Package 'fxTWAPLS'

October 13, 2020

Title An Improved Version of WA-PLS

Version 0.0.2

Description The goal of this package is to provide an improved version of WA-PLS (Weighted Averaging Partial Least Squares) by including the tolerances of taxa and the frequency of the sampled climate variable. This package also provides a way of leave-out cross-validation that removes both the test site and sites that are both geographically close and climatically close for each cycle, to avoid the risk of pseudo-replication.

License GPL-3

Encoding UTF-8

LazyData true

URL https://github.com/special-uor/fxTWAPLS, https://special-uor.github.io/fxTWAPLS/, https://research.reading.ac.uk/palaeoclimate/

BugReports https://github.com/special-uor/fxTWAPLS/issues

Imports doParallel, foreach, geosphere, ggplot2, MASS, matrixStats, parallel, tictoc

Suggests badger, hexSticker, knitr, kableExtra, pkgdown, rmarkdown, scales, testthat

Depends R (>= 3.6)

RoxygenNote 7.1.1

Language en-GB

NeedsCompilation no

Author Mengmeng Liu [aut] (<https://orcid.org/0000-0001-6250-0148>), Roberto Villegas-Diaz [aut, cre] (<https://orcid.org/0000-0001-5036-8661>), SPECIAL Research Group @ University of Reading [cph]

Maintainer Roberto Villegas-Diaz <r.villegas-diaz@reading.ac.uk>

Repository CRAN

Date/Publication 2020-10-13 14:50:03 UTC

R topics documented:

`comb_pb` *Combine results with progress bar*

Description

Combine results with progress bar, to be used in combination with `foreach::foreach`

Usage

```
comb_pb(iterator, FUN = rbind, ...)
```

Arguments

`iterator`	number of iterations
`FUN`	function to combine the results (default: `rbind`)
`...`	optional parameters

Examples

```
# Load binary operator for backend
`%do%` <- foreach::`%do%`
N <- 5
out <- foreach::foreach(i = 1:N,
                        .combine = comb_pb(N)) %do% {
                        Sys.sleep(1)
                        i
                       }
```

cv.pr.w *Pseudo-removed leave-out cross-validation*

Description

Pseudo-removed leave-out cross-validation

Usage

```
cv.pr.w(
  modern_taxa,
  modern_climate,
  nPLS = 5,
  trainfun,
  predictfun,
  pseudo,
  usefx = FALSE,
  fx = NA,
  cpus = 4,
  test_mode = TRUE,
  test_it = 5
)
```

Arguments

modern_taxa	the modern taxa abundance data, each row represents a sampling site, each column represents a taxon.
modern_climate	the modern climate value at each sampling site
nPLS	the number of components to be extracted
trainfun	training function you want to use, either WAPLS.w or TWAPLS.w
predictfun	predict function you want to use: if trainfun is WAPLS.w, then this should be WAPLS.predict.w; if trainfun is TWAPLS.w, then this should be TWAPLS.predict.w
pseudo	the geographically and climatically close sites to each test site, obtained from get_pseudo function
usefx	boolean flag on whether or not use fx correction.
fx	the frequency of the climate value for fx correction: if usefx is FALSE, this should be NA; otherwise, this should be obtained from the fx function.
cpus	number of CPUs for simultaneous iterations to execute, check parallel::detectCores() for available CPUs on your machine.
test_mode	boolean flag to execute the function with a limited number of iterations, test_it, for testing purposes only.
test_it	number of iterations to use in the test mode

Value

leave-one-out cross validation results

See Also

fx, TWAPLS.w, TWAPLS.predict.w, WAPLS.w, and WAPLS.predict.w

Examples

```
# Load modern pollen data
modern_pollen <- read.csv(system.file("extdata",
                                      "Modern_Pollen_gdd_alpha_Tmin.csv",
                                      package = "fxTWAPLS",
                                      mustWork = TRUE))
#' # Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "Abies")
taxaColMax <- which(colnames(modern_pollen) == "Zygophyllaceae")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

point <- modern_pollen[, c("Long", "Lat")]

test_mode <- TRUE # It should be set to FALSE before running
dist <- fxTWAPLS::get_distance(point,
                               cpus = 2, # Remove the following line
                               test_mode = test_mode)
pseudo_Tmin <- fxTWAPLS::get_pseudo(dist,
                                    modern_pollen$Tmin,
                                    cpus = 2, # Remove the following line
                                    test_mode = test_mode)
# Test WAPLS
cv_pr_Tmin <- fxTWAPLS::cv.pr.w(taxa,
                                modern_pollen$Tmin,
                                nPLS = 5,
                                fxTWAPLS::TWAPLS.w,
                                fxTWAPLS::TWAPLS.predict.w,
                                pseudo_Tmin,
                                cpus = 2, # Remove the following line
                                test_mode = test_mode)
# Test TWAPLS
cv_pr_t_Tmin <- fxTWAPLS::cv.pr.w(taxa,
                                  modern_pollen$Tmin,
                                  nPLS = 5,
                                  fxTWAPLS::TWAPLS.w,
                                  fxTWAPLS::TWAPLS.predict.w,
                                  pseudo_Tmin,
                                  cpus = 2, # Remove the following line
                                  test_mode = test_mode)
```

cv.w *Leave-one-out cross-validation*

Description

Leave-one-out cross-validation as rioja (https://cran.r-project.org/package=rioja)

Usage

```
cv.w(
  modern_taxa,
  modern_climate,
  nPLS = 5,
  trainfun,
  predictfun,
  usefx = FALSE,
  fx = NA,
  cpus = 4,
  test_mode = FALSE,
  test_it = 5
)
```

Arguments

modern_taxa	the modern taxa abundance data, each row represents a sampling site, each column represents a taxon.
modern_climate	the modern climate value at each sampling site
nPLS	the number of components to be extracted
trainfun	training function you want to use, either WAPLS.w or TWAPLS.w
predictfun	predict function you want to use: if trainfun is WAPLS.w, then this should be WAPLS.predict.w; if trainfun is TWAPLS.w, then this should be TWAPLS.predict.w
usefx	boolean flag on whether or not use fx correction.
fx	the frequency of the climate value for fx correction: if usefx is FALSE, this should be NA; otherwise, this should be obtained from the fx function.
cpus	number of CPUs for simultaneous iterations to execute, check parallel::detectCores() for available CPUs on your machine.
test_mode	boolean flag to execute the function with a limited number of iterations, test_it, for testing purposes only.
test_it	number of iterations to use in the test mode

Value

leave-one-out cross validation results

See Also

fx, TWAPLS.w, TWAPLS.predict.w, WAPLS.w, and WAPLS.predict.w

Examples

```
# Load modern pollen data
modern_pollen <- read.csv(system.file("extdata",
                                      "Modern_Pollen_gdd_alpha_Tmin.csv",
                                      package = "fxTWAPLS",
                                      mustWork = TRUE))

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "Abies")
taxaColMax <- which(colnames(modern_pollen) == "Zygophyllaceae")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Get the frequency of each climate variable fx
fx_Tmin <- fxTWAPLS::fx(modern_pollen$Tmin, bin = 0.02)
fx_gdd <- fxTWAPLS::fx(modern_pollen$gdd, bin = 20)
fx_alpha <- fxTWAPLS::fx(modern_pollen$alpha, bin = 0.002)

# MTCO
## fx
fit_Tmin <- fxTWAPLS::WAPLS.w(taxa, modern_pollen$Tmin, nPLS = 5)
## LOOCV
test_mode <- TRUE # It should be set to FALSE before running
### without fx
cv_Tmin <- fxTWAPLS::cv.w(taxa,
                          modern_pollen$Tmin,
                          nPLS = 5,
                          fxTWAPLS::WAPLS.w,
                          fxTWAPLS::WAPLS.predict.w,
                          cpus = 2, # Remove the following line
                          test_mode = test_mode)
### with fx
cv_f_Tmin <- fxTWAPLS::cv.w(taxa,
                            modern_pollen$Tmin,
                            nPLS = 5,
                            fxTWAPLS::WAPLS.w,
                            fxTWAPLS::WAPLS.predict.w,
                            usefx = TRUE,
                            fx = fx_Tmin,
                            cpus = 2, # Remove the following line
                            test_mode = test_mode)
```

---

fx *Get frequency of the climate value*

Description

Function to get the frequency of the climate value, which will be used to provide `fx` correction for WA-PLS and TWA-PLS

Usage

```
fx(x, bin)
```

Arguments

`x`	the modern climate values
`bin`	bin-width to get the frequency of the modern climate values

Value

the frequency of the modern climate values

See Also

`cv.w`, `cv.pr.w`, and `sse.sample`

Examples

```
# Load modern pollen data
modern_pollen <- read.csv(system.file("extdata",
                                      "Modern_Pollen_gdd_alpha_Tmin.csv",
                                      package = "fxTWAPLS",
                                      mustWork = TRUE))

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "Abies")
taxaColMax <- which(colnames(modern_pollen) == "Zygophyllaceae")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Get the frequency of each climate variable fx
fx_Tmin <- fxTWAPLS::fx(modern_pollen$Tmin, bin = 0.02)
fx_gdd <- fxTWAPLS::fx(modern_pollen$gdd, bin = 20)
fx_alpha <- fxTWAPLS::fx(modern_pollen$alpha, bin = 0.002)
```

`get_distance` *Get the distance between points*

Description

Get the distance between points, the output will be used in `get_pseudo`

Usage

```
get_distance(point, cpus = 4, test_mode = FALSE, test_it = 5)
```

Arguments

point	each row represents a sampling site, the first column is longitude and the second column is latitude, both in decimal format
cpus	number of CPUs for simultaneous iterations to execute, check `parallel::detectCores()` for available CPUs on your machine.
test_mode	boolean flag to execute the function with a limited number of iterations, `test_it`, for testing purposes only.
test_it	number of iterations to use in the test mode

Value

distance matrix, the value at the `i-th` row, means the distance between the `i-th` sampling site and the whole sampling sites

See Also

get_pseudo

Examples

```
# Load modern pollen data
modern_pollen <- read.csv(system.file("extdata",
                                      "Modern_Pollen_gdd_alpha_Tmin.csv",
                                      package = "fxTWAPLS",
                                      mustWork = TRUE))
point <- modern_pollen[, c("Long", "Lat")]

test_mode <- TRUE # It should be set to FALSE before running
dist <- fxTWAPLS::get_distance(point,
                               cpus = 2, # Remove the following line
                               test_mode = test_mode)
```

---

get_pseudo *Get geographically and climatically close sites*

Description

Get the sites which are both geographically and climatically close to the test site, which could result in pseudo-replication and inflate the cross-validation statistics. The output will be used in cv.pr.w

Usage

```
get_pseudo(dist, x, cpus = 4, test_mode = FALSE, test_it = 5)
```

Arguments

dist	distance matrix which contains the distance from other sites.
x	the modern climate values
cpus	number of CPUs for simultaneous iterations to execute, check `parallel::detectCores()` for available CPUs on your machine.
test_mode	boolean flag to execute the function with a limited number of iterations, `test_it`, for testing purposes only.
test_it	number of iterations to use in the test mode

Value

the geographically and climatically close sites to each test site.

See Also

get_distance

Examples

```
# Load modern pollen data
modern_pollen <- read.csv(system.file("extdata",
                                      "Modern_Pollen_gdd_alpha_Tmin.csv",
                                      package = "fxTWAPLS",
                                      mustWork = TRUE))
point <- modern_pollen[, c("Long", "Lat")]

test_mode <- TRUE # It should be set to FALSE before running
dist <- fxTWAPLS::get_distance(point,
                               cpus = 2, # Remove the following line
                               test_mode = test_mode)
pseudo_Tmin <- fxTWAPLS::get_pseudo(dist,
                                    modern_pollen$Tmin,
                                    cpus = 2, # Remove the following line
                                    test_mode = test_mode)
```

---

hex_logo *Create hexagonal logo for the package*

Description

Create hexagonal logo for the package

Usage

```
hex_logo(
  subplot = system.file("images/cave-painting.png", package = "fxTWAPLS"),
  dpi = 600,
  h_color = "#000000",
  h_fill = "#696969",
  output = system.file("images/logo.png", package = "fxTWAPLS"),
  package = "fxTWAPLS",
  p_color = "#eeeeee",
  url = "https://github.com/special-uor/fxTWAPLS",
  u_size = 1.25
)
```

Arguments

subplot	image to use as the main logo
dpi	plot resolution (dots-per-inch)
h_color	colour for hexagon border
h_fill	colour to fill hexagon
output	output file (hexagonal logo)
package	title for logo (package name)
p_color	colour for package name
url	URL for package repository or website
u_size	text size for URL

Value

hexagonal logo

Examples

```
hex_logo()
```

---

par_benchmark *Perform parallel benchmarks on a function*

Description

Perform parallel benchmarks on a function

Usage

```
par_benchmark(CPUS, FUN, plot = FALSE, quiet = FALSE, ...)
```

Arguments

CPUS	vector with the number of CPUs
FUN	parallel function, MUST have a parameter called cpus
plot	boolean flag to request a plot for the results
quiet	boolean flag to print results of each execution
...	optional arguments for the function, must be named; e.g. x = test_df

Examples

```
# Define toy function that sleeps for (2/cpus) seconds
a <- function(cpus) {Sys.sleep(2/cpus)}
par_benchmark(c(1, 2), a)

par_benchmark(c(1, 2), a, plot = TRUE)
```

---

plot_residuals *Plot the residuals*

Description

Plot the residuals, the black line is 0 line, the red line is the locally estimated scatterplot smoothing, which shows the degree of local compression

Usage

```
plot_residuals(train_output, col)
```

Arguments

train_output	Training output, can be the output of WA-PLS, WA-PLS with fx correction, TWA-PLS, or TWA-PLS with fx correction
col	choose which column of the fitted value to plot, in other words, how many number of components you want to use

Value

plotting status

See Also

TWAPLS.w and WAPLS.w

Examples

```
# Load modern pollen data
modern_pollen <- read.csv(system.file("extdata",
                                      "Modern_Pollen_gdd_alpha_Tmin.csv",
                                      package = "fxTWAPLS",
                                      mustWork = TRUE))

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "Abies")
taxaColMax <- which(colnames(modern_pollen) == "Zygophyllaceae")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Get the frequency of each climate variable fx
fx_Tmin <- fxTWAPLS::fx(modern_pollen$Tmin, bin = 0.02)
fx_gdd <- fxTWAPLS::fx(modern_pollen$gdd, bin = 20)
fx_alpha <- fxTWAPLS::fx(modern_pollen$alpha, bin = 0.002)

# MTCO
## WAPLS
fit_Tmin <- fxTWAPLS::WAPLS.w(taxa, modern_pollen$Tmin, nPLS = 5)
fit_f_Tmin <- fxTWAPLS::WAPLS.w(taxa,
                                modern_pollen$Tmin,
                                nPLS = 5,
                                usefx = TRUE,
                                fx = fx_Tmin)
## TWAPLS
fit_t_Tmin <- fxTWAPLS::TWAPLS.w(taxa, modern_pollen$Tmin, nPLS = 5)
fit_tf_Tmin <- fxTWAPLS::TWAPLS.w(taxa,
                                  modern_pollen$Tmin,
                                  nPLS = 5,
                                  usefx = TRUE,
                                  fx = fx_Tmin)
fxTWAPLS::plot_residuals(fit_Tmin, 3)
fxTWAPLS::plot_residuals(fit_f_Tmin, 3)
fxTWAPLS::plot_residuals(fit_t_Tmin, 3)
fxTWAPLS::plot_residuals(fit_tf_Tmin, 3)
```

`plot_train` *Plot the training results*

Description

Plot the training results, the black line is the 1:1 line, the red line is the linear regression line to fitted and x, which shows the degree of overall compression

Usage

```
plot_train(train_output, col)
```

Arguments

train_output	Training output, can be the output of WA-PLS, WA-PLS with fx correction, TWA-PLS, or TWA-PLS with fx correction
col	choose which column of the fitted value to plot, in other words, how many number of components you want to use

Value

plotting status

See Also

TWAPLS.w and WAPLS.w

Examples

```
# Load modern pollen data
modern_pollen <- read.csv(system.file("extdata",
                                      "Modern_Pollen_gdd_alpha_Tmin.csv",
                                      package = "fxTWAPLS",
                                      mustWork = TRUE))

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "Abies")
taxaColMax <- which(colnames(modern_pollen) == "Zygophyllaceae")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Get the frequency of each climate variable fx
fx_Tmin <- fxTWAPLS::fx(modern_pollen$Tmin, bin = 0.02)
fx_gdd <- fxTWAPLS::fx(modern_pollen$gdd, bin = 20)
fx_alpha <- fxTWAPLS::fx(modern_pollen$alpha, bin = 0.002)

# MTCO
## WAPLS
fit_Tmin <- fxTWAPLS::WAPLS.w(taxa, modern_pollen$Tmin, nPLS = 5)
fit_f_Tmin <- fxTWAPLS::WAPLS.w(taxa,
                                modern_pollen$Tmin,
                                nPLS = 5,
                                usefx = TRUE,
                                fx = fx_Tmin)
## TWAPLS
fit_t_Tmin <- fxTWAPLS::TWAPLS.w(taxa, modern_pollen$Tmin, nPLS = 5)
fit_tf_Tmin <- fxTWAPLS::TWAPLS.w(taxa,
                                  modern_pollen$Tmin,
                                  nPLS = 5,
                                  usefx = TRUE,
                                  fx = fx_Tmin)
fxTWAPLS::plot_train(fit_Tmin, 3)
fxTWAPLS::plot_train(fit_f_Tmin, 3)
fxTWAPLS::plot_train(fit_t_Tmin, 3)
fxTWAPLS::plot_train(fit_tf_Tmin, 3)
```

`rand.t.test.w` *Random t-test*

Description

Do a random t-test to the cross-validation results

Usage

```
rand.t.test.w(cvoutput, n.perm = 999)
```

Arguments

`cvoutput`	cross-validation output either from `cv.w` or `cv.pr.w`
`n.perm`	the number of permutation times to get the p value, which assesses whether using the current number of components is significantly different from using one less.

Value

a matrix of the statistics of the cross-validation results

See Also

`cv.w` and `cv.pr.w`

Examples

```
# Load modern pollen data
modern_pollen <- read.csv(system.file("extdata",
                                      "Modern_Pollen_gdd_alpha_Tmin.csv",
                                      package = "fxTWAPLS",
                                      mustWork = TRUE))

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "Abies")
taxaColMax <- which(colnames(modern_pollen) == "Zygophyllaceae")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Get the frequency of each climate variable fx
fx_Tmin <- fxTWAPLS::fx(modern_pollen$Tmin, bin = 0.02)
fx_gdd <- fxTWAPLS::fx(modern_pollen$gdd, bin = 20)
fx_alpha <- fxTWAPLS::fx(modern_pollen$alpha, bin = 0.002)

# MTCO
## fx
fit_Tmin <- fxTWAPLS::WAPLS.w(taxa, modern_pollen$Tmin, nPLS = 5)
```

```
## LOOCV
test_mode <- TRUE # It should be set to FALSE before running
### without fx
cv_Tmin <- fxTWAPLS::cv.w(taxa,
                          modern_pollen$Tmin,
                          nPLS = 5,
                          fxTWAPLS::WAPLS.w,
                          fxTWAPLS::WAPLS.predict.w,
                          cpus = 2, # Remove the following line
                          test_mode = test_mode)
### with fx
cv_f_Tmin <- fxTWAPLS::cv.w(taxa,
                            modern_pollen$Tmin,
                            nPLS = 5,
                            fxTWAPLS::WAPLS.w,
                            fxTWAPLS::WAPLS.predict.w,
                            usefx = TRUE,
                            fx = fx_Tmin,
                            cpus = 2, # Remove the following line
                            test_mode = test_mode)

## Random t-test
rand_Tmin <- fxTWAPLS::rand.t.test.w(cv_Tmin, n.perm = 999)
rand_f_Tmin <- fxTWAPLS::rand.t.test.w(cv_f_Tmin, n.perm = 999)
```

`sse.sample` *Function to calculate Sample Specific Errors*

Description

Function to calculate Sample Specific Errors

Usage

```
sse.sample(
  modern_taxa,
  modern_climate,
  fossil_taxa,
  trainfun,
  predictfun,
  nboot,
  nPLS,
  nsig,
  usefx,
  fx,
  cpus = 4,
  seed = NULL,
```

```
  test_mode = FALSE,
  test_it = 5
)
```

Arguments

modern_taxa	the modern taxa abundance data, each row represents a sampling site, each column represents a taxon.
modern_climate	the modern climate value at each sampling site
fossil_taxa	fossil taxa abundance data to reconstruct past climates, each row represents a site to be reconstructed, each column represents a taxon.
trainfun	training function you want to use, either WAPLS.w or TWAPLS.w
predictfun	predict function you want to use: if trainfun is WAPLS.w, then this should be WAPLS.predict.w; if trainfun is TWAPLS.w, then this should be TWAPLS.predict.w
nboot	the number of bootstrap cycles you want to use
nPLS	the number of components to be extracted
nsig	the significant number of components to use to reconstruct past climates, this can be obtained from the cross-validation results.
usefx	boolean flag on whether or not use fx correction.
fx	the frequency of the climate value for fx correction: if usefx is FALSE, this should be NA; otherwise, this should be obtained from the fx function.
cpus	number of CPUs for simultaneous iterations to execute, check parallel::detectCores() for available CPUs on your machine.
seed	seed for reproducibility
test_mode	boolean flag to execute the function with a limited number of iterations, test_it, for testing purposes only.
test_it	number of iterations to use in the test mode

Value

the bootstrapped standard error for each site

See Also

fx, TWAPLS.w, TWAPLS.predict.w, WAPLS.w, and WAPLS.predict.w

Examples

```
# Load modern pollen data
modern_pollen <- read.csv(system.file("extdata",
                                      "Modern_Pollen_gdd_alpha_Tmin.csv",
                                      package = "fxTWAPLS",
                                      mustWork = TRUE))

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "Abies")
```

```
taxaColMax <- which(colnames(modern_pollen) == "Zygophyllaceae")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Get the frequency of each climate variable fx
fx_Tmin <- fxTWAPLS::fx(modern_pollen$Tmin, bin = 0.02)
fx_gdd <- fxTWAPLS::fx(modern_pollen$gdd, bin = 20)
fx_alpha <- fxTWAPLS::fx(modern_pollen$alpha, bin = 0.002)

# Load reconstruction data
Holocene <- read.csv(system.file("extdata",
                                 "Holocene.csv",
                                 package = "fxTWAPLS",
                                 mustWork = TRUE),
                     row.names = 1)
taxaColMin <- which(colnames(Holocene) == "Abies")
taxaColMax <- which(colnames(Holocene) == "Zygophyllaceae")
core <- Holocene[, taxaColMin:taxaColMax]

# MTCO
## fx
fit_Tmin <- fxTWAPLS::WAPLS.w(taxa, modern_pollen$Tmin, nPLS = 5)
## SSE
nboot <- 5 # Recommended 1000
### without fx
sse_Tmin_WAPLS <- fxTWAPLS::sse.sample(modern_taxa = taxa,
                                       modern_climate = modern_pollen$Tmin,
                                       fossil_taxa = core,
                                       trainfun = fxTWAPLS::WAPLS.w,
                                       predictfun = fxTWAPLS::WAPLS.predict.w,
                                       nboot = nboot,
                                       nPLS = 5,
                                       nsig = 3,
                                       usefx = FALSE,
                                       fx = NA,
                                       cpus = 2,
                                       seed = 1)
### with fx
sse_f_Tmin_WAPLS <- fxTWAPLS::sse.sample(modern_taxa = taxa,
                                         modern_climate = modern_pollen$Tmin,
                                         fossil_taxa = core,
                                         trainfun = fxTWAPLS::WAPLS.w,
                                         predictfun = fxTWAPLS::WAPLS.predict.w,
                                         nboot = nboot,
                                         nPLS = 5,
                                         nsig = 3,
                                         usefx = TRUE,
                                         fx = fx_Tmin,
                                         cpus = 2,
                                         seed = 1)
```

TWAPLS.predict.w *TWA-PLS predict function*

Description

TWA-PLS predict function

Usage

```
TWAPLS.predict.w(TWAPLSoutput, fossil_taxa)
```

Arguments

TWAPLSoutput	the output of the TWAPLS.w training function, either with or without fx correction
fossil_taxa	fossil taxa abundance data to reconstruct past climates, each row represents a site to be reconstructed, each column represents a taxon.

Value

a list of the reconstruction results. fit is the fitted value.

See Also

TWAPLS.w

Examples

```
# Load modern pollen data
modern_pollen <- read.csv(system.file("extdata",
                                      "Modern_Pollen_gdd_alpha_Tmin.csv",
                                      package = "fxTWAPLS",
                                      mustWork = TRUE))

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "Abies")
taxaColMax <- which(colnames(modern_pollen) == "Zygophyllaceae")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Load reconstruction data
Holocene <- read.csv(system.file("extdata",
                                 "Holocene.csv",
                                 package = "fxTWAPLS",
                                 mustWork = TRUE),
                     row.names = 1)
taxaColMin <- which(colnames(Holocene) == "Abies")
taxaColMax <- which(colnames(Holocene) == "Zygophyllaceae")
core <- Holocene[, taxaColMin:taxaColMax]
```

```
# Get the frequency of each climate variable fx
fx_Tmin <- fxTWAPLS::fx(modern_pollen$Tmin, bin = 0.02)
fx_gdd <- fxTWAPLS::fx(modern_pollen$gdd, bin = 20)
fx_alpha <- fxTWAPLS::fx(modern_pollen$alpha, bin = 0.002)

# MTCO
## Train
fit_t_Tmin <- fxTWAPLS::TWAPLS.w(modern_taxa = taxa,
                                 modern_climate = modern_pollen$Tmin,
                                 nPLS = 5)
fit_tf_Tmin <- fxTWAPLS::TWAPLS.w(modern_taxa = taxa,
                                  modern_climate = modern_pollen$Tmin,
                                  nPLS = 5,
                                  usefx = TRUE,
                                  fx = fx_Tmin)

## Predict
fossil_t_Tmin <- fxTWAPLS::TWAPLS.predict.w(fit_t_Tmin, core)
fossil_tf_Tmin <- fxTWAPLS::TWAPLS.predict.w(fit_tf_Tmin, core)
```

`TWAPLS.w` *TWA-PLS training function*

Description

TWA-PLS training function, which can perform `fx` correction

Usage

```
TWAPLS.w(modern_taxa, modern_climate, nPLS = 5, usefx = FALSE, fx = NA)
```

Arguments

`modern_taxa`	the modern taxa abundance data, each row represents a sampling site, each column represents a taxon.
`modern_climate`	the modern climate value at each sampling site
`nPLS`	the number of components to be extracted
`usefx`	boolean flag on whether or not use `fx` correction.
`fx`	the frequency of the climate value for `fx` correction: if `usefx` is FALSE, this should be `NA`; otherwise, this should be obtained from the `fx` function.

Value

a list of the training results, which will be used by the predict function. fit is the fitted value of modern training result.

See Also

fx, TWAPLS.predict.w, and WAPLS.w

Examples

```
# Load modern pollen data
modern_pollen <- read.csv(system.file("extdata",
                                      "Modern_Pollen_gdd_alpha_Tmin.csv",
                                      package = "fxTWAPLS",
                                      mustWork = TRUE))

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "Abies")
taxaColMax <- which(colnames(modern_pollen) == "Zygophyllaceae")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Get the frequency of each climate variable fx
fx_Tmin <- fxTWAPLS::fx(modern_pollen$Tmin, bin = 0.02)
fx_gdd <- fxTWAPLS::fx(modern_pollen$gdd, bin = 20)
fx_alpha <- fxTWAPLS::fx(modern_pollen$alpha, bin = 0.002)

# MTCO
fit_t_Tmin <- fxTWAPLS::TWAPLS.w(taxa, modern_pollen$Tmin, nPLS = 5)
fit_tf_Tmin <- fxTWAPLS::TWAPLS.w(taxa,
                                  modern_pollen$Tmin,
                                  nPLS = 5,
                                  usefx = TRUE,
                                  fx = fx_Tmin)
```

WAPLS.predict.w *WA-PLS predict function*

Description

WA-PLS predict function

Usage

```
WAPLS.predict.w(WAPLSoutput, fossil_taxa)
```

Arguments

WAPLSoutput	the output of the WAPLS.w training function, either with or without fx correction
fossil_taxa	fossil taxa abundance data to reconstruct past climates, each row represents a site to be reconstructed, each column represents a taxon.

Value

a list of the reconstruction results. fit is the fitted value.

See Also

WAPLS.w

Examples

```
# Load modern pollen data
modern_pollen <- read.csv(system.file("extdata",
                                      "Modern_Pollen_gdd_alpha_Tmin.csv",
                                      package = "fxTWAPLS",
                                      mustWork = TRUE))

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "Abies")
taxaColMax <- which(colnames(modern_pollen) == "Zygophyllaceae")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Load reconstruction data
Holocene <- read.csv(system.file("extdata",
                                 "Holocene.csv",
                                 package = "fxTWAPLS",
                                 mustWork = TRUE),
                     row.names = 1)
taxaColMin <- which(colnames(Holocene) == "Abies")
taxaColMax <- which(colnames(Holocene) == "Zygophyllaceae")
core <- Holocene[, taxaColMin:taxaColMax]

# Get the frequency of each climate variable fx
fx_Tmin <- fxTWAPLS::fx(modern_pollen$Tmin, bin = 0.02)
fx_gdd <- fxTWAPLS::fx(modern_pollen$gdd, bin = 20)
fx_alpha <- fxTWAPLS::fx(modern_pollen$alpha, bin = 0.002)

# MTCO
## Train
fit_Tmin <- fxTWAPLS::WAPLS.w(modern_taxa = taxa,
                              modern_climate = modern_pollen$Tmin,
                              nPLS = 5)
fit_f_Tmin <- fxTWAPLS::WAPLS.w(modern_taxa = taxa,
                                modern_climate = modern_pollen$Tmin,
                                nPLS = 5,
                                usefx = TRUE,
                                fx = fx_Tmin)
## Predict
fossil_Tmin <- fxTWAPLS::WAPLS.predict.w(fit_Tmin, core)
fossil_f_Tmin <- fxTWAPLS::WAPLS.predict.w(fit_f_Tmin, core)
```

WAPLS.w *WA-PLS training function*

Description

WA-PLS training function, which can perform fx correction

Usage

```
WAPLS.w(modern_taxa, modern_climate, nPLS = 5, usefx = FALSE, fx = NA)
```

Arguments

modern_taxa	the modern taxa abundance data, each row represents a sampling site, each column represents a taxon.
modern_climate	the modern climate value at each sampling site
nPLS	the number of components to be extracted
usefx	boolean flag on whether or not use fx correction.
fx	the frequency of the climate value for fx correction: if usefx is FALSE, this should be NA; otherwise, this should be obtained from the fx function.

Value

a list of the training results, which will be used by the predict function. fit is the fitted value of modern training result.

See Also

fx, TWAPLS.w, and WAPLS.predict.w

Examples

```
# Load modern pollen data
modern_pollen <- read.csv(system.file("extdata",
                                      "Modern_Pollen_gdd_alpha_Tmin.csv",
                                      package = "fxTWAPLS",
                                      mustWork = TRUE))

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "Abies")
taxaColMax <- which(colnames(modern_pollen) == "Zygophyllaceae")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Get the frequency of each climate variable fx
fx_Tmin <- fxTWAPLS::fx(modern_pollen$Tmin, bin = 0.02)
fx_gdd <- fxTWAPLS::fx(modern_pollen$gdd, bin = 20)
fx_alpha <- fxTWAPLS::fx(modern_pollen$alpha, bin = 0.002)

# MTCO
fit_Tmin <- fxTWAPLS::WAPLS.w(taxa, modern_pollen$Tmin, nPLS = 5)
fit_f_Tmin <- fxTWAPLS::WAPLS.w(taxa,
                                modern_pollen$Tmin,
                                nPLS = 5,
                                usefx = TRUE,
```

```
fx = fx_Tmin)
```

Index