Package: tidyeof (via r-universe)

May 28, 2026

Type Package

Title Tidy Empirical Orthogonal Functions and Spatial Downscaling

Version 0.1.0

Description An R package for conducting empirical orthogonal function (EOF) analysis in the tidyverse framework. Functions to isolate modes of variability from spatiotemporal data, run various diagnostics, and use these patterns for spatial downscaling via canonical correlation analysis.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1)

Suggests testthat (>= 3.0.0), patchwork, furrr, lubridate, quarto

Config/testthat/edition 3

VignetteBuilder quarto

URL https://github.com/nick-gauthier/tidyEOF

BugReports https://github.com/nick-gauthier/tidyEOF/issues

Imports rlang, glue, cli, stars, cubelyr, dplyr, tidyr, sf, ggplot2, scico, purrr, units, MASS, irlba, tibble

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://flmnh-ai.r-universe.dev

Date/Publication 2026-02-25 21:56:49 UTC

RemoteUrl https://github.com/nick-gauthier/tidyeof

RemoteRef HEAD

RemoteSha dddb4629a8eb7cffdcbd14550cf975aec385a1e5

Contents

`[.patterns` *Extract subset of PCs from a patterns object*

Description

Extract subset of PCs from a patterns object

Usage

```
## S3 method for class 'patterns'
x[i, ...]
```

Arguments

x	A patterns object
i	Index or names of PCs to keep
...	Additional arguments (unused)

Value

A patterns object with only selected PCs

---

area_weights *Compute area-based weights for spatial data*

Description

Calculates area weights for spatial data using st_area(). Works uniformly for raster grids, irregular geometries, and different coordinate systems.

Usage

```
area_weights(dat)
```

Arguments

dat	A stars object with spatial dimensions

Value

Numeric weights (sqrt of normalized areas)

---

common_patterns *Compute Common EOF Patterns from Multiple Sources*

Description

Performs joint PCA on multiple datasets that share a spatial domain. Each dataset is anomalized with its own climatology, then concatenated along time for joint PCA. The resulting shared spatial patterns get source-specific amplitudes and climatologies, enabling CCA coupling and cross-source prediction.

Usage

```
common_patterns(
  datasets,
  k = 4,
  scale = TRUE,
  rotate = FALSE,
  monthly = FALSE,
  weight = TRUE,
  irlba_threshold = 5e+05
)
```

Arguments

datasets	Named list of stars objects sharing the same spatial grid. Names become the source identifiers used for extraction.
k	Number of EOF modes to retain
scale	Logical, whether to scale anomalies by standard deviation (default TRUE)
rotate	Logical, whether to apply varimax rotation (default FALSE)
monthly	Logical, whether to use monthly climatology (default FALSE)
weight	Logical, whether to apply area weighting (default TRUE)
irlba_threshold	Minimum data elements to trigger IRLBA (default 500000)

Value

A ‘common_patterns‘ S3 object. Source-specific patterns are extracted with ‘$‘ or ‘[[‘ using source names (e.g., ‘cpat$era‘). Each extracted element is a standard ‘patterns‘ object with shared EOFs but source-specific climatology and amplitudes.

Examples

```
## Not run:
cpat <- common_patterns(
  list(era = era_coarse, phyda = phyda_coarse),
  k = 11, scale = TRUE
)

# Extract source-specific patterns
cpat$era     # patterns object with ERA climatology + amplitudes
cpat$phyda   # patterns object with PHYDA climatology + amplitudes

# Use with existing couple/predict workflow
fine_pat <- patterns(era_fine, k = 13)
coupled <- couple(cpat$era, fine_pat, k = 9)
predict(coupled, era_new)

# Cross-source prediction
predict(coupled, phyda_new, predictor_patterns = cpat$phyda)
```

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

`couple` *Couple Pattern Relationships Using CCA*

Description

This function couples predictor and response patterns using Canonical Correlation Analysis (CCA) as the primary method. CCA finds linear combinations of predictor and response patterns that maximize correlation between them.

Usage

```
couple(
  predictor_patterns,
  response_patterns,
  k = NULL,
  method = "cca",
  center = FALSE,
  validate = TRUE
)
```

Arguments

`predictor_patterns`	A patterns object containing predictor patterns (e.g., from patterns())
`response_patterns`	A patterns object containing response patterns (e.g., from patterns())
`k`	Number of CCA modes to retain. If NULL, uses min(ncol(predictor), ncol(response))
`method`	Coupling method. Currently only "cca" is supported
`center`	Logical, whether to center the data before CCA (default: FALSE)
`validate`	Logical, whether to validate input patterns compatibility

Value

A coupled_patterns object containing:

`cca`	The CCA results from cancor()
`predictor_patterns`	The original predictor patterns
`response_patterns`	The original response patterns
`k`	Number of CCA modes retained
`method`	Coupling method used

Examples

```
## Not run:
# Get patterns from your data
pred_patterns <- patterns(predictor_data, k = 5)
resp_patterns <- patterns(response_data, k = 5)

# Couple the patterns
coupled <- couple(pred_patterns, resp_patterns, k = 3)

# Make predictions
predictions <- predict(coupled, new_predictor_data)

## End(Not run)
```

---

eigen_test *Test EOF significance using modified Rule N*

Description

Tests whether the k-th eigenvalue is significantly different from noise using a modified Rule N approach based on Tracy-Widom distribution.

Usage

```
eigen_test(lambdas, k, M, n, p = 0.05)
```

Arguments

lambdas	Vector of eigenvalues from PCA
k	Index of eigenvalue to test
M	Number of spatial points (grid cells)
n	Number of time steps
p	Significance level (default 0.05)

Details

Based on the gamma approximation to the Tracy-Widom distribution described in Cheng & Wallace (1993) and implemented following Overland & Preisendorfer (1982). Constants (shape = 46.4, scale factor = 0.186, location = 9.85) derive from fitting the gamma CDF to the Tracy-Widom Type 1 distribution.

Value

Logical, TRUE if eigenvalue is significant at level p

get_anomalies *Calculate anomalies from a climatological mean*

Description

Calculate anomalies from a climatological mean

Usage

```
get_anomalies(dat, clim = NULL, scale = FALSE, monthly = FALSE)
```

Arguments

dat	A stars object with dimensions (x, y, time)
clim	Optional climatology from get_climatology(). If NULL, computed internally
scale	Logical. If TRUE, divide by standard deviation
monthly	Logical. If TRUE, compute monthly anomalies

Value

A stars object with anomalies

get_canonical_correlations *Get Canonical Correlations from Coupled Patterns*

Description

Extract the canonical correlations and related statistics

Usage

```
get_canonical_correlations(object, k = NULL)
```

Arguments

object	A coupled_patterns object
k	Number of modes to return (default: all available)

Value

Data frame with canonical correlation statistics

get_canonical_patterns *Get Canonical Spatial Patterns from Coupled Patterns*

Description

Computes the spatial patterns corresponding to each canonical mode by taking linear combinations of the original EOFs weighted by CCA coefficients. These are the spatial patterns that, when projected onto the data, yield the canonical variates.

Usage

```
get_canonical_patterns(object, type = c("predictor", "response"), k = NULL)
```

Arguments

object	A coupled_patterns object
type	Either "predictor" or "response"
k	Number of canonical modes to extract (default: all available)

Value

A stars object with canonical spatial patterns (dimension "CV" instead of "PC")

Examples

```
## Not run:
coupled <- couple(pred_patterns, resp_patterns, k = 3)

# Get canonical patterns for response side
resp_canonical <- get_canonical_patterns(coupled, type = "response")
plot(resp_canonical)

# Compare to original EOFs
plot(coupled$response_patterns$eofs)

## End(Not run)
```

get_canonical_variables *Get Canonical Variables from Coupled Patterns*

Description

Extract canonical variables from either predictor or response patterns

Usage

```
get_canonical_variables(
  object,
  data,
  type = c("predictor", "response"),
  k = NULL
)
```

Arguments

object	A coupled_patterns object
data	Original data (patterns object or amplitudes tibble)
type	Either "predictor" or "response"
k	Number of canonical modes to extract

Value

Tibble with canonical variables

get_climatology *Calculate climatological mean and standard deviation for spatial data*

Description

Computes climatological statistics (mean and standard deviation) for a spatial field, either annually or monthly. Preserves spatial dimensions and units from the input data.

Usage

```
get_climatology(dat, monthly = FALSE)
```

Arguments

dat	A stars object containing a spatial field with dimensions (x, y, time)
monthly	Logical. If TRUE, computes monthly climatology. If FALSE (default), computes statistics over the entire period.

Value

A list with two stars objects:

mean	Climatological mean with original spatial dimensions and units
sd	Climatological standard deviation with same structure

Examples

```
# Create sample data
library(stars)
times <- seq(as.Date("2000-01-01"), as.Date("2002-12-31"), by = "month")
x <- seq(0, 1, length.out = 10)
y <- seq(0, 1, length.out = 10)
dat <- stars::st_as_stars(array(rnorm(10*10*36), c(10, 10, 36))) %>%
  st_set_dimensions(1, values = x, name = "x") %>%
  st_set_dimensions(2, values = y, name = "y") %>%
  st_set_dimensions(3, values = times, name = "time")

# Calculate annual climatology
clim <- get_climatology(dat)

# Calculate monthly climatology
monthly_clim <- get_climatology(dat, monthly = TRUE)
```

---

get_correlation *Calculate teleconnections between a patterns object and another layer*

Description

Computes pixel-wise correlations between a spatiotemporal field and each PC amplitude time series. Checks for overlapping time steps between the raster field and the PC amplitudes.

Usage

```
get_correlation(dat, patterns, amplitudes = NULL)
```

Arguments

dat	A stars object with a time dimension
patterns	A patterns object from patterns()
amplitudes	Optional amplitudes tibble (defaults to patterns$amplitudes)

Value

A stars object with correlation values for each PC

get_fdr *Calculate FDR-corrected significance contours for teleconnections*

Description

Computes pixel-wise correlations with FDR correction and returns significance contour lines as sf polygons.

Usage

```
get_fdr(dat, patterns, fdr = 0.1, amplitudes = NULL)
```

Arguments

dat	A stars object with a time dimension
patterns	A patterns object from patterns()
fdr	False discovery rate threshold (default 0.1)
amplitudes	Optional amplitudes tibble (defaults to patterns$amplitudes)

Value

An sf object with significance contour polygons for each PC

patterns *Get EOFs and PCs from spatiotemporal data*

Description

This function performs Empirical Orthogonal Function (EOF) analysis on spatial-temporal data. For large datasets, it automatically uses IRLBA (Implicitly Restarted Lanczos Bidiagonalization Algorithm) for efficient computation when the irlba package is available.

Usage

```
patterns(
  dat,
  k = 4,
  scale = FALSE,
  rotate = FALSE,
  monthly = FALSE,
  weight = TRUE,
  irlba_threshold = 5e+05
)
```

Arguments

dat	A ‘stars‘ object containing spatial and temporal dimensions
k	The number of PC/EOF modes to retain
scale	Logical, whether to scale before PCA
rotate	Logical, whether to apply Varimax rotation
monthly	Logical, whether to use monthly climatology
weight	Logical, whether to apply area weighting
irlba_threshold	Minimum number of data elements to trigger IRLBA usage (default: 50000). Set to Inf to always use base prcomp().

Value

A ‘patterns‘ object containing EOFs, amplitudes, and metadata

---

plot.common_patterns *Plot method for common_patterns objects*

Description

Shows shared EOFs on top and overlaid amplitude time series (colored by source) on the bottom.

Usage

```
## S3 method for class 'common_patterns'
plot(
  x,
  scale = c("standardized", "variance", "raw"),
  scale_y = c("fixed", "free"),
  overlay = NULL,
  overlay_color = "grey30",
  overlay_fill = NA,
  ...
)
```

Arguments

x	A common_patterns object
scale	Amplitude scaling: "standardized" (default), "variance", or "raw"
scale_y	Y-axis scaling: "fixed" (default) or "free"
overlay	Optional sf object to overlay on EOF maps
overlay_color	Color for overlay geometry (default "grey30")
overlay_fill	Fill for overlay geometry (default NA)
...	Additional arguments (currently unused)

Value

A patchwork object (EOFs + amplitudes)

`plot.coupled_patterns` *Plot coupled patterns diagnostics*

Description

Provides visualization helpers for ‘coupled_patterns‘ objects. The default ‘type = "combined"‘ mirrors the patterns plotting workflow by displaying the predictor and response spatial patterns alongside their canonical variate time series. Additional types include canonical correlation bars, standalone canonical variate panels, canonical spatial patterns, or direct access to the underlying predictor / response pattern plots.

Usage

```
## S3 method for class 'coupled_patterns'
plot(
  x,
  type = c("combined", "correlations", "canonical", "canonical_patterns", "predictor",
    "response"),
  side = c("predictor", "response", "both"),
  data = NULL,
  k = NULL,
  scaled = FALSE,
  ...
)
```

Arguments

`x`	A ‘coupled_patterns‘ object.
`type`	Plot type: one of ‘"combined"‘, ‘"correlations"‘, ‘"canonical"‘, ‘"canonical_patterns"‘, ‘"predictor"‘, or ‘"response"‘.
`side`	When ‘type = "canonical"‘ or ‘type = "canonical_patterns"‘, choose from the predictor side, response side, or both (values: ‘"predictor"‘, ‘"response"‘, ‘"both"‘). Ignored for other plot types.
`data`	Optional amplitudes or patterns for the canonical variate plots. For ‘side = "both"‘, a list with elements ‘predictor‘ and ‘response‘ may be supplied. Defaults to the training patterns stored in ‘x‘ when omitted.
`k`	Number of canonical modes to display (defaults to all available).
`scaled`	Logical, passed to the underlying pattern plots when relevant (defaults to ‘FALSE‘).
`...`	Additional arguments forwarded to ‘plot.patterns()‘ for ‘type = "predictor"‘ or ‘type = "response"‘ calls.

Value

A ggplot object (or a patchwork object when ‘type = "combined"‘ or ‘type = "canonical_patterns"‘ with ‘side = "both"‘).

`plot.patterns` *Plot method for patterns objects*

Description

Plot method for patterns objects

Usage

```
## S3 method for class 'patterns'
plot(
  x,
  type = "combined",
  scaled = FALSE,
  rawdata = NULL,
  scale = c("standardized", "variance", "raw"),
  scale_y = c("fixed", "free"),
  events = NULL,
  overlay = NULL,
  overlay_color = "grey30",
  overlay_fill = NA,
  ...
)
```

Arguments

`x`	A patterns object
`type`	Type of plot: "combined" (default), "eofs", or "amplitudes"
`scaled`	For EOFs: show correlations (TRUE) or raw loadings (FALSE)
`rawdata`	Optional raw data for correlation calculation when scaled = TRUE
`scale`	For amplitudes: scaling method ("standardized", "variance", "raw")
`scale_y`	For amplitudes: y-axis scaling ("fixed" or "free")
`events`	For amplitudes: optional dates to mark with vertical lines
`overlay`	Optional sf object to overlay on EOF maps (e.g., coastlines, boundaries)
`overlay_color`	Color for overlay geometry (default "grey30")
`overlay_fill`	Fill for overlay geometry (default NA for no fill)
`...`	Additional arguments (currently unused)

Value

A ggplot2 object or patchwork object for combined plots

predict.coupled_patterns *Predict Method for Coupled Patterns*

Description

This function makes predictions using a coupled_patterns object created by couple(). It applies the learned CCA relationship to new predictor data to predict response patterns.

Usage

```
## S3 method for class 'coupled_patterns'
predict(
  object,
  newdata,
  k = NULL,
  reconstruct = TRUE,
  predictor_patterns = NULL,
  ...
)
```

Arguments

object	A coupled_patterns object from couple()
newdata	New predictor data (stars object) for making predictions
k	Number of CCA modes to use for prediction. If NULL, uses all available modes
reconstruct	Logical, whether to reconstruct the full spatial field (default: TRUE)
predictor_patterns	Optional patterns object to use instead of the one stored in the coupled object. Useful for cross-source prediction with common EOFs: the override patterns share the same EOF space but carry a different climatology.
...	Additional arguments (currently unused)

Value

If reconstruct=TRUE, returns a stars object with reconstructed spatial fields. If reconstruct=FALSE, returns a tibble with predicted amplitudes.

Examples

```
## Not run:
# Create coupled patterns
coupled <- couple(pred_patterns, resp_patterns, k = 3)

# Make predictions on new data
predictions <- predict(coupled, new_predictor_data)
```

```
# Just get predicted amplitudes without spatial reconstruction
amplitudes <- predict(coupled, new_predictor_data, reconstruct = FALSE)

# Cross-source prediction with common EOFs
cpat <- common_patterns(list(era = era, phyda = phyda), k = 5)
coupled <- couple(cpat$era, fine_patterns, k = 3)
predict(coupled, phyda_new, predictor_patterns = cpat$phyda)

## End(Not run)
```

`prep_cv_folds` *Prepare cross-validation folds for CCA tuning*

Description

Computes EOF patterns at maximum k once per fold, enabling cheap truncation during grid search. This is the expensive step - call it once, then use ‘tune_cca()‘ for fast hyperparameter exploration.

Usage

```
prep_cv_folds(
  predictor,
  response,
  kfolds = 5,
  max_k_pred = 10,
  max_k_resp = 10,
  scale = FALSE,
  scale_pred = NULL,
  scale_resp = NULL,
  rotate = FALSE,
  monthly = FALSE,
  weight = TRUE,
  common_with = NULL
)
```

Arguments

`predictor`	A stars object with predictor data
`response`	A stars object with response data
`kfolds`	Number of cross-validation folds (default 5)
`max_k_pred`	Maximum number of predictor EOFs to compute (default 10)
`max_k_resp`	Maximum number of response EOFs to compute (default 10)
`scale`	Logical, whether to scale data before EOF extraction (default FALSE). Sets the default for both predictor and response; use ‘scale_pred‘ and/or ‘scale_resp‘ to override individually.

scale_pred	Logical, whether to scale predictor data before EOF extraction. Overrides 'scale' for the predictor when not NULL (default NULL).
scale_resp	Logical, whether to scale response data before EOF extraction. Overrides 'scale' for the response when not NULL (default NULL).
rotate	Logical, whether to apply varimax rotation (default FALSE)
monthly	Logical, whether to compute monthly climatology (default FALSE)
weight	Logical, whether to apply area weighting (default TRUE)
common_with	Optional named list of additional stars objects to include in common EOF computation via [common_patterns()]. When provided, predictor patterns are computed jointly with these datasets. The primary predictor is included under the name '.primary'. Test times are excluded from all datasets to prevent leakage.

Value

A cv_folds S3 object containing:

folds	List of fold data, each containing train patterns and test data
max_k_pred	Maximum predictor k used
max_k_resp	Maximum response k used
kfolds	Number of folds
common_times	Vector of overlapping time steps
pattern_opts	List of pattern extraction options
common_with_sources	Names of common EOF sources (if used)

Examples

```
## Not run:
# Standard folds
cv <- prep_cv_folds(coarse_data, fine_data,
                    kfolds = 5, max_k_pred = 10, max_k_resp = 10)

# With common EOFs from additional sources
cv <- prep_cv_folds(coarse_data, fine_data,
                    common_with = list(phyda = phyda_coarse),
                    kfolds = 5, max_k_pred = 10, max_k_resp = 10)

# Then tune over k_pred and k_resp (unchanged)
results <- tune_cca(cv, k_pred = 1:10, k_resp = 1:10)

## End(Not run)
```

prep_folds *Prepare contiguous cross-validation folds*

Description

Divides time steps into k contiguous folds for temporal cross-validation. Folds are kept contiguous to preserve temporal structure.

Usage

```
prep_folds(times, kfolds = 5)
```

Arguments

times	Vector of time values to split
kfolds	Number of folds (default 5)

Value

List of k vectors, each containing the time values for that fold

See Also

[prep_cv_folds()] for preparing complete CV folds with pre-computed patterns

[tune_cca()] for hyperparameter grid search

print.coupled_patterns *Print method for coupled_patterns*

Description

Print method for coupled_patterns

Usage

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

Arguments

x	A coupled_patterns object
...	Additional arguments (ignored)

print.cv_folds *Print method for cv_folds objects*

Description

Print method for cv_folds objects

Usage

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

Arguments

x	A cv_folds object
...	Additional arguments (ignored)

print.patterns *Print method for patterns objects*

Description

Print method for patterns objects

Usage

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

Arguments

x	A patterns object
...	Additional arguments passed to print

project_patterns *Project New Data onto Existing Patterns*

Description

This function projects new spatial-temporal data onto existing EOF patterns, returning the corresponding principal component time series. This is a core function used in pattern-based downscaling and reconstruction.

Usage

```
project_patterns(patterns, newdata)
```

Arguments

patterns	A patterns object containing EOFs, climatology, and other metadata
newdata	A stars object with new spatial-temporal data to project

Value

A tibble with time column and PC amplitude columns

Examples

```
## Not run:
# Get patterns from training data
pat <- patterns(training_data, k = 5)

# Project new data onto these patterns
new_amplitudes <- project_patterns(pat, new_data)

## End(Not run)
```

project_patterns_multiple *Project Multiple Datasets onto Patterns*

Description

Convenience function to project multiple datasets onto the same patterns

Usage

```
project_patterns_multiple(patterns, data_list, names = NULL)
```

Arguments

patterns	A patterns object
data_list	List of stars objects to project
names	Optional names for the datasets

Value

List of projected amplitude tibbles

---

reconstruct *Reconstruct spatial field from EOF amplitudes*

Description

Converts PC amplitudes back into a full spatial-temporal field by multiplying by the EOF patterns and adding back the climatology.

Usage

```
reconstruct(target_patterns, amplitudes = NULL)
```

Arguments

target_patterns	A patterns object containing EOFs and climatology
amplitudes	Amplitudes to use for reconstruction. Can be: - NULL (default): uses original amplitudes from target_patterns - tibble: with time column and PC columns - stars object: will be projected onto patterns first

Details

For bounded variables like precipitation, the reconstructed field may contain small negative values due to EOF truncation. To clamp these, use ‘mutate(result, across(everything(), ~pmax(.x, 0 * .x)))‘ (the ‘0 * .x‘ trick preserves units).

Value

A stars object with reconstructed spatial-temporal data

restore_climatology *Restore original field from anomalies and climatology*

Description

Reverses the operation of `get_anomalies()`, adding the climatological mean (and optionally multiplying by standard deviation) back to anomaly fields.

Usage

```
restore_climatology(anomalies, clim, scale = FALSE, monthly = FALSE)
```

Arguments

`anomalies`	A stars object containing anomalies (from `get_anomalies()`)
`clim`	A climatology list with `mean` and `sd` stars objects (from `get_climatology()`)
`scale`	Logical. If TRUE, multiply by standard deviation before adding mean (use when anomalies were standardized)
`monthly`	Logical. If TRUE, restore using monthly climatology

Value

A stars object with the original field restored

Examples

```
## Not run:
clim <- get_climatology(dat)
anom <- get_anomalies(dat, clim)
restored <- restore_climatology(anom, clim)

## End(Not run)
```

screeplot.patterns *Scree plot for EOF patterns*

Description

Scree plot for EOF patterns

Usage

```
## S3 method for class 'patterns'
screeplot(x, k = NULL, kmax = 10, rule_n = FALSE, ...)
```

Arguments

x	A patterns object from patterns()
k	Optional number of components to highlight with vertical line
kmax	Maximum number of components to show (default 10)
rule_n	Logical, whether to show the modified Rule N significance cutoff as a dashed blue line (default FALSE)
...	Additional arguments (currently unused)

Value

A ggplot object

Examples

```
## Not run:
pat <- patterns(data, k = 5)
screeplot(pat)
screeplot(pat, k = 3, kmax = 8)
screeplot(pat, rule_n = TRUE)  # show significance cutoff

## End(Not run)
```

---

summarize_cv *Summarize cross-validation results*

Description

Aggregates results across folds and identifies best hyperparameters.

Usage

```
summarize_cv(cv_results, metric = "rmse", minimize = TRUE)
```

Arguments

cv_results	A tibble from ‘tune_cca()‘
metric	Which metric to optimize (default "rmse")
minimize	Logical, whether to minimize (TRUE for RMSE) or maximize (FALSE for correlations). Default TRUE.

Value

A tibble with mean and sd of each metric per parameter combination, sorted by the target metric. Best parameters are attached as attribute "best_params".

Examples

```
## Not run:
results <- tune_cca(cv, k_pred = 1:5, k_resp = 1:5)
summary <- summarize_cv(results, metric = "rmse", minimize = TRUE)

# Get best parameters (k_pred, k_resp, k_cca)
attr(summary, "best_params")

## End(Not run)
```

`summarize_eof_cv` *Summarize EOF cross-validation results*

Description

Aggregates results across folds and identifies optimal k.

Usage

```
summarize_eof_cv(cv_results, metric = "rmse", minimize = TRUE)
```

Arguments

`cv_results`	A tibble from ‘tune_eof()‘
`metric`	Which metric to optimize (default "rmse")
`minimize`	Logical, whether to minimize (TRUE for RMSE) or maximize (FALSE for correlations). Default TRUE.

Value

A tibble with mean and sd of each metric per k value, sorted by the target metric. Best k is attached as attribute "best_k".

`summary.coupled_patterns`
Summary method for coupled_patterns

Description

Summary method for coupled_patterns

Usage

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

Arguments

object	A coupled_patterns object
...	Additional arguments (ignored)

---

tune_cca *Tune CCA hyperparameters via cross-validation*

Description

Performs grid search over k_pred, k_resp, and optionally k_cca using precomputed patterns from ‘prep_cv_folds()‘. Pattern truncation is cheap, so this runs quickly even with large grids.

Usage

```
tune_cca(
  cv_folds,
  k_pred = 1:10,
  k_resp = 1:10,
  k_cca = NULL,
  metrics = c("rmse", "cor_spatial", "cor_temporal"),
  parallel = FALSE
)
```

Arguments

cv_folds	A cv_folds object from ‘prep_cv_folds()‘
k_pred	Vector of predictor EOF counts to try (default 1:10)
k_resp	Vector of response EOF counts to try (default 1:10)
k_cca	Vector of CCA mode counts to try, or NULL (default) to use ‘min(k_pred, k_resp)‘ for each combination. Using fewer CCA modes than the maximum can act as regularization.
metrics	Character vector of metrics to compute. Options: "rmse", "cor_spatial", "cor_temporal" (default: all three)
parallel	Logical, whether to use furrr for parallel execution (default FALSE)

Value

A tibble with columns: k_pred, k_resp, k_cca, fold, and one column per metric requested.

Examples

```
## Not run:
cv <- prep_cv_folds(coarse_data, fine_data, kfolds = 5,
                    max_k_pred = 10, max_k_resp = 10)

# Grid search over k_pred and k_resp (k_cca = min automatically)
results <- tune_cca(cv, k_pred = 1:10, k_resp = 1:10)

# Also tune k_cca for regularization
results <- tune_cca(cv, k_pred = 1:10, k_resp = 1:10, k_cca = 1:5)

# Summarize and find best params
summary <- summarize_cv(results, metric = "rmse")

## End(Not run)
```

---

tune_eof *Cross-validate EOF truncation for a single field*

Description

Evaluates reconstruction skill for different numbers of EOFs using k-fold cross-validation. For each fold, EOFs are fit on training data, test data is projected onto those EOFs, and the reconstruction is compared to the original test data.

Usage

```
tune_eof(
  data,
  k = 1:10,
  kfolds = 5,
  max_k = max(k),
  metrics = c("rmse", "cor_spatial", "cor_temporal"),
  scale = FALSE,
  monthly = FALSE,
  weight = TRUE
)
```

Arguments

data	A stars object with spatial-temporal data
k	Vector of EOF counts to evaluate (default 1:10)
kfolds	Number of cross-validation folds (default 5)
max_k	Maximum EOFs to compute per fold (default max(k))
metrics	Character vector of metrics to compute. Options: "rmse", "cor_spatial", "cor_temporal" (default: all three)

scale	Logical, whether to scale data before EOF extraction (default FALSE)
monthly	Logical, whether to compute monthly climatology (default FALSE)
weight	Logical, whether to apply area weighting (default TRUE)

Value

A tibble with columns: k, fold, and one column per metric.

Examples

```
## Not run:
# Find optimal k for precipitation field
results <- tune_eof(precip_data, k = 1:15, kfolds = 5)
summary <- summarize_eof_cv(results, metric = "rmse")

# Plot reconstruction skill vs k
library(ggplot2)
results %>%
  group_by(k) %>%
  summarize(rmse = mean(rmse)) %>%
  ggplot(aes(k, rmse)) + geom_line() + geom_point()

## End(Not run)
```

Index