# Package 'fCWTr'

June 17, 2024

**Type** Package

**Title** Fast Continuous Wavelet Transform

**Version** 0.2.1

**Maintainer** Lukas Schneiderbauer <lukas.schneiderbauer@gmail.com>

**Description** Enables the usage of the fast continuous wavelet transform, originally implemented in the 'C++' library 'fCWT' by Lukas Arts. See Arts, P.A. and Van den Broek, E.L. (2022) <doi:10.1038/s43588-021-00183-z> for details. The package includes simple helpers such as a plotting function.

**License** GPL (>= 3)

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.1

**Depends** R (>= 4.1)

**Suggests** ggplot2 (>= 3.3.0), hms (>= 1.1.0), viridis (>= 0.6.0), rlang (>= 1.0.0), testthat (>= 3.0.0)

**LinkingTo** cpp11

**SystemRequirements** fftw (>= 3.3.0), openmp (>= 15.0.0)

**Config/testthat/edition** 3

**URL** https://lschneiderbauer.github.io/fCWTr/, https://github.com/lschneiderbauer/fCWTr

**BugReports** https://github.com/lschneiderbauer/fCWTr/issues

**NeedsCompilation** yes

**Author** Lukas Schneiderbauer [aut, cre, cph] (<https://orcid.org/0000-0002-0975-6803>), Lukas P. A. Arts [cph] (Author of the majority of the C++ code, indicated in the respective header files.), Egon L. van den Broek [cph] (Author of the majority of the C++ code, indicated in the respective header files.)

**Repository** CRAN

**Date/Publication** 2024-06-17 16:10:05 UTC

# Contents



---

`as.data.frame.fcwtr_scalogram` *Coerce the scalogram matrix to a data frame*

---

## Description

Internally, the scalogram resulting from `fcwt()` is represented by a numeric matrix. This method coerces this matrix into a reasonable data frame. Note that this conversion has a significant run time cost.

## Usage

```
## S3 method for class 'fcwtr_scalogram'
as.data.frame(x, ...)
```

## Arguments

| | |
|---|---|
| x | An object resulting from `fcwt()`. |
| ... | additional arguments to be passed to or from methods. |

## Value

A `data.frame()` object representing the scalogram data with four columns:

**time_ind** An integer index uniquely identifying time slices.

**time** The time difference to the first time slice in physical units. The time unit is the inverse of the frequency unit chosen by the user for the `sample_freq` argument of `fcwt()`.

**freq** The frequency in the same units as the `sample_freq` argument of `fcwt()`.

**value** The fCWT result for the particular time-frequency combination.

## Examples

```
fcwt(
  sin((1:5000) * 2 * pi * 440 / 44100),
  sample_freq = 44100,
  n_freqs = 10
) |>
as.data.frame() |>
head()
```

---

`fcwt` *Fast continuous wavelet transform*

---

## Description

The core function of this package making use of the fCWT library. It processes an input signal in form of a real valued numeric vector interpreted as an evenly spaced time series and returns the absolute values of a spectogram, i.e. a graph with a time and a frequency dimension.

## Usage

```
fcwt(
  signal,
  sample_freq,
  n_freqs,
  freq_begin = 2 * sample_freq/length(signal),
  freq_end = sample_freq/2,
  sigma = 1,
  remove_coi = TRUE,
  n_threads = 2L
)
```

## Arguments

| | |
|---|---|
| `signal` | Real-valued time series. The time steps are assumed to be evenly spaced. |
| `sample_freq` | Sampling rate of input time series. This number primarily establishes a connection to physical units which is used in other frequency definitions as well as the units of the output data. |
| `n_freqs` | Number of frequency bins generated by the CWT. The frequencies are linearly distributed. Computation time increases when raising the number of frequency bins. |
| `freq_begin`, `freq_end` | Optionally specifies the frequency range `[freq_end, freq_begin]`. If not specified the maximal meaningful frequency range, depending on the input signal, is taken. The range and `sample_freq` need to be specified in the same units. |

`sigma` Sets a dimensionless parameter modifying the wavelet spread which in the time-domain is roughly given by $\Sigma_t \sim \sqrt{2}\frac{\sigma}{f}$. Changing this parameter adjusts the time/frequency uncertainty balance. Defaults to 1. Larger (lower) value of sigma corresponds to a better (worse) frequency resolution and a worse (better) time resolution.

`remove_coi` Boundary effects can result in nonphysical artifacts. If `remove_coi = TRUE`, those are effectively removed by setting corresponding values to `NA`. We define the essential support of the (Gaussian) wavelet to be four times its standard deviation, $4\Sigma_t = 2 * \sqrt{2}\frac{\sigma}{f}$, and so a wavelet touches the boundary if the distance of the center of the wavelet to the boundary is less then $4\Sigma_t$. Values that fall into that range are removed if `remove_coi = TRUE`.

`n_threads` Number of threads used by the computation, if supported by your platform. Defaults to 2 threads (to accomodate CRAN requirements).

## Details

The wavelet used in this calculation is the so called Morlet wavelet, a sinusoidal wave modulated by a Gaussian whose spread is controlled by the argument `sigma`.

See the original paper Arts, L.P.A., van den Broek, E.L. The fast continuous wavelet transformation (fCWT) for real-time, high-quality, noise-resistant time–frequency analysis. Nat Comput Sci 2, 47–58 (2022). doi:10.1038/s4358802100183z

## Value

The spectogram, a numeric real-valued matrix with dimensions `dim = c(length(signal), n_freqs)`. This matrix is wrapped into a S3-class `fcwtr_scalogram` so that plotting and coercion functions can be used conveniently.

## Examples

```
ts_sin_440 <- sin((1:5000) * 2 * pi * 440 / 44100)

res <-
  fcwt(
    ts_sin_440,
    sample_freq = 44100,
    freq_begin = 50,
    freq_end = 1000,
    n_freqs = 10,
    sigma = 5
  )
```

---

`fcwt_batch` *Fast continuous wavelet transform - Batch processing*

---

## Description

Performs a fast continuous wavelet transform on long sequences by sequentially processing junks of the input signal and keeping only low-resolution output data to preserve memory. This is only useful for very long signals whose output does not fit into memory as a whole. It should not be used on short signals since boundary artefacts are discarded (and those potentially dominate for short sequences).

## Usage

```
fcwt_batch(
  signal,
  sample_freq,
  n_freqs,
  time_resolution,
  freq_begin = 2 * sample_freq/length(signal),
  freq_end = sample_freq/2,
  sigma = 1,
  max_batch_size = ceiling(4 * 10^9/(n_freqs * 4)),
  n_threads = 2L,
  progress_bar = FALSE
)
```

## Arguments

| | |
|---|---|
| `signal` | Real-valued time series. The time steps are assumed to be evenly spaced. |
| `sample_freq` | Sampling rate of input time series. This number primarily establishes a connection to physical units which is used in other frequency definitions as well as the units of the output data. |
| `n_freqs` | Number of frequency bins generated by the CWT. The frequencies are linearly distributed. Computation time increases when raising the number of frequency bins. |
| `time_resolution` | The time resolution in inverse units of `sample_freq` of the result. Memory consumption is directly related to that. Can not be higher than the time resolution of the input signal. |
| `freq_begin`, `freq_end` | Optionally specifies the frequency range `[freq_end, freq_begin]`. If not specified the maximal meaningful frequency range, depending on the input signal, is taken. The range and `sample_freq` need to be specified in the same units. |
| `sigma` | Sets a dimensionless parameter modifying the wavelet spread which in the time-domain is roughly given by $\Sigma_t \sim \sqrt{2}\frac{\sigma}{f}$. Changing this parameter adjusts the time/frequency uncertainty balance. Defaults to 1. Larger (lower) value of sigma corresponds to a better (worse) frequency resolution and a worse (better) time resolution. |
| `max_batch_size` | The maximal batch size that is used for splitting up the input sequence. This limits the maximal memory that is used. Defaults to roughly 4GB. The actual batch size is optimized for use with FFTW. |

| | |
|---|---|
| n_threads | Number of threads used by the computation, if supported by your platform. Defaults to 2 threads (to accomodate CRAN requirements). |
| progress_bar | Monitoring progress can sometimes be useful when performing time consuming operations. Setting progress_bar = TRUE enables printing a progress bar to the console. Defaults to FALSE. |

## Details

In case of input sequences that exceed the a certain size, the output sequence will not fit into the local memory and the fcwt cannot be performed in one run. For instance, in case of processing a song of 10 minutes length (assuming a sampling rate of 44100 Hz), the size of the output vector is `10 * 60 seconds * 44100 Hz * nfreqs * 4 bytes`, which for e.g. `nfreqs = 200`, equals ~ 21 GB, hence nowadays already at the limit of the hardware of a modern personal computer.

In cases where the required output time-resolution is smaller than the time resolution of the input signal, one can perform the `fcwt()` and reduce the output size by averaging. (The input signal time resolution can in general not be reduced since high-frequency information would get lost.)

This function splits up the input sequence into batches, processes each batch separately, reduces the time resolution, and adds the outputs together.

Attention: Boundary artefacts are removed, so some high frequency information at the beginning and the end of the sequence is lost. (The amount depends on the minimal frequency captured `min_freq`.)

## Value

The spectogram, a numeric real-valued matrix with dimensions roughly `dim ~ c(length(signal) * time_resolution * sample_freq, n_freqs)`. The exact length of the output depends on boundary effect details. This matrix is wrapped into a S3-class `fcwtr_scalogram` so that plotting and coercion functions can be used conveniently.

## See Also

`fcwt()`

## Examples

```
res <-
  fcwt_batch(
    ts_sin_sin,
    sample_freq = 44100,
    freq_begin = 100,
    freq_end = 11000,
    n_freqs = 10,
    sigma = 10,
    max_batch_size = 50000,
    time_resolution = 0.001
  )
```

`plot.fcwtr_scalogram` *Scalogram plotting*

## Description

Plots the scalogram resulting from `fcwt()`. Requires ggplot2.

## Usage

```
## S3 method for class 'fcwtr_scalogram'
plot(x, n = 1000, ...)
```

## Arguments

| | |
|---|---|
| `x` | An object resulting from `fcwt()`. |
| `n` | The plotting function reduces the time resolution by averaging to generate a reasonable graphics format. `n` is the number of time steps that are plotted. Defaults to `n = 1000`. |
| `...` | other arguments passed to specific methods |

## Value

No return value, called for side effects.

## Examples

```
ts_sin_440 <- sin((1:4410) * 2 * pi * 440 / 44100)

res <-
  fcwt(
    ts_sin_440,
    sample_freq = 44100,
    freq_begin = 50,
    freq_end = 1000,
    n_freqs = 10,
    sigma = 5
  )

plot(res)
```

## ts_sin_440 *Pure sine wave*

### Description

Assuming a sample rate of 44100 Hz, the sine wave's frequency is 440 Hz.

### Usage

```
ts_sin_440
```

### Format

A numeric vector containing a time series signal.

## ts_sin_sin *A sinusoidal wave with varying frequency*

### Description

The frequency itself is changing in a sinusoidal fashion.

### Usage

```
ts_sin_sin
```

### Format

A numeric vector containing a time series signal.

## ts_sin_superpos *Superimposed sine waves of different frequencies and different amplitudes*

### Description

Assuming a sample rate of 44100 Hz, the superimposed signals' frequencies are 440 Hz, 880 Hz, 100 Hz, 500 Hz, 1200 Hz and 50 Hz.

### Usage

```
ts_sin_superpos
```

### Format

A numeric vector containing a time series signal.

# Index