

# Package ‘fxTWAPLS’

November 25, 2022

**Title** An Improved Version of WA-PLS

**Version** 0.1.2

**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

**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** doFuture, foreach, future, geosphere, ggplot2, JOPS, MASS, parallel, progressr

**Suggests** lintr (>= 3.0.0), magrittr, progress, scales, spelling, styler, tictoc

**Depends** R (>= 3.6)

**RoxygenNote** 7.2.2

**Language** en-GB

**NeedsCompilation** no

**Author** Mengmeng Liu [aut] (<<https://orcid.org/0000-0001-6250-0148>>),  
Iain Colin Prentice [aut] (<<https://orcid.org/0000-0002-1296-6764>>),  
Cajo J. F. ter Braak [aut] (<<https://orcid.org/0000-0002-0414-8745>>),  
Sandy P. Harrison [aut] (<<https://orcid.org/0000-0001-5687-1903>>),  
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@outlook.com>

**Repository** CRAN

**Date/Publication** 2022-11-25 16:40:01 UTC

## R topics documented:

cv.pr.w . . . . .	2
cv.w . . . . .	4
fx . . . . .	6
fx_pspline . . . . .	7
get_distance . . . . .	8
get_pseudo . . . . .	9
pb . . . . .	11
plot_residuals . . . . .	12
plot_train . . . . .	13
rand.t.test.w . . . . .	14
sse.sample . . . . .	15
TWAPLS.predict.w . . . . .	17
TWAPLS.w . . . . .	19
TWAPLS.w2 . . . . .	21
WAPLS.predict.w . . . . .	23
WAPLS.w . . . . .	24
WAPLS.w2 . . . . .	26
<b>Index</b>	<b>28</b>

---

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_method = "bin",
  bin = 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 <a href="#">WAPLS.w</a> or <a href="#">TWAPLS.w</a> .
predictfun	Predict function you want to use: if trainfun is <a href="#">WAPLS.w</a> , then this should be <a href="#">WAPLS.predict.w</a> ; if trainfun is <a href="#">TWAPLS.w</a> , then this should be <a href="#">TWAPLS.predict.w</a> .
pseudo	The geographically and climatically close sites to each test site, obtained from <a href="#">get_pseudo</a> function.
usefx	Boolean flag on whether or not use fx correction.
fx_method	Binned or p-spline smoothed fx correction: if usefx = FALSE, this should be NA; otherwise, fx function will be used when choosing "bin"; <a href="#">fx_p spline</a> function will be used when choosing "pspline".
bin	Binwidth to get fx, needed for both binned and p-splined method. if usefx = FALSE, this should be NA;
cpus	Number of CPUs for simultaneous iterations to execute, check <code>parallel::detectCores()</code> 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

```

## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "taxa0")
taxaColMax <- which(colnames(modern_pollen) == "taxaN")
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
)

cv_pr_tf_Tmin2 <- fxTWAPLS::cv.pr.w(
  taxa,
  modern_pollen$Tmin,
  nPLS = 5,
  fxTWAPLS::TWAPLS.w2,
  fxTWAPLS::TWAPLS.predict.w,
  pseudo_Tmin,
  usefx = TRUE,
  fx_method = "bin",
  bin = 0.02,
  cpus = 2, # Remove the following line
  test_mode = test_mode
)

# Run with progress bar
`%>%` <- magrittr::`%>%`
cv_pr_tf_Tmin2 <- fxTWAPLS::cv.pr.w(
  taxa,
  modern_pollen$Tmin,
  nPLS = 5,
  fxTWAPLS::TWAPLS.w2,
  fxTWAPLS::TWAPLS.predict.w,
  pseudo_Tmin,
  usefx = TRUE,
  fx_method = "bin",
  bin = 0.02,
  cpus = 2, # Remove the following line
  test_mode = test_mode
) %>%
  fxTWAPLS::pb()

## End(Not run)

```

## 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_method = "bin",
  bin = 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 <a href="#">WAPLS.w</a> or <a href="#">TWAPLS.w</a> .
predictfun	Predict function you want to use: if trainfun is <a href="#">WAPLS.w</a> , then this should be <a href="#">WAPLS.predict.w</a> ; if trainfun is <a href="#">TWAPLS.w</a> , then this should be <a href="#">TWAPLS.predict.w</a> .
usefx	Boolean flag on whether or not use fx correction.
fx_method	Binned or p-spline smoothed fx correction: if usefx = FALSE, this should be NA; otherwise, fx function will be used when choosing "bin"; <a href="#">fx_p spline</a> function will be used when choosing "pspline".
bin	Binwidth to get fx, needed for both binned and p-splined method. if usefx = FALSE, this should be NA;
cpus	Number of CPUs for simultaneous iterations to execute, check <code>parallel::detectCores()</code> 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

```
## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "taxa0")
taxaColMax <- which(colnames(modern_pollen) == "taxaN")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

## LOOCV
test_mode <- TRUE # It should be set to FALSE before running
cv_tf_Tmin2 <- fxTWAPLS::cv.w(
  taxa,
  modern_pollen$Tmin,
  nPLS = 5,
  fxTWAPLS::TWAPLS.w2,
  fxTWAPLS::TWAPLS.predict.w,
  usefx = TRUE,
  fx_method = "bin",
  bin = 0.02,
  cpus = 2, # Remove the following line
  test_mode = test_mode
)

# Run with progress bar
`%>%` <- magrittr::`%>%`
cv_tf_Tmin2 <- fxTWAPLS::cv.w(
  taxa,
  modern_pollen$Tmin,
  nPLS = 5,
  fxTWAPLS::TWAPLS.w2,
  fxTWAPLS::TWAPLS.predict.w,
  usefx = TRUE,
  fx_method = "bin",
  bin = 0.02,
  cpus = 2, # Remove the following line
  test_mode = test_mode
) %>% fxTWAPLS::pb()

## End(Not run)
```

---

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, show_plot = FALSE)
```

**Arguments**

x	Numeric vector with the modern climate values.
bin	Binwidth to get the frequency of the modern climate values.
show_plot	Boolean flag to show a plot of $fx \sim x$ .

**Value**

Numeric vector with the frequency of the modern climate values.

**See Also**

[cv.w](#), [cv.pr.w](#), and [sse.sample](#)

**Examples**

```
## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

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

## End(Not run)
```

---

fx\_pspline

*Get frequency of the climate value with p-spline smoothing*


---

**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_pspline(x, bin, show_plot = FALSE)
```

**Arguments**

x	Numeric vector with the modern climate values.
bin	Binwidth to get the frequency of the modern climate values, the curve will be p-spline smoothed later
show_plot	Boolean flag to show a plot of $fx \sim x$ .

**Value**

Numeric vector with the frequency of the modern climate values.

**See Also**

[cv.w](#), [cv.pr.w](#), and [sse.sample](#)

**Examples**

```
## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

# Get the frequency of each climate variable fx
fx_pspline_Tmin <- fxTWAPLS::fx_pspline(
  modern_pollen$Tmin,
  bin = 0.02,
  show_plot = TRUE
)
fx_pspline_gdd <- fxTWAPLS::fx_pspline(
  modern_pollen$gdd,
  bin = 20,
  show_plot = TRUE
)
fx_pspline_alpha <- fxTWAPLS::fx_pspline(
  modern_pollen$alpha,
  bin = 0.002,
  show_plot = TRUE
)

## End(Not run)
```

---

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 <code>parallel::detectCores()</code> for available CPUs on your machine.
test_mode	Boolean flag to execute the function with a limited number of iterations, <code>test_it</code> , 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**

```
## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

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
)
# Run with progress bar
`%>%` <- magrittr::`%>%`
dist <- fxTWAPLS::get_distance(
  point,
  cpus = 2, # Remove the following line
  test_mode = test_mode
) %>%
  fxTWAPLS::pb()

## End(Not run)
```

---

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**

<code>dist</code>	Distance matrix which contains the distance from other sites.
<code>x</code>	The modern climate values.
<code>cpus</code>	Number of CPUs for simultaneous iterations to execute, check <code>parallel::detectCores()</code> for available CPUs on your machine.
<code>test_mode</code>	Boolean flag to execute the function with a limited number of iterations, <code>test_it</code> , for testing purposes only.
<code>test_it</code>	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**

```
## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

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$Tmin,
  cpus = 2, # Remove the following line
  test_mode = test_mode
)
# Run with progress bar
`%>%` <- magrittr::`%>%`
pseudo_Tmin <- fxTWAPLS::get_pseudo(
  dist,
  modern_pollen$Tmin,
  cpus = 2, # Remove the following line
  test_mode = test_mode
) %>%
  fxTWAPLS::pb()
```

```
## End(Not run)
```

---

pb *Show progress bar*

---

## Description

Show progress bar

## Usage

```
pb(expr, ...)
```

## Arguments

expr	R expression.
...	Arguments passed on to <a href="#">progressr::with_progress</a>
cleanup	If TRUE, all progression handlers will be shutdown at the end regardless of the progression is complete or not.
delay_terminal	If TRUE, output and conditions that may end up in the terminal will be delayed.
delay_stdout	If TRUE, standard output is captured and relayed at the end just before any captured conditions are relayed.
delay_conditions	A character vector specifying <a href="#">base::condition</a> classes to be captured and relayed at the end after any captured standard output is relayed.
interrupts	Controls whether interrupts should be detected or not. If TRUE and an interrupt is signaled, progress handlers are asked to report on the current amount of progress when the evaluation was terminated by the interrupt, e.g. when a user pressed Ctrl-C in an interactive session, or a batch process was interrupted because it ran out of time.
interval	(numeric) The minimum time (in seconds) between successive progression updates from handlers.
enable	(logical) If FALSE, then progress is not reported. The default is to report progress in interactive mode but not batch mode. See below for more details.

## Value

Return data from the function called.

---

plot_residuals	<i>Plot the residuals</i>
----------------	---------------------------

---

### 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

```
## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "taxa0")
taxaColMax <- which(colnames(modern_pollen) == "taxaN")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

fit_tf_Tmin2 <- fxTWAPLS::TWAPLS.w2(
  taxa,
  modern_pollen$Tmin,
  nPLS = 5,
  usefx = TRUE,
  fx_method = "bin",
  bin = 0.02
)

nsig <- 3 # This should be got from the random t-test of the cross validation
fxTWAPLS::plot_residuals(fit_tf_Tmin2, nsig)
```

```
## End(Not run)
```

---

plot_train	<i>Plot the training results</i>
------------	----------------------------------

---

### 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

```
## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "taxa0")
taxaColMax <- which(colnames(modern_pollen) == "taxaN")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

fit_tf_Tmin2 <- fxTWAPLS::TWAPLS.w2(
  taxa,
  modern_pollen$Tmin,
  nPLS = 5,
  usefx = TRUE,
  fx_method = "bin",
  bin = 0.02
```

```

)

nsig <- 3 # This should be got from the random t-test of the cross validation
fxTWAPLS::plot_train(fit_tf_Tmin2, nsig)

## End(Not run)

```

---

 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 <code>cv.w</code> or <code>cv.pr.w</code> .
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. Each component is described below:

R2 the coefficient of determination (the larger, the better the fit).

Avg.Bias average bias.

Max.Bias maximum bias.

Min.Bias minimum bias.

RMSEP root-mean-square error of prediction (the smaller, the better the fit).

delta.RMSEP the percent change of RMSEP using the current number of components than using one component less.

p assesses whether using the current number of components is significantly different from using one component less, which is used to choose the last significant number of components to avoid over-fitting.

- The degree of overall compression is assessed by doing linear regression to the cross-validation result and the observed climate values.

- Compre.b0: the intercept.
- Compre.b1: the slope (the closer to 1, the less the overall compression).
- Compre.b0.se: the standard error of the intercept.
- Compre.b1.se: the standard error of the slope.

**See Also**

[cv.w](#) and [cv.pr.w](#)

**Examples**

```
## Not run:  
  
## Random t-test  
rand_pr_tf_Tmin2 <- fxTWAPLS::rand.t.test.w(cv_pr_tf_Tmin2, n.perm = 999)  
  
# note: choose the last significant number of components based on the p-value,  
# see details at Liu Mengmeng, Prentice Iain Colin, ter Braak Cajo J. F.,  
# Harrison Sandy P.. 2020 An improved statistical approach for reconstructing  
# past climates from biotic assemblages. Proc. R. Soc. A. 476: 20200346.  
# <https://doi.org/10.1098/rspa.2020.0346>  
  
## End(Not run)
```

---

sse.sample

*Calculate Sample Specific Errors*

---

**Description**

Calculate Sample Specific Errors

**Usage**

```
sse.sample(  
  modern_taxa,  
  modern_climate,  
  fossil_taxa,  
  trainfun,  
  predictfun,  
  nboot,  
  nPLS,  
  nsig,  
  usefx = FALSE,  
  fx_method = "bin",  
  bin = NA,  
  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 <a href="#">WAPLS.w</a> or <a href="#">TWAPLS.w</a> .
predictfun	Predict function you want to use: if trainfun is <a href="#">WAPLS.w</a> , then this should be <a href="#">WAPLS.predict.w</a> ; if trainfun is <a href="#">TWAPLS.w</a> , then this should be <a href="#">TWAPLS.predict.w</a> .
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_method	Binned or p-spline smoothed fx correction: if usefx = FALSE, this should be NA; otherwise, <a href="#">fx</a> function will be used when choosing "bin"; <a href="#">fx_p spline</a> function will be used when choosing "pspline".
bin	Binwidth to get fx, needed for both binned and p-splined method. if usefx = FALSE, this should be NA;
cpus	Number of CPUs for simultaneous iterations to execute, check <code>parallel::detectCores()</code> for available CPUs on your machine.
seed	Seed for reproducibility.
test_mode	Boolean flag to execute the function with a limited number of iterations, <code>test_it</code> , 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**

```
## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "taxa0")
taxaColMax <- which(colnames(modern_pollen) == "taxaN")
taxa <- modern_pollen[, taxaColMin:taxaColMax]
```



```

# Load reconstruction data
Holocene <- read.csv("/path/to/Holocene.csv")
taxaColMin <- which(colnames(Holocene) == "taxa0")
taxaColMax <- which(colnames(Holocene) == "taxaN")
core <- Holocene[, taxaColMin:taxaColMax]

## SSE
nboot <- 5 # Recommended 1000
nsig <- 3 # This should be got from the random t-test of the cross validation

sse_tf_Tmin2 <- fxTWAPLS::sse.sample(
  modern_taxa = taxa,
  modern_climate = modern_pollen$Tmin,
  fossil_taxa = core,
  trainfun = fxTWAPLS::TWAPLS.w2,
  predictfun = fxTWAPLS::TWAPLS.predict.w,
  nboot = nboot,
  nPLS = 5,
  nsig = nsig,
  usefx = TRUE,
  fx_method = "bin",
  bin = 0.02,
  cpus = 2,
  seed = 1
)

# Run with progress bar
`%>%` <- magrittr::`%>%`
sse_tf_Tmin2 <- fxTWAPLS::sse.sample(
  modern_taxa = taxa,
  modern_climate = modern_pollen$Tmin,
  fossil_taxa = core,
  trainfun = fxTWAPLS::TWAPLS.w2,
  predictfun = fxTWAPLS::TWAPLS.predict.w,
  nboot = nboot,
  nPLS = 5,
  nsig = nsig,
  usefx = TRUE,
  fx_method = "bin",
  bin = 0.02,
  cpus = 2,
  seed = 1
) %>% fxTWAPLS::pb()

## End(Not run)

```

**Description**

TWA-PLS predict function

**Usage**

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

**Arguments**

TWAPLSoutput	The output of the <a href="#">TWAPLS.w</a> 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. Each element in the list is described below:

fit the fitted values using each number of components.

nPLS the total number of components extracted.

**See Also**

[TWAPLS.w](#)

**Examples**

```
## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "taxa0")
taxaColMax <- which(colnames(modern_pollen) == "taxaN")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Load reconstruction data
Holocene <- read.csv("/path/to/Holocene.csv")
taxaColMin <- which(colnames(Holocene) == "taxa0")
taxaColMax <- which(colnames(Holocene) == "taxaN")
core <- Holocene[, taxaColMin:taxaColMax]

## Train
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_method = "bin",
  bin = 0.02
```

```

)
fit_t_Tmin2 <- fxTWAPLS::TWAPLS.w2(taxa, modern_pollen$Tmin, nPLS = 5)
fit_tf_Tmin2 <- fxTWAPLS::TWAPLS.w2(
  taxa,
  modern_pollen$Tmin,
  nPLS = 5,
  usefx = TRUE,
  fx_method = "bin",
  bin = 0.02
)

## Predict
fossil_t_Tmin <- fxTWAPLS::TWAPLS.predict.w(fit_t_Tmin, core)
fossil_tf_Tmin <- fxTWAPLS::TWAPLS.predict.w(fit_tf_Tmin, core)
fossil_t_Tmin2 <- fxTWAPLS::TWAPLS.predict.w(fit_t_Tmin2, core)
fossil_tf_Tmin2 <- fxTWAPLS::TWAPLS.predict.w(fit_tf_Tmin2, core)

## End(Not run)

```

---

TWAPLS.w

*TWA-PLS training function*


---

## Description

TWA-PLS training function, which can perform fx correction.  $1/fx^2$  correction will be applied at step 7.

## Usage

```

TWAPLS.w(
  modern_taxa,
  modern_climate,
  nPLS = 5,
  usefx = FALSE,
  fx_method = "bin",
  bin = 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.

<code>fx_method</code>	Binned or p-spline smoothed <code>fx</code> correction: if <code>usefx = FALSE</code> , this should be NA; otherwise, <code>fx</code> function will be used when choosing "bin"; <code>fx_p spline</code> function will be used when choosing "pspline".
<code>bin</code>	Binwidth to get <code>fx</code> , needed for both binned and p-splined method. if <code>usefx = FALSE</code> , this should be NA;

### Value

A list of the training results, which will be used by the predict function. Each element in the list is described below:

`fit` the fitted values using each number of components.  
`x` the observed modern climate values.  
`taxon_name` the name of each taxon.  
`optimum` the updated taxon optimum  
`comp` each component extracted (will be used in step 7 regression).  
`u` taxon optimum for each component (step 2).  
`t` taxon tolerance for each component (step 2).  
`z` a parameter used in standardization for each component (step 5).  
`s` a parameter used in standardization for each component (step 5).  
`orth` a list that stores orthogonalization parameters (step 4).  
`alpha` a list that stores regression coefficients (step 7).  
`meanx` mean value of the observed modern climate values.  
`nPLS` the total number of components extracted.

### See Also

[fx](#), [TWAPLS.predict.w](#), and [WAPLS.w](#)

### Examples

```
## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "taxa0")
taxaColMax <- which(colnames(modern_pollen) == "taxaN")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Training
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_method = "bin",
    bin = 0.02
  )

  ## End(Not run)

```

---

 TWAPLS.w2

*TWA-PLS training function v2*


---

### Description

TWA-PLS training function, which can perform fx correction. 1/fx correction will be applied at step 2 and step 7.

### Usage

```

TWAPLS.w2(
  modern_taxa,
  modern_climate,
  nPLS = 5,
  usefx = FALSE,
  fx_method = "bin",
  bin = NA
)

```

### Arguments

<code>modern_taxa</code>	The modern taxa abundance data, each row represents a sampling site, each column represents a taxon.
<code>modern_climate</code>	The modern climate value at each sampling site.
<code>nPLS</code>	The number of components to be extracted.
<code>usefx</code>	Boolean flag on whether or not use fx correction.
<code>fx_method</code>	Binned or p-spline smoothed fx correction: if <code>usefx = FALSE</code> , this should be NA; otherwise, <code>fx</code> function will be used when choosing "bin"; <code>fx_p spline</code> function will be used when choosing "pspline".
<code>bin</code>	Binwidth to get fx, needed for both binned and p-splined method. if <code>usefx = FALSE</code> , this should be NA;

### Value

A list of the training results, which will be used by the predict function. Each element in the list is described below:

`fit` the fitted values using each number of components.  
`x` the observed modern climate values.

taxon\_name the name of each taxon.  
 optimum the updated taxon optimum  
 comp each component extracted (will be used in step 7 regression).  
 u taxon optimum for each component (step 2).  
 t taxon tolerance for each component (step 2).  
 z a parameter used in standardization for each component (step 5).  
 s a parameter used in standardization for each component (step 5).  
 orth a list that stores orthogonalization parameters (step 4).  
 alpha a list that stores regression coefficients (step 7).  
 meanx mean value of the observed modern climate values.  
 nPLS the total number of components extracted.

### See Also

[fx](#), [TWAPLS.predict.w](#), and [WAPLS.w](#)

### Examples

```

## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "taxa0")
taxaColMax <- which(colnames(modern_pollen) == "taxaN")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Training
fit_t_Tmin2 <- fxTWAPLS::TWAPLS.w2(taxa, modern_pollen$Tmin, nPLS = 5)
fit_tf_Tmin2 <- fxTWAPLS::TWAPLS.w2(
  taxa,
  modern_pollen$Tmin,
  nPLS = 5,
  usefx = TRUE,
  fx_method = "bin",
  bin = 0.02
)

## End(Not run)

```

---

WAPLS.predict.w	<i>WA-PLS predict function</i>
-----------------	--------------------------------

---

**Description**

WA-PLS predict function

**Usage**

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

**Arguments**

WAPLSoutput	The output of the <a href="#">WAPLS.w</a> 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. Each element in the list is described below:

`fit` The fitted values using each number of components.

`nPLS` The total number of components extracted.

**See Also**

[WAPLS.w](#)

**Examples**

```
## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "taxa0")
taxaColMax <- which(colnames(modern_pollen) == "taxaN")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Load reconstruction data
Holocene <- read.csv("/path/to/Holocene.csv")
taxaColMin <- which(colnames(Holocene) == "taxa0")
taxaColMax <- which(colnames(Holocene) == "taxaN")
core <- Holocene[, taxaColMin:taxaColMax]

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

```

    taxa,
    modern_pollen$Tmin,
    nPLS = 5,
    usefx = TRUE,
    fx_method = "bin",
    bin = 0.02
  )
fit_Tmin2 <- fxTWAPLS::WAPLS.w2(taxa, modern_pollen$Tmin, nPLS = 5)
fit_f_Tmin2 <- fxTWAPLS::WAPLS.w2(
  taxa,
  modern_pollen$Tmin,
  nPLS = 5,
  usefx = TRUE,
  fx_method = "bin",
  bin = 0.02
)
## Predict
fossil_Tmin <- fxTWAPLS::WAPLS.predict.w(fit_Tmin, core)
fossil_f_Tmin <- fxTWAPLS::WAPLS.predict.w(fit_f_Tmin, core)
fossil_Tmin2 <- fxTWAPLS::WAPLS.predict.w(fit_Tmin2, core)
fossil_f_Tmin2 <- fxTWAPLS::WAPLS.predict.w(fit_f_Tmin2, core)

## End(Not run)

```

---

WAPLS.w

*WA-PLS training function*


---

### Description

WA-PLS training function, which can perform fx correction.  $1/fx^2$  correction will be applied at step 7.

### Usage

```

WAPLS.w(
  modern_taxa,
  modern_climate,
  nPLS = 5,
  usefx = FALSE,
  fx_method = "bin",
  bin = 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_method	Binned or p-spline smoothed fx correction: if usefx = FALSE, this should be NA; otherwise, fx function will be used when choosing "bin"; <a href="#">fx_pspline</a> function will be used when choosing "pspline".
bin	Binwidth to get fx, needed for both binned and p-splined method. if usefx = FALSE, this should be NA;

### Value

A list of the training results, which will be used by the predict function. Each element in the list is described below:

`fit` the fitted values using each number of components.  
`x` the observed modern climate values.  
`taxon_name` the name of each taxon.  
`optimum` the updated taxon optimum ( $u^*$  in the WA-PLS paper).  
`comp` each component extracted (will be used in step 7 regression).  
`u` taxon optimum for each component (step 2).  
`z` a parameter used in standardization for each component (step 5).  
`s` a parameter used in standardization for each component (step 5).  
`orth` a list that stores orthogonalization parameters (step 4).  
`alpha` a list that stores regression coefficients (step 7).  
`meanx` mean value of the observed modern climate values.  
`nPLS` the total number of components extracted.

### See Also

[fx](#), [TWAPLS.w](#), and [WAPLS.predict.w](#)

### Examples

```
## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "taxa0")
taxaColMax <- which(colnames(modern_pollen) == "taxaN")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Training
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_method = "bin",
bin = 0.02
)

## End(Not run)

```

---

WAPLS.w2

*WA-PLS training function v2*


---

### Description

WA-PLS training function, which can perform fx correction. 1/fx correction will be applied at step 2 and step 7.

### Usage

```

WAPLS.w2(
  modern_taxa,
  modern_climate,
  nPLS = 5,
  usefx = FALSE,
  fx_method = "bin",
  bin = NA
)

```

### Arguments

<code>modern_taxa</code>	The modern taxa abundance data, each row represents a sampling site, each column represents a taxon.
<code>modern_climate</code>	The modern climate value at each sampling site.
<code>nPLS</code>	The number of components to be extracted.
<code>usefx</code>	Boolean flag on whether or not use fx correction.
<code>fx_method</code>	Binned or p-spline smoothed fx correction: if <code>usefx = FALSE</code> , this should be NA; otherwise, <code>fx</code> function will be used when choosing "bin"; <code>fx_p spline</code> function will be used when choosing "pspline".
<code>bin</code>	Binwidth to get fx, needed for both binned and p-splined method. if <code>usefx = FALSE</code> , this should be NA;

### Value

A list of the training results, which will be used by the predict function. Each element in the list is described below:

`fit` the fitted values using each number of components.

x the observed modern climate values.  
 taxon\_name the name of each taxon.  
 optimum the updated taxon optimum ( $u^*$  in the WA-PLS paper).  
 comp each component extracted (will be used in step 7 regression).  
 u taxon optimum for each component (step 2).  
 z a parameter used in standardization for each component (step 5).  
 s a parameter used in standardization for each component (step 5).  
 orth a list that stores orthogonalization parameters (step 4).  
 alpha a list that stores regression coefficients (step 7).  
 meanx mean value of the observed modern climate values.  
 nPLS the total number of components extracted.

### See Also

[fx](#), [TWAPLS.w](#), and [WAPLS.predict.w](#)

### Examples

```

## Not run:
# Load modern pollen data
modern_pollen <- read.csv("/path/to/modern_pollen.csv")

# Extract taxa
taxaColMin <- which(colnames(modern_pollen) == "taxa0")
taxaColMax <- which(colnames(modern_pollen) == "taxaN")
taxa <- modern_pollen[, taxaColMin:taxaColMax]

# Training
fit_Tmin2 <- fxTWAPLS::WAPLS.w2(taxa, modern_pollen$Tmin, nPLS = 5)
fit_f_Tmin2 <- fxTWAPLS::WAPLS.w2(
  taxa,
  modern_pollen$Tmin,
  nPLS = 5,
  usefx = TRUE,
  fx_method = "bin",
  bin = 0.02
)

## End(Not run)

```

# Index

`base::condition`, [11](#)

`cv.pr.w`, [2](#), [7–9](#), [14](#), [15](#)  
`cv.w`, [4](#), [7](#), [8](#), [14](#), [15](#)

`fx`, [3](#), [5](#), [6](#), [16](#), [20–22](#), [25–27](#)  
`fx_p spline`, [3](#), [5](#), [7](#), [16](#), [20](#), [21](#), [25](#), [26](#)

`get_distance`, [8](#), [10](#)  
`get_pseudo`, [3](#), [8](#), [9](#), [9](#)

`pb`, [11](#)  
`plot_residuals`, [12](#)  
`plot_train`, [13](#)  
`progressr::with_progress`, [11](#)

`rand.t.test.w`, [14](#)

`sse.sample`, [7](#), [8](#), [15](#)

`TWAPLS.predict.w`, [3](#), [5](#), [16](#), [17](#), [20](#), [22](#)  
`TWAPLS.w`, [3](#), [5](#), [12](#), [13](#), [16](#), [18](#), [19](#), [25](#), [27](#)  
`TWAPLS.w2`, [21](#)

`WAPLS.predict.w`, [3](#), [5](#), [16](#), [23](#), [25](#), [27](#)  
`WAPLS.w`, [3](#), [5](#), [12](#), [13](#), [16](#), [20](#), [22](#), [23](#), [24](#)  
`WAPLS.w2`, [26](#)