loadings and scores for PCA analyses carried out by the base R `prcomp()` and `princomp()` functions and for the `pca_test()` function from this package. If you specify only `pc_no` you will flip the loadings and scores for that PC. You can also specify a variable which you would like to have a positive loading in the resulting PCA.

Usage

```
pc_flip(pca_obj, pc_no, flip_var = NULL)
```

Arguments

`pca_obj`	The result of a call to `prcomp()`, `princomp()` or `pca_test`.
`pc_no`	An integer, indicating which PC is to be flipped.
`flip_var`	An optional name of a variable which will become positive in the PC indicated by `pc_no`.

Value

An object matching the class of `pca_obj` with relevant PC modified.

Examples

```
pca_obj <- prcomp(onze_intercepts |> dplyr::select(-speaker), scale=TRUE)

# flip the second PC
flipped_pca <- pc_flip(pca_obj, pc_no = 2)

# flip (if necessary) the third PC, so that the "F1_GOOSE" variable has
# a positive loading
flipped_pca <- pc_flip(pca_obj, pc_no = 3, flip_var = "F1_GOOSE")
```

`permutation_test` *Run permutation test on PCA analysis.*

Description

[Superseded] Permute data fed to PCA a given number of times, collecting the number of significant pairwise correlations in the permuted data and the variances explained for a given number of PCs.

Usage

```
permutation_test(
  pca_data,
  pc_n = 5,
  n = 100,
  scale = TRUE,
  cor.method = "pearson"
)
```

Arguments

`pca_data`	data fed to the `prcomp` function. Remove non-continuous variables.
`pc_n`	the number of PCs to collect variance explained from.
`n`	the number of times to permute that data. **Warning:** high values will take a long time to compute.
`scale`	whether the PCA variables should be scaled (default = TRUE).
`cor.method`	method to use for correlations (default = "pearson"). Alternative is "spearman".

Details

This function is now superseded. Use `correlation_test()` for pairwise correlations and `pca_test()` for variance explained and loadings.

Value

object of class `permutation_test`

- `$permuted_variances` n x pc_no matrix of variances explained by first pc_no PCs in n permutations of original data.
- `$permuted_correlations` list of length n of significant pairwise correlations in n permutations of the data (<= 0.05).
- `$actual_variances` pc_n x 2 tibble of variances explained by first pc_n PCs with original data.
- `$actual_correlations` the number of significant pairwise correlations (<= 0.05) in the original data.

Examples

```
permutation_test(
  onze_intercepts |> dplyr::select(-speaker),
  pc_n = 5,
  n = 10,
  scale = TRUE,
  cor.method = 'pearson'
 )
```

`plot_correlation_counts`

Plot of correlation counts from `correlation_test` *object*

Description

Plot the number of statistically significant pairwise correlations in a data set given an alpha value against the distribution of counts of statistically significant pairwise correlations in permuted data. This is an informal test which is useful to convincing yourself that there is structure in your data which PCA might be able to uncover.

Usage

```
plot_correlation_counts(cor_test, alpha = 0.05, half_violin = FALSE)
```

Arguments

`cor_test`	an object of class `correlation_test` generated by `correlation_test`.
`alpha`	significance level for counting correlation as significant.
`half_violin`	Plot correlation counts using a half violin plot and half point plot. Quantiles are not currently supported.

Details

The resulting plot presents the distribution of *counts* of statistically significant correlations at a given alpha level in the permuted data and the count of statistically significant correlations in the original data. If the red dot is above the uppermost line inside the blue violin plot, we say the number of statistically significant correlations in the real data is itself statistically significant. Usually this is used as a rough sanity check in the course of a PCA workflow and we want to see the red dot well above the violin (as in the example below).

The resulting plot is a `ggplot2` plot and can be modified using functions from that package. For instance, titles can be removed using the `ggplot2::labs()` function (as in the examples below).

Value

`ggplot` object.

Examples

```
# Test correlations (use at least n = 100)
cor_test <- correlation_test(onze_intercepts |>
  dplyr::select(-speaker), n = 10)
cor_plot <- plot_correlation_counts(cor_test)
cor_plot

# make statistical test more strict by reducing the alpha.
cor_plot_strict <- plot_correlation_counts(cor_test, alpha = 0.01)

# modify plot using `ggplot2` functions, e.g.
cor_plot_strict +
  ggplot2::labs(title = NULL) +
  ggplot2::theme_bw()
```

`plot_correlation_magnitudes` *Plot distribution of correlations from* `correlation_test` *object*

Description

This plot type is used in Brand et al. (2021). It presents the magnitudes of the correlations from the real data as a solid red line, and the correlations from each iteration of the permutation test as light blue lines. This gives a visual sense of the distribution of random correlations compared with those in the actual data. If there are significant pairwise correlations in the data, the thick red line should be visually lower and wider across the plot than the thinner blue lines. If there are no significant pairwise correlations, then the thick red line will have the same shape as the blue lines.

Usage

```
plot_correlation_magnitudes(cor_test)
```

Arguments

`cor_test`	an object of class `correlation_test` generated by `correlation_test`.

Value

`ggplot` object.

References

Brand, James, Jen Hay, Lynn Clark, Kevin Watson & Márton Sóskuthy (2021): Systematic co-variation of monophthongs across speakers of New Zealand English. Journal of Phonetics. Elsevier. 88. 101096. doi:10.1016/j.wocn.2021.101096

Examples

```
# Test correlations (use at least n = 100)
cor_test <- correlation_test(onze_intercepts |>
  dplyr::select(-speaker), n = 10)
cor_plot <- plot_correlation_magnitudes(cor_test)
cor_plot

# modify plot using `ggplot2` functions, e.g.
cor_plot +
  ggplot2::labs(title = NULL) +
  ggplot2::theme_bw()
```

`plot_loadings` *Plot PC index loadings from* `pca_test` *object.*

Description

Index loadings (Vieira 2012) are presented with confidence intervals on the sampling distribution generated by bootstrapping and a null distribution generated by permutation.

Usage

```
plot_loadings(
  pca_test,
  pc_no = 1,
  violin = FALSE,
  filter_boots = FALSE,
  quantile_threshold = 0.25
)
```

Arguments

`pca_test`	an object of class pca_test_results generated by `pca_test`.
`pc_no`	An integer indicating which PC to plot.
`violin`	If TRUE, violin plots are added for the confidence intervals of the sampling distribution.
`filter_boots`	if TRUE, only bootstrap iterations in which the variable with the highest median loading is above `quantile_threshold`.
`quantile_threshold`	a real value between 0 and 1. Use this to change the threshold used for filtering bootstrap iterations. The default is 0.25.

Details

If PCs are unstable, there is an option (`filter_boots`) to take only the bootstrap iterations in which the variable with the highest median loading across all iterations is above `quantile_threshold` (default: 0.25). This helps to reveal reliable connections of this variable with other variables in the data set.

Value

`ggplot` object.

References

Vieira, Vasco (2012): Permutation tests to estimate significances on Principal Components Analysis. *Computational Ecology and Software* 2. 103–123.

Examples

```
onze_pca <- pca_test(onze_intercepts |> dplyr::select(-speaker), n = 10)
# Plot PC1
plot_loadings(onze_pca, pc_no=1)
# Plot PC2 with violins (not particularly useful in this case!)
plot_loadings(onze_pca, pc_no=2, violin = TRUE)
```

`plot_mds_test` *Plot* `mds_test()` *results*

Description

[Experimental] Plot output from `mds_test()`.

Usage

```
plot_mds_test(mds_test)
```

Arguments

`mds_test`	Object of class `mds_test_results` (generated by `mds_test()`).

Value

`ggplot` object.

Examples

```
mds_result <- mds_test(
    sim_matrix,
    n_boots = 10,
    n_perms = 10,
    test_dimensions = 3,
    mds_type = 'interval'
 )
 plot_mds_test(mds_result)
```

`plot_pc_input` *Plot Scores from Significant PCs Against PCA Input*

Description

It is sometimes useful to see the relationship between PCs and the raw values of the input data fed into PCA. This function takes the results of running `pca_test`, the scores for each speaker from the pca object, and the raw data fed into the PCA analysis. In the usual model-to-pca analysis pipeline, the resulting plot depicts by-speaker random intercepts for each vowel and an indication of which variables are significantly loaded onto the PCs. It allows the researcher to visualise the strength of the relationship between intercepts and PC scores.

Usage

```
plot_pc_input(pca_object, pca_data, pca_test)
```

Arguments

`pca_object`	Output of `prcomp`.
`pca_data`	Data fed into `prcomp`. This should not include speaker identifiers.
`pca_test`	Output of `pca_test`

Value

a `ggplot` object.

Examples

```
pca_data <- onze_intercepts |> dplyr::select(-speaker)
onze_pca <- prcomp(pca_data, scale = TRUE)
onze_pca_test <- pca_test(pca_data, n = 5) # Increase n to at least 100 in practice.
plot_pc_input(onze_pca, pca_data, onze_pca_test)
```

`plot_pc_vs` *Plot PC loadings in vowel space*

Description

Plot loadings from a PCA analysis carried out on vocalic data. Vowel positions mean values are at the mean with arrows indicating loadings. Loadings are multiplied by the standard deviation, by vowel, of the initial input data. This is OK for getting a quick, intuitive, interpretation of what the PCs mean in the vowel space. When using a model-to-PCA pipeline, it is not recommended to use these plots directly in publications as the models should more reliably control variation in vocalic readings than taking the standard mean and standard deviation.

Usage

```
plot_pc_vs(vowel_data, pca_obj, pc_no = 1, is_sig = FALSE)
```

Arguments

`vowel_data`	A dataframe whose first four columns are speaker ids, vowel ids, F1 values, and F2 values.
`pca_obj`	The result of a call to `prcomp()`, `princomp()` or `pca_test()`.
`pc_no`	An integer, indicating which PC to plot (default is PC1).
`is_sig`	A boolean, indicating whether only 'significant' loadings, according to `pca_test` should be plotted (only works with objects of class `pca_test_results`).

Value

a `ggplot` object.

Examples

```
onze_pca <- prcomp(onze_intercepts |> dplyr::select(-speaker), scale=TRUE)
# Default is to plot PC1
plot_pc_vs(onze_vowels, onze_pca)
# Or plot another PC with `pc_no`
plot_pc_vs(onze_vowels, onze_pca, pc_no = 3)
```

`plot_permutation_test` *Create plot from* `permutation_test()`.

Description

[Superseded] Plots results of a permutation test carried out with the `permutation_test()` function. Now use either `correlation_test()` or `pca_test()` and the associated plotting functions.

Usage

```
plot_permutation_test(permutation_results, violin = FALSE)
```

Arguments

`permutation_results`	object of class `permutation_results`.
`violin`	Determines whether the variances explained are depicted by distinct violin plots for each PC or by connected lines. the advantage of lines is that they correctly indicate that values for each PC depend on one another within a given permutation. That is, if an earlier PC soaks up a lot of the variation in a data set, then there is less variation left to explain by subsequent PCs. Default value is `FALSE`.

Value

`ggplot` object.

Examples

```
onze_perm <- permutation_test(
  onze_intercepts |> dplyr::select(-speaker),
  pc_n = 5,
  n = 10,
  scale = TRUE,
  cor.method = 'pearson'
 )
plot_permutation_test(onze_perm)
```

`plot_procrustes_loadings` *Plot loadings with confidence bands from* `procrustes_loadings()`

Description

[Experimental] Plot index loadings or loadings with confidence intervals and null distributions generated by bootstrapping and permutation followed by Procrustes rotation. This approach works when PC loadings are unstable due to multiple PCs explaining similar amounts of variance. This is an alternative to the use of bootstrapping without Procrustes rotation (as in `pca_test()`) and avoids the need for the use of the `filter_boots` argument to `plot_loadings()`.

Usage

```
plot_procrustes_loadings(proc_loadings, pc_no = 1, loadings_confint = 0.9)
```

Arguments

`proc_loadings`	a tibble, generated by `procrustes_loadings()`
`pc_no`	an integer indicating which PC to plot.
`loadings_confint`	confidence limits for generated confidence intervals. (default: 0.9 to match `pca_test()`).

Value

a `ggplot` object.

Examples

```
proc_loadings <- procrustes_loadings(
  pca_data = onze_intercepts |> dplyr::select(-speaker),
  max_pcs = 3,
  index = TRUE,
  n = 10, # set this to at least 100 in actual use.
  scale = TRUE
 )

 plot_procrustes_loadings(proc_loadings, pc_no = 2)
```

`plot_variance_explained` *Create plot of variances explained from* `pca_test` *object*

Description

The variance explained by each PC in a dataset is plotted with confidence intervals generated by bootstrapping and a null distribution generated by permutation. The function accepts the result of calling the `pca_test` function.

Usage

```
plot_variance_explained(pca_test, pc_max = NA, percent = TRUE)
```

Arguments

`pca_test`	an object of class pca_test_results generated by `pca_test`.
`pc_max`	the maximum number of PCs to plot. If NA, plot all PCs.
`percent`	if TRUE, represent variance explained as a percentage. If FALSE, represent as eigenvalues.

Details

By default, variance explained is represented as a percentage. If the argument `percent` is set to FALSE, then the variance explained is represented by the eigenvalues corresponding to each PC.

Value

`ggplot` object.

Examples

```
onze_pca <- pca_test(onze_intercepts |> dplyr::select(-speaker), n = 10)
# Plot with percentages
plot_variance_explained(onze_pca)
# Plot with eigenvalues and only the first 5 PCs.
plot_variance_explained(onze_pca, pc_max = 5, percent = FALSE)
```

`plot_vowel_space` *Plot vowel space for speaker or speakers.*

Description

Given vowel data with the first column identifying speakers, the second identifying vowels, the third containing F1 and the fourth containing F2 values, plot a vowel space using the speaker's mean values for each vowel. Typically it is best to produce a plot from scratch. The primary purpose of this function is to generate quick plots for interactive use, rather than to produce plots for publication.

Usage

```
plot_vowel_space(
  vowel_data,
  speakers = NULL,
  vowel_colours = NULL,
  label_size = 4,
  means_only = TRUE,
  ellipses = FALSE,
  point_alpha = 0.1,
  facet = TRUE
)
```

Arguments

`vowel_data`	data frame of vowel tokens as described above.
`speakers`	list of speaker identifiers for speaker whose vowel space is to be plotted.
`vowel_colours`	a named list of vowel = colour entries to indicate which colour to plot each vowel.
`label_size`	It is often convenient to adjust the size of the labels (in pts). Default is 4.

`means_only`	whether to plot means only or all data points. Default: TRUE.
`ellipses`	whether to 95% confidence ellipses. Only works if means_only is FALSE. Default is FALSE.
`point_alpha`	alpha value for data points if means_only is FALSE.
`facet`	whether to plot distinct speakers in distinct facets. Default is TRUE.

Value

`ggplot` object.

Examples

```
# Plot mean vowel space across
plot_vowel_space(
  onze_vowels,
  speakers = NULL,
  vowel_colours = NULL,
  label_size = 4,
  means_only = TRUE,
  ellipses = FALSE,
  point_alpha = 0.1,
  facet = FALSE
 )
```

`procrustes_loadings` *Generate distribution of (index) loadings using the bootstrap and Procrustes rotation.*

Description

[Experimental] Generate distribution of loadings or signed index loadings for Principal Components. These are used in order to estimate confidence intervals for loadings and, if signed index loadings are used, also a null distribution for tests of statistical significance. Plot the results using `plot_procrustes_loadings()`.

Usage

```
procrustes_loadings(pca_data, max_pcs, index = TRUE, n = 500, scale = TRUE)
```

Arguments

`pca_data`	data fed to the `prcomp` function.
`max_pcs`	maximum number of PCs to rotate.
`index`	whether to use signed index loadings rather than loadings (default: TRUE)
`n`	the number of bootstrapped and permuted samples.
`scale`	whether the variables in `pca_data` should be scaled before PCA (default: TRUE)

Value

a tibble, with columns:

- `source` either "Sampling", "Null" or "Original", identifying where the loadings comes from. "Original" identifies loadings from the full dataset, "Sampling" identifies loadings from the bootstrapped samples, "Null" identifes loadings from permuted versions of the data.
- `id` identifies which iteration of either permutation or bootstrapping the loading comes from.
- `variable` indicates the variable corresponding to the loading.
- a column containing the loading for each `PC` up to `max_pcs`.

Examples

```
proc_loadings <- procrustes_loadings(
  pca_data = onze_intercepts |> dplyr::select(-speaker),
  max_pcs = 3,
  index = TRUE,
  n = 10, # set this to at least 100 in actual use.
  scale = TRUE
 )
```

`qb_intervals` *Formant and amplitude for intervals of QuakeBox monologues*

Description

QuakeBox monologues are divided into intervals of fixed length within mean values are calcualted for formants, amplitude, and articulation rate. Data from 77 speakers is provide (the same sample as `qb_vowels`).

Usage

```
qb_intervals
```

Format

A data frame with 53940 rows and 10 variables:

interval_length Length of interval in seconds.

speaker Anonymised speaker code (char).

interval Time in seconds at which interval ends.

articulation_rate Mean articulation rate within interval.

amplitude Mean maximum amplitude within interval.

DRESS_F1 Speaker intercept from GAMM model of DRESS F1.

DRESS_F2 Speaker intercept from GAMM model of DRESS F2.

FLEECE_F1	Speaker intercept from GAMM model of FLEECE F1.
FLEECE_F2	Speaker intercept from GAMM model of FLEECE F2.
GOOSE_F1	Speaker intercept from GAMM model of GOOSE F1.
GOOSE_F2	Speaker intercept from GAMM model of GOOSE F2.
KIT_F1	Speaker intercept from GAMM model of KIT F1.
KIT_F2	Speaker intercept from GAMM model of KIT F2.
LOT_F1	Speaker intercept from GAMM model of LOT F1.
LOT_F2	Speaker intercept from GAMM model of LOT F2.
NURSE_F1	Speaker intercept from GAMM model of NURSE F1.
NURSE_F2	Speaker intercept from GAMM model of NURSE F2.
START_F1	Speaker intercept from GAMM model of START F1.
START_F2	Speaker intercept from GAMM model of START F2.
STRUT_F1	Speaker intercept from GAMM model of STRUT F1.
STRUT_F2	Speaker intercept from GAMM model of STRUT F2.
THOUGHT_F1	Speaker intercept from GAMM model of THOUGHT F1.
THOUGHT_F2	Speaker intercept from GAMM model of THOUGHT F2.
TRAP_F1	Speaker intercept from GAMM model of TRAP F1.
TRAP_F2	Speaker intercept from GAMM model of TRAP F2.

Details

Two interval lengths are given: 60 seconds and 240 seconds.

Formant data is z-scored by speaker and vowel, while the amplitude and articulation rate are z-scored by speaker.

Original data was generated for Wilson Black et al. (2023).

Source

`https://osf.io/m8nkh/`

References

Wilson Black, Joshua, Jennifer Hay, Lynn Clark & James Brand (2023): The overlooked effect of amplitude on within-speaker vowel variation. Linguistics Vanguard. Walter de Gruyter GmbH. 9(1). 173–189. doi:10.1515/lingvan-2022-0086

`qb_vowels` *Formants from QuakeBox 1*

Description

A dataset containing formant values, amplitude, articulation rate, and following segment data for 10 New Zealand English monophthongs, along with participant demographics.

Usage

```
qb_vowels
```

Format

A data frame with 26331 rows and 14 variables:

speaker Anonymised speaker code (char).

vowel Wells lexical sets for 10 NZE monophthongs. Levels: DRESS, FLEECE, GOOSE, KIT, LOT, NURSE, START, STRUT, THOUGHT, TRAP, FOOT (char).

F1_50 First formant in Hz, extracted from vowel mid-point using LaBB-CAT interface with Praat.

F2_50 Second formant in Hz, extracted from vowel mid-point using LaBB-CAT interface with Praat.

participant_age_category Age category of speaker. Values: 18-25, 26-35, 36-45, ..., 76-85 (char).

participant_gender Gender of participant. Values: M, F (char).

participant_nz_ethnic New Zealand ethnic category of participant. Values: NZ mixed ethnicity, NZ European, Other (char).

word_freq Frequency of word from which vowel token is taken in CELEX.

word Anonymised word id (char).

time Time in seconds at which vowel segment starts.

vowel_duration Length of vowel in seconds.

articulation_rate Articulation rate of utterance from which token is taken.

following_segment_category Category of following segment. NB: liquids have already been removed. Levels: labial, velar, other (factor).

amplitude Maximum amplitude of word from which vowel token is taken, generated by LaBB-CAT interface with Praat.

Details

Original data was generated for Wilson Black et al. (2023).

Source

`https://osf.io/m8nkh/`

References

Wilson Black, Joshua, Jennifer Hay, Lynn Clark & James Brand (2023): The overlooked effect of amplitude on within-speaker vowel variation. Linguistics Vanguard. Walter de Gruyter GmbH. 9(1). 173–189. doi:10.1515/lingvan-2022-0086

`sim_matrix` *Similarity matrix from online perception test.*

Description

Mean similarity ratings for 38 QuakeBox speakers from an online pairwise similarity task. Random noise added.

Usage

```
sim_matrix
```

Format

A 38x38 matrix

`summary.correlation_test` *Summary function for correlation test object. Set alpha to change significance level.*

Description

Set alpha to change significance level and n_cors to change number of pairwise correlations given.

Usage

```
## S3 method for class 'correlation_test'
summary(object, alpha = 0.05, n_cors = 5, ...)
```

Arguments

`object`	object of class `correlation test`,
`alpha`	significance level for counting correlation as significant.
`n_cors`	number of pairwise correlations to list.
`...`	additional arguments affecting the summary produced.

Value

a `glue` object.

Index