Package: fftab (via r-universe)

March 27, 2025

Title Tidy Manipulation of Fourier Transformed Data

Version 0.1.0.9000

Description The 'fftab' package stores Fourier coefficients in a tibble and allows their manipulation in various ways. Functions are available for converting between complex, rectangular ('re', 'im'), and polar ('mod', 'arg') representations, as well as for extracting components as vectors or matrices. Inputs can include vectors, time series, and arrays of arbitrary dimensions, which are restored to their original form when inverting the transform. Since 'fftab' stores Fourier frequencies as columns in the tibble, many standard operations on spectral data can be easily performed using tidy packages like 'dplyr'.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests ggfortify, patchwork, testthat (>= 3.0.0), zoo

Config/testthat/edition 3

Imports dplyr, ggplot2, lifecycle, tibble, tidyr

URL `https://github.com/thk686/fftab`, `https://thk686.github.io/fftab/`

BugReports `https://github.com/thk686/fftab/issues`

Config/Needs/website rmarkdown

Config/pak/sysreqs libicu-dev

Repository https://thk686.r-universe.dev

RemoteUrl https://github.com/thk686/fftab

RemoteRef HEAD

RemoteSha 8786a3a34c8bb063b6c2038110e4f7e3c4328b9d

Contents

`add_cplx` *Add Additional Representations to Fourier Coefficients*

Description

These functions add additional representations to a `fftab` object without removing or modifying existing representations.

Usage

```
add_cplx(x)

add_rect(x)

add_polr(x)
```

Arguments

`x`	A `fftab` object containing Fourier coefficients and associated metadata.

Details

- `add_cplx()`: Adds a **complex** (`"cplx"`) representation to the Fourier coefficients.
- `add_rect()`: Adds a **rectangular** (`"rect"`) representation to the Fourier coefficients.
- `add_polr()`: Adds a **polar** (`"polr"`) representation to the Fourier coefficients.

These functions are useful for working with multiple representations simultaneously without overwriting existing data.

Value

A `fftab` object with the additional representation included.

See Also

- `fftab()`

Examples

```
matrix(1:9, 3) |>
  fftab() |>
  print(n = 3) |>
  add_polr() |>
  print(n = 3) |>
  add_rect() |>
  print(n = 3) |>
  add_cplx() |>
  print(n = 3)
```

`add_l2nm` *Add L2 Norm and Squared L2 Norm of Frequency Dimensions*

Description

These functions compute and append the L2 norm and squared L2 norm of the frequency dimensions (`.dim_*` columns) as new columns in a `fftab` object.

Usage

```
add_l2nm(x)

add_l2sq(x)

get_l2nm(x)

get_l2sq(x)
```

Arguments

`x`	A `fftab` object containing frequency dimensions (`.dim_*`) and associated metadata.

Details

- `add_l2nm()`: Appends a column `l2nm` containing the L2 norm, calculated as the square root of the sum of squared values across `.dim_*` columns.
- `add_l2sq()`: Appends a column `l2sq` containing the squared L2 norm, calculated as the sum of squared values across `.dim_*` columns.
- `get_l2nm()`: Returns a numeric vector representing the L2 norm for each row.
- `get_l2sq()`: Returns a numeric vector representing the squared L2 norm for each row.

Value

A vector or `fftab` object with additional columns:

- `l2nm`: The L2 norm of the frequency dimensions.
- `l2sq`: The squared L2 norm of the frequency dimensions.

See Also

- `fftab()`
- `tibble::add_column()`

Examples

```
matrix(1:9, 3) |>
  fftab() |>
  print(n = 3) |>
  add_l2nm() |>
  print(n = 3) |>
  add_l2sq() |>
  print(n = 3)
```

`can_repr` *Manage Representations of a* `fftab` *Object*

Description

These functions handle representation management for a `fftab` object:

Usage

```
can_repr(x, repr)

get_repr(x)

set_repr(x, repr)
```

Arguments

`x`	A `fftab` object.
`repr`	For `can_repr()`, a character vector specifying representations (`"polr"`, `"rect"`, `"cplx"`).

Details

- `get_repr()`: Retrieve current representations.
- `can_repr()`: Check if the object supports specific representations.
- `set_repr()`: Convert the object to one or more specified representations.

Value

- `can_repr()`: Logical value (`TRUE` or `FALSE`) indicating if the object supports the specified representations.
- `get_repr()`: A character vector of current representations.
- `set_repr()`: A modified `fftab` object with the specified representation(s).

See Also

`to_cplx()`, `has_cplx()`

Examples

```
fftab(c(1, 0, -1, 0)) |> can_repr("cplx")

fftab(c(1, 0, -1, 0)) |> get_repr()

fftab(c(1, 0, -1, 0)) |> set_repr(c("polr", "rect"))
```

`cross_spec` *Compute the Cross-Spectrum (Cross FFT)*

Description

The `cross_spec` function computes the cross-spectrum between two inputs using the Fourier transform. It supports multiple input types including numeric vectors, time series (`ts`), arrays, and `fftab` objects. The function provides options for normalization and controlling whether the conjugate of the second input is used.

Usage

```
cross_spec(a, b, norm = FALSE, conj = TRUE)

## Default S3 method:
cross_spec(a, b, norm = FALSE, conj = TRUE)

## S3 method for class 'ts'
cross_spec(a, b, norm = FALSE, conj = TRUE)

## S3 method for class 'array'
cross_spec(a, b, norm = FALSE, conj = TRUE)

## S3 method for class 'fftab'
cross_spec(a, b, norm = FALSE, conj = TRUE)
```

Arguments

`a`	The first input for the cross FFT. Supported types include numeric vectors, `ts` objects, arrays, and `fftab` objects.
`b`	The second input for the cross FFT. Must match the dimensions or structure of `a`.
`norm`	Logical; if `TRUE`, normalizes the Fourier transforms before computation. Default is `FALSE`.
`conj`	Logical; if `TRUE`, uses the complex conjugate of the Fourier transform of `b`. Default is `TRUE`.

Value

An object representing the cross-spectrum:

- For `default` and `fftab` methods: A `fftab` object.
- For `ts` objects: A `fftab` object with `.tsp` attributes inherited from `a`.
- For arrays: A `fftab` object with `.dim` attributes inherited from `a`.

Methods (by class)

- `cross_spec(default)`: Default method for computing cross FFT. Converts inputs to `fftab` objects before computation.
- `cross_spec(ts)`: Method for time series (`ts`) objects. Ensures the time series frequencies are consistent and preserves the `tsp` attribute.
- `cross_spec(array)`: Method for array inputs. Ensures dimensions are consistent and preserves the `dim` attribute.
- `cross_spec(fftab)`: Method for `fftab` objects. Performs the cross-frequency transform directly using the Fourier transforms of `a` and `b`.

See Also

`fftab()`

Examples

```
cross_spec(rnorm(8), rnorm(8), norm = TRUE)

cross_spec(
  ts(rnorm(8), frequency = 4),
  ts(rnorm(8), frequency = 4)
)
```

`fftab` *Perform FFT and IFFT with Tidy Results*

Description

Provides functions to compute the Fast Fourier Transform (FFT) and its inverse (IFFT) while maintaining results in a tabular format. Supports vectors, time series (`ts`), and arrays as inputs.

Usage

```
fftab(x, norm = FALSE)

## Default S3 method:
fftab(x, norm = FALSE)

## S3 method for class 'ts'
fftab(x, norm = FALSE)

## S3 method for class 'array'
fftab(x, norm = FALSE)

ifftab(x)
```

Arguments

`x`	Input object for which to compute the FFT or IFFT. This can be: • A numeric vector (default method for `fftab`). • A time series object (`ts`) for `fftab.ts`. • A multidimensional numeric array for `fftab.array`. • A `fftab` object for `ifftab`.
`norm`	Logical. If `TRUE`, computes normalized coefficients for FFT.

Details

- `fftab` organizes FFT results into a tibble for downstream analysis.
- `ifftab` ensures that reconstructed signals match the input structure (e.g., arrays, `ts`).

Value

- `fftab`: A tibble containing:
  - Fourier frequencies (`.dim_1`, `.dim_2`, etc.).
  - FFT values stored in the `fx` column as complex values.
- `ifftab`: A vector, array, or time series object representing the reconstructed signal.

FFT

The `fftab` function computes the FFT for different input types:

- **Default Input** (`fftab.default`): Computes FFT for numeric vectors.
- **Time Series Input** (`fftab.ts`): Handles FFT for `ts` objects, scaling frequencies appropriately.
- **Array Input** (`fftab.array`): Processes multidimensional arrays.

Results are returned as a tibble containing Fourier frequencies and FFT values.

IFFT

The `ifftab` function reconstructs the original signal from a `fftab` object. It supports vectors, arrays, and time series inputs. The inverse transform preserves the original structure (e.g., array dimensions or time series attributes).

See Also

`stats::fft()`

Examples

```
fftab(c(1, 0, -1, 0))

fftab(c(1, 0, -1, 0)) |> ifftab()

ts(sin(1:10), frequency = 12) |> fftab()

array(1:8, dim = c(2, 2, 2)) |> fftab()
```

`fourier_frequencies` *Compute Fourier Frequencies*

Description

Computes Fourier frequencies for various types of inputs, such as scalars, vectors, matrices, time series, or arrays. This generic function dispatches appropriate methods based on the input type.

Usage

```
fourier_frequencies(x)

## Default S3 method:
fourier_frequencies(x)

## S3 method for class 'ts'
fourier_frequencies(x)

## S3 method for class 'array'
fourier_frequencies(x)
```

Arguments

`x`	The input object. Supported input types: • **Scalar or vector**: The length of the sequence. • **Time series** (`ts`): Frequencies are scaled based on the sampling rate. • **Multidimensional array or matrix**: Frequencies are computed for each dimension.

Details

This function has the following methods:

- **Default Input** (`fourier_frequencies.default`): Computes normalized Fourier frequencies for scalar or vector inputs.
- **Time Series Input** (`fourier_frequencies.ts`): Computes frequencies scaled by the frequency attribute of a `ts` object.
- **Multidimensional Arrays** (`fourier_frequencies.array`): Computes frequencies for each dimension of a matrix or array.

See the examples for details on each case.

Value

A tibble where:

- `.dim_1`, `.dim_2`, ..., represent the Fourier frequencies for each dimension.

See Also

`tidyr::expand_grid()`, `frequency()`

Examples

```
# Default input (vector)
fourier_frequencies(8)

# Time series input
```

```
ts(rnorm(36), frequency = 12) |> fourier_frequencies()

# Multidimensional array input
array(1:27, dim = c(3, 3, 3)) |> fourier_frequencies()

# Matrix input
matrix(1:9, nrow = 3, ncol = 3) |> fourier_frequencies()
```

---

get_fx *Extract Fourier Coefficients and Components*

Description

These utility functions extract specific components from a `fftab` object. `get_fx` retrieves the raw Fourier coefficients, while `get_fx_norm` ensures the coefficients are either normalized or not normalized based on the `norm` parameter.

Usage

```
get_fx(x)

get_fx_norm(x, norm = FALSE)

get_re(x)

get_im(x)

get_mod(x)

get_arg(x)
```

Arguments

`x`	A `fftab` object containing FFT results.
`norm`	Logical. If `TRUE`, forces normalized coefficients. If `FALSE`, ensures non-normalized coefficients.

Details

- `get_fx`: Returns coefficients as they are stored in the `fftab` object.
- `get_fx_norm`: Adjusts coefficients if they are not in the desired normalization state.
- `get_re`, `get_im`: Extract real and imaginary components.
- `get_mod`, `get_arg`: Compute magnitude and phase of coefficients.

Value

The requested components:

- `get_fx`: A complex vector of raw Fourier coefficients (`fx`) as stored in the object.
- `get_fx_norm`: A complex vector of Fourier coefficients, explicitly normalized or non-normalized based on the `norm` parameter.
- `get_re`: A numeric vector of real parts (`re`).
- `get_im`: A numeric vector of imaginary parts (`im`).
- `get_mod`: A numeric vector of magnitudes (`mod`).
- `get_arg`: A numeric vector of phase angles (`arg`), in radians.

See Also

`to_cplx()`, `to_rect()`, `to_polr()`

Examples

```
fftab(c(1, 0, -1, 0)) |> get_fx()

fftab(c(1, 0, -1, 0)) |> get_fx_norm(norm = TRUE)

fftab(c(1, 0, -1, 0)) |> get_re()

fftab(c(1, 0, -1, 0)) |> get_im()

fftab(c(1, 0, -1, 0)) |> get_mod()

fftab(c(1, 0, -1, 0)) |> get_arg()
```

`get_rect` *Extract Rectangular or Polar Components*

Description

The `get_rect` and `get_polr` functions extract specific components from a `fftab` object, representing the Fourier coefficients in either rectangular or polar form.

Usage

```
get_rect(x)

get_polr(x)
```

Arguments

`x`	A matrix object containing FFT results.

Value

- `get_rect`: A matrix with two columns:
  - `re`: The real part of the coefficients.
  - `im`: The imaginary part of the coefficients.
- `get_polr`: A matrix with two columns:
  - `mod`: The magnitude of the coefficients.
  - `arg`: The phase angle of the coefficients, in radians.

See Also

`get_fx()`, `get_re()`, `get_mod()`, `to_rect()`, `to_polr()`

Examples

```
fftab(c(1, 0, -1, 0)) |> get_rect()

fftab(c(1, 0, -1, 0)) |> get_polr()
```

`has_cplx` *Check Representations of a* `fftab` *Object*

Description

These functions check if specific representations are present in a `fftab` object:

Usage

```
has_cplx(x)

has_rect(x)

has_polr(x)
```

Arguments

`x`	A `fftab` object.

Details

- `has_cplx()`: Checks if the object has complex representation (`fx` column).
- `has_rect()`: Checks if the object has rectangular representation (`re`, `im` columns).
- `has_polr()`: Checks if the object has polar representation (`mod`, `arg` columns).

Value

Logical value (`TRUE` or `FALSE`) indicating whether the specified representation exists.

See Also

`to_cplx()`, `get_repr()`

Examples

```
fftab(c(1, 0, -1, 0)) |> has_cplx()

fftab(c(1, 0, -1, 0)) |> has_rect()

fftab(c(1, 0, -1, 0)) |> has_polr()
```

`phase_diff` *Compute Phase Difference and Maximum Correlation Between Two Signals*

Description

Computes the phase difference and maximum normalized correlation between two input signals after phase-aligning the second signal (`b`) to the first signal (`a`).

Usage

```
phase_diff(a, b)
```

Arguments

`a`	A numeric vector or time series representing the first signal.
`b`	A numeric vector or time series representing the second signal.

Details

[Experimental]

This function performs the following steps:

1. Computes the Fourier Transform of both input signals using `fftab`.
2. Calculates the **cross-spectrum** of the signals.
3. Converts the cross-spectrum to polar form and computes the weighted average phase difference.
4. Adjusts the phase of the second signal (`b`) using `.shift_phase` to maximize alignment with the first signal (`a`).
5. Computes the **normalized correlation** between the phase-aligned signals.

The correlation is normalized using the variances of both signals and will generally be **higher** than the correlation between the original signals due to the optimal phase alignment.

Value

A numeric vector of length two:

- The first element represents the **phase difference** (in radians) required to maximize alignment between the two signals.
- The second element represents the **maximum normalized correlation** achieved after phase alignment.

See Also

- `fftab()`
- `cross_spec()`

Examples

```
phase_diff(
  sin(seq(0, 2 * pi, length.out = 128)),
  cos(seq(0, 2 * pi, length.out = 128))
)
```

`plot.fftab` *Plot the modulus of FFT results*

Description

Plots the modulus of the FFT results against the frequencies.

Usage

```
## S3 method for class 'fftab'
plot(x, ...)
```

Arguments

`x`	A `fftab` object. This should contain Fourier-transformed data.
`...`	Additional arguments passed to `ggplot2::ggplot`.

Value

A `ggplot` object representing the modulus of FFT results plotted against the frequencies. The plot shows the modulus (`mod`) on the y-axis and frequency values on the x-axis.

`to_angf` *Convert between cyclic and angular frequencies*

Description

These functions convert dimensions of frequency from cyclic (measured in cycles) to angular (measured in radians) or vice versa. This transformation scales dimensions by a factor of 2 * pi.

- `to_angf()`: Converts from cyclic to angular frequency.
- `to_cycf()`: Converts from angular to cyclic frequency.

Usage

```
to_angf(x)

to_cycf(x)
```

Arguments

`x`	An `fftab` object containing frequency dimensions. Must include columns prefixed with `.dim_` and have an attribute `.is_angular` indicating the frequency type.

Value

An `fftab` object with dimensions scaled appropriately and the `.is_angular` attribute updated.

Examples

```
# Convert cyclic to angular frequencies
rnorm(64) |>
  fftab() |>
  to_angf() |>
  to_rect() |>
  dplyr::slice_max(abs(.dim_1), n = 5)

# Convert angular back to cyclic frequencies
rnorm(64) |>
  fftab() |>
  to_angf() |>
  to_cycf() |>
  to_rect() |>
  dplyr::slice_max(abs(.dim_1), n = 5)
```

to_cplx — *Convert a `fftab` Object Between Representations*

Description

These functions convert a `fftab` object to a specified representation:

- `to_cplx()`: Converts to complex representation (`fx`).
- `to_rect()`: Converts to rectangular representation (`re`, `im`).
- `to_polr()`: Converts to polar representation (`mod`, `arg`).

Usage

```
to_cplx(x, .keep = "unused")

to_rect(x, .keep = "unused")

to_polr(x, .keep = "unused")
```

Arguments

`x`	A `fftab` object.
`.keep`	Specifies which columns to retain. See `dplyr::mutate()`.

Details

- `to_cplx()`: Converts from rectangular (`re`, `im`) or polar (`mod`, `arg`) components to complex form.
- `to_rect()`: Converts from complex (`fx`) or polar components to rectangular form.
- `to_polr()`: Converts from complex (`fx`) or rectangular components to polar form.

Value

A modified `fftab` object containing the specified representation:

- `to_cplx()`: Adds the `fx` column for complex values.
- `to_rect()`: Adds the `re` and `im` columns for rectangular components.
- `to_polr()`: Adds the `mod` and `arg` columns for polar components.

See Also

`has_cplx()`, `get_repr()`

Examples

```
fftab(c(1, 0, -1, 0)) |> to_cplx()

fftab(c(1, 0, -1, 0)) |> to_rect()

fftab(c(1, 0, -1, 0)) |> to_polr()
```

Index