Package ‘detrendr’

March 16, 2020

Type Package

Title Detrend Images

Version 0.6.5

Maintainer Rory Nolan <rorynolan@gmail.com>

Description Detrend fluorescence microscopy image series for fluorescence fluctuation and correlation spectroscopy ('FCS' and 'FFS') analysis. This package contains functionality published in a 2016 paper <doi:10.1093/bioinformatics/btx434> but it has been extended since then with the Robin Hood algorithm and thus contains unpublished work.

License BSD_3_clause + file LICENSE

URL https://rorynolan.github.io/detrendr,
     https://www.github.com/rorynolan/detrendr

BugReports https://www.github.com/rorynolan/detrendr/issues

Depends R (>= 3.1)

Imports autothresholdr (>= 1.3.3), checkmate (>= 1.9.3), doParallel, dplyr, filestrings (>= 3.1.5), foreach, ijitiff (>= 2.0.2), iterators, magrittr (>= 1.5), matrixStats, parallel, plyr, purrr, Rcpp (>= 1.0.1), RcppParallel (>= 5.0.0), rlang (>= 0.3.3), stringr (>= 1.4.0), tools, withr (>= 2.1.0)

Suggests abind, arrayhelpers, covr, ggplot2, glue (>= 1.3.0), graphics, gridExtra, knitr, lattice, MASS, mgcv, pacman, markdown, spelling, testthat, tidy

LinkingTo Rcpp (>= 1.0.1), ReppParallel (>= 5.0.0)

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.0.2

SystemRequirements GNU make
**apply_on_pillars**

Apply a function to each pillar of a 3-dimensional array.

**Description**

Define a 'pillar' of a 3-dimensional array as pillar i,j off array arr being arr[i,j,]. This function applies a specified function to each pillar.

**Usage**

```
apply_on_pillars(arr3d, FUN)
```

**Arguments**

- `arr3d` A 3-dimensional array.
- `FUN` A function which takes a vector as input and, for a given input length, outputs a vector of constant length (can be 1).
Value

If FUN is returning length 1 vectors, a matrix whereby mat[i,j] = FUN(arr3d[i,j,j]). If FUN is returning vectors of length l > 1, a 3-dimensional array whereby arr[i,j,j] = FUN(arr3d[i,j,j]).

---

**Value**

Find the best polynomial degree for polynomial detrending.

---

**Description**

Use Nolan’s algorithm to find the ideal polynomial degree for polynomial detrending.

**Usage**

best_degree(img, parallel = FALSE, purpose = c("FCS", "FFS"))

**Arguments**

- **img**
  A 4-dimensional array in the style of an *ijtiff_img* (indexed by img[y, x, channel, frame]) or a 3-dimensional array which is a single channel of an *ijtiff_img* (indexed by img[y, x, frame]).

- **parallel**
  Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use parallel = TRUE.

- **purpose**
  What type of calculation do you intend to perform on the detrended image? If it is an FFS (fluorescence fluctuation spectroscopy) calculation (like number and brightness), choose 'FFS’. If it is an FCS (fluorescence correlation spectroscopy) calculation (like cross-correlated number and brightness or autocorrelation), choose 'FCS’. The difference is that if purpose is 'FFS’, the time series is corrected for non-stationary mean and variance, whereas if purpose is 'FCS’, the time series is corrected for non-stationary mean only. ‘purpose” is not required for *Robin Hood* detrending.

**Value**

If no detrend is necessary, this function returns NA. If a detrend is required, this function returns a natural number which is the ideal polynomial degree for polynomial detrending. If there are multiple channels, the function returns a vector, one degree parameter for each channel.

**References**

Examples

```r
## Not run:
## These examples are not run on CRAN because they take too long.
## You can still try them for yourself.
img <- ijtiff::read_tif(system.file("extdata", "bleached.tif",
    package = "detrendr")
best_degree(img, parallel = 2)
## End(Not run)
```

### best_l

**Find the best length parameter for boxcar detrending.**

#### Description

Use Nolan’s algorithm to find the ideal length parameter for boxcar detrending. Boxcar detrending is also referred to as ‘running average’.

#### Usage

```r
best_l(img, parallel = FALSE, purpose = c("FCS", "FFS"))
```

#### Arguments

- `img`: A 4-dimensional array in the style of an `ijtiff_img` (indexed by `img[y,x,channel,frame]`) or a 3-dimensional array which is a single channel of an `ijtiff_img` (indexed by `img[y,x,frame]`).
- `parallel`: Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.
- `purpose`: What type of calculation do you intend to perform on the detrended image? If it is an FFS (fluorescence fluctuation spectroscopy) calculation (like number and brightness), choose 'FFS'. If it is an FCS (fluorescence correlation spectroscopy) calculation (like cross-correlated number and brightness or autocorrelation), choose 'FCS'. The difference is that if `purpose` is 'FFS', the time series is corrected for non-stationary mean and variance, whereas if `purpose` is 'FCS', the time series is corrected for non-stationary mean only. 'purpose' is not required for Robin Hood detrending.

#### Value

If no detrend is necessary, this function returns `NA`. If a detrend is required, this function returns a natural number which is the ideal length parameter for boxcar detrending. If there are multiple channels, the function returns a vector, one 1 parameter for each channel.
**best_swaps**

Find the best swaps parameter for Robin Hood detrending.

### Description

Use Nolan's algorithm to find the ideal swaps parameter for Robin Hood detrending.

### Usage

```r
best_swaps(img, quick = FALSE)
```

### Arguments

- **img**: A 4-dimensional array in the style of an `ijtiff_img` (indexed by `img[y,x,channel,frame]`) or a 3-dimensional array which is a single channel of an `ijtiff_img` (indexed by `img[y,x,frame]`).
- **quick**: If `FALSE` (the default), the swap finding routine is run several times to get a consensus for the best parameter. If `TRUE`, the swap finding routine is run only once.

### Value

A natural number. The ideal swaps parameter for boxcar detrending. If there are multiple channels, the function returns a vector, one swaps parameter for each channel.
Examples

## Not run:
## These examples are not run on CRAN because they take too long.
## You can still try them for yourself.
img <- ijtiff::read_tif(system.file("extdata", "bleached.tif",
    package = "detrendr"
))
best_swaps(img)

## End(Not run)

---

**best_tau**

*Find the best tau parameter for exponential smoothing detrending.*

Description

Use Nolan’s algorithm to find the ideal tau parameter for exponential smoothing detrending.

Usage

```r
best_tau(img, cutoff = 0.05, parallel = FALSE, purpose = c("FCS", "FFS"))
```

Arguments

- **img**
  
  A 4-dimensional array in the style of an ijtiff_img (indexed by `img[y,x,channel,frame]`) or a 3-dimensional array which is a single channel of an ijtiff_img (indexed by `img[y,x,frame]`).

- **cutoff**
  
  In exponential filtering detrending, for the weighted average, every point gets a weight. This can slow down the computation massively. However, many of the weights will be approximately zero. With cutoff, we say that any point with weight less than or equal to `cutoff` times the maximum weight may be ignored; so with `cutoff = 0.05`, any weight less than 5\% value of this parameter is sensible and its value should not be set to anything else without good reason.

- **parallel**
  
  Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

- **purpose**
  
  What type of calculation do you intend to perform on the detrended image? If it is an FFS (fluorescence fluctuation spectroscopy) calculation (like number and brightness), choose ‘FFS’. If it is an FCS (fluorescence correlation spectroscopy) calculation (like cross-correlated number and brightness or autocorrelation), choose ‘FCS’. The difference is that if `purpose` is ‘FFS’, the time series is corrected for non-stationary mean and variance, whereas if `purpose` is ‘FCS’, the time series is corrected for non-stationary mean only. ‘purpose” is not required for Robin Hood detrending.
Value

If no detrend is necessary, this function returns NA. If a detrend is required, this function returns a natural number which is the ideal tau parameter for exponential smoothing detrending. If there are multiple channels, the function returns a vector, one tau parameter for each channel.

References


Examples

```r
## Not run:
## These examples are not run on CRAN because they take too long.
## You can still try them for yourself.
img <- ijtiff::read_tif(system.file("extdata", "bleached.tif", package = "detrendr") )[, , 1, ]
best_tau(img, parallel = 2)

## End(Not run)
```

brightness_pillars

*Get the brightness of pillars of a 3d array.*

Description

For an *ijtiff_img*-style array `img` (indexed as `img[y,x,channel,frame]`), 3-dimensional array `mat3d`, pillar `xy` of channel `ch` is defined as `img[y,x,ch,]`. This function computes the brightness of each pillar.

Usage

```r
brightness_pillars(img, parallel = FALSE)
```

Arguments

- `img` A 4-dimensional array in the style of an *ijtiff_img* (indexed by `img[y,x,channel,frame]`) or a 3-dimensional array which is a single channel of an *ijtiff_img* (indexed by `img[y,x,frame]`).
- `parallel` Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use `parallel = TRUE`.

Value

An *ijtiff_img*-style array `arr` with one frame. `arr[y,x,ch,1]` is equal to `var(img[y,x,ch,]) / mean(img[y,x,ch,])`. 
Examples

aaa <- array(1:16, dim = c(2, 2, 4))
brightness_pillars(aaa)

detrend-directory  
Detrend all TIFF images in an entire folder.

Description

Batch processing. Apply any of the available detrending routines to detrend all of the TIFF images in a folder, saving the detrended images as TIFF files in the same folder.

Usage

dir_detrend_robinhood(
  folder_path = ".",
  swaps = "auto",
  thresh = NULL,
  quick = FALSE,
  msg = TRUE
)

dir_detrend_rh(
  folder_path = ".",
  swaps = "auto",
  thresh = NULL,
  quick = FALSE,
  msg = TRUE
)

dir_detrend_boxcar(
  folder_path = ".",
  l,
  purpose = c("FCS", "FFS"),
  thresh = NULL,
  parallel = FALSE,
  msg = TRUE
)

dir_detrend_exp(
  folder_path = ".",
  tau,
  purpose = c("FCS", "FFS"),
  thresh = NULL,
  parallel = FALSE,
  msg = TRUE
)
dir_detrend_polynom(
  folder_path = ".",
  degree,
  purpose = c("FCS", "FFS"),
  thresh = NULL,
  parallel = FALSE,
  msg = TRUE
)

Arguments

folder_path  The path (relative or absolute) to the folder you wish to process.
swaps        The number of swaps (giving of one count from rich to poor) to perform during the Robin Hood detrending. Set this to "auto" (the default) to use Nolan's algorithm to automatically find a suitable value for this parameter (recommended). For multi-channel images, it is possible to have a different swaps for each channel by specifying swaps as a vector or list.
thresh       The threshold or thresholding method (see autothresholdr::mean_stack_thresh()) to use on the image prior to detrending.
quick        If FALSE (the default), the swap finding routine is run several times to get a consensus for the best parameter. If TRUE, the swap finding routine is run only once.
msg          Receive messages to tell you how the processing of the directory is going? Default is yes.
l            The length parameter for boxcar detrending. The size of the sliding window will be $2 \times l + 1$. This must be a positive integer. Set this to "auto" to use Nolan's algorithm to automatically find a suitable value for this parameter (recommended). For multi-channel images, it is possible to have a different $l$ for each channel by specifying $l$ as a vector or list.
purpose      What type of calculation do you intend to perform on the detrended image? If it is an FFS (fluorescence fluctuation spectroscopy) calculation (like number and brightness), choose 'FFS'. If it is an FCS (fluorescence correlation spectroscopy) calculation (like cross-correlated number and brightness or autocorrelation), choose 'FCS'. The difference is that if purpose is 'FFS', the time series is corrected for non-stationary mean and variance, whereas if purpose is 'FCS', the time series is corrected for non-stationary mean only. 'purpose' is not required for Robin Hood detrending.
parallel    Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use parallel = TRUE.
tau          The tau parameter for exponential filtering detrending. This must be a positive number. Set this to "auto" to use Nolan's algorithm to automatically find a suitable value for this parameter (recommended). For multi-channel images, it is possible to have a different tau for each channel by specifying tau as a vector or list.
degree

The degree of the polynomial to use for the polynomial detrending. This must be a positive integer. Set this to "auto" to use Nolan’s algorithm to automatically find a suitable value for this parameter (recommended). For multi-channel images, it is possible to have a different degree for each channel by specifying degree as a vector or list.

Details

These functions include a thresholding option, unlike their non-batch processing counterparts which they wrap (i.e. `img_detrend_boxcar`, `img_detrend_exp` and `img_detrend_polynom`). This is because, when working interactively, it’s easy to threshold and then detrend, but for batch processing, it’s not so easy to efficiently do one after the other, so it’s nice to have that taken care of should you want it.

Value

Silently, a character vector of the paths to the detrended images.

Examples

```r
## Not run:
setwd(tempdir())
file.copy(
  c(
    system.file("extdata", "bleached.tif", package = "detrendr"),
    system.file("img", "2ch_ij.tif", package = "ijtiff")
  ),
  "."
)
dir_detrend_robinhood(thresh = "huang")
dir_detrend_boxcar(l = "auto", thresh = "tri", purpose = "FFS")
dir_detrend_exp(tau = "auto", thresh = "tri", purpose = "FCS")
dir_detrend_polynom(degree = "auto", thresh = "huang", purpose = "FFS")
## End(Not run)
```

detrended_img

Detrended image class.

Description

A `detrended_img` is a 4-dimensional array of positive integers in the style of an `ijtiff_img` (indexed by `img[y, x, channel, frame]`) which is the result of a detrending routine. It has 4 attributes:

- `method` The detrending method used. This must be one of "boxcar", "exponential" or "polynomial".
- `parameter` The value of the parameter used. This will be the `l`, `tau` or `degree` parameter for the respective methods.
- `auto` A boolean that is TRUE if the parameter was found automatically or FALSE if it was manually selected.
purpose Either "FCS" or "FFS" to denote whether the detrending was done for the purpose of fluorescence correlation spectroscopy or fluorescence fluctuation spectroscopy calculations respectively. purpose is not required for Robin Hood detrending.

Usage

detrended_img(img, method, parameter, auto, purpose = NULL)

Arguments

img The detrended image series. A 4-dimensional array of non-negative integers in the style of an ijtiff_img, or a 3-dimensional array of non-negative integers which represents a single channel of an ijtiff_img-style array (indexed by img[y,x,frame]).

method The method used. One of "robinhood", "boxcar", "exponential" or "polynomial".

parameter A number. The detrend parameter used. One per channel.

auto Logical. Was automatic detrending used? One per channel.

purpose Either "FCS" or "FFS". Was the image detrended for the purpose of doing FCS or FFS calculations? See detrending. purpose is not required for Robin Hood detrending.

Details

Sometimes when detrending, you can get slight negative values in the detrended image. These values should really just be zero, so this constructor function sets negative values of img to zero.

Value

An object of class detrended_img.

detrending Detrend images.

Description

Correct images for bleaching (or any other effect that introduces an unwanted trend) by detrending.

Usage

img_detrend_robinhood(img, swaps = "auto", quick = FALSE)

img_detrend_rh(img, swaps = "auto", quick = FALSE)

img_detrend_boxcar(img, l, purpose = c("FCS", "FFS"), parallel = FALSE)

img_detrend_exp(
  img,
tau,
cutoff = 0.05,
purpose = c("FCS", "FFS"),
parallel = FALSE
)

img_detrend_polynom(img, degree, purpose = c("FCS", "FFS"), parallel = FALSE)

Arguments

img  A 4-dimensional array in the style of an ijtiff_img (indexed by img[y, x, channel, frame]) or a 3-dimensional array which is a single channel of an ijtiff_img (indexed by img[y, x, frame]).

swaps  The number of swaps (giving of one count from rich to poor) to perform during the Robin Hood detrending. Set this to "auto" (the default) to use Nolan's algorithm to automatically find a suitable value for this parameter (recommended). For multi-channel images, it is possible to have a different swaps for each channel by specifying swaps as a vector or list.

quick  If FALSE (the default), the swap finding routine is run several times to get a consensus for the best parameter. If TRUE, the swap finding routine is run only once.

l  The length parameter for boxcar detrending. The size of the sliding window will be $2 \times l + 1$. This must be a positive integer. Set this to "auto" to use Nolan's algorithm to automatically find a suitable value for this parameter (recommended). For multi-channel images, it is possible to have a different l for each channel by specifying l as a vector or list.

purpose  What type of calculation do you intend to perform on the detrended image? If it is an FFS (fluorescence fluctuation spectroscopy) calculation (like number and brightness), choose 'FFS'. If it is an FCS (fluorescence correlation spectroscopy) calculation (like cross-correlated number and brightness or autocorrelation), choose 'FCS'. The difference is that if purpose is 'FFS', the time series is corrected for non-stationary mean and variance, whereas if purpose is 'FCS', the time series is corrected for non-stationary mean only. 'purpose' is not required for Robin Hood detrending.

parallel  Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use parallel = TRUE.

tau  The tau parameter for exponential filtering detrending. This must be a positive number. Set this to "auto" to use Nolan's algorithm to automatically find a suitable value for this parameter (recommended). For multi-channel images, it is possible to have a different tau for each channel by specifying tau as a vector or list.

cutoff  In exponential filtering detrending, for the weighted average, every point gets a weight. This can slow down the computation massively. However, many of the weights will be approximately zero. With cutoff, we say that any point with weight less than or equal to cutoff times the maximum weight may be ignored; so with cutoff = 0.05, any weight less than 5\% of its value should not be set to anything else without good reason.
degree

The degree of the polynomial to use for the polynomial detrending. This must be a positive integer. Set this to "auto" to use Nolan’s algorithm to automatically find a suitable value for this parameter (recommended). For multi-channel images, it is possible to have a different degree for each channel by specifying degree as a vector or list.

Details

There are 4 detrending methods available: Robin Hood, boxcar, exponential filtering and polynomial. Robin Hood is described in Nolan et al., 2018. The others are described in Nolan et al., 2017.

- **Robin Hood** is a method whereby counts are taken from frames with higher mean intensity and given directly to frames of lower intensity.
- **Boxcar** detrending with parameter \( l \) is a moving average detrending method using a sliding window of size \( 2l + 1 \).
- **Exponential filtering** detrending is a moving weighted average method where for parameter \( \tau \) the weights are calculated as \( \exp(-t/\tau) \) where \( t \) is the distance from the point of interest.
- **Polynomial** detrending works by fitting a polynomial line to a series of points and then correcting the series to remove the trend detailed by this polynomial fit.

Value

The detrended image, an object of class detrended_img.

References


Examples

```r
## Not run:
## These examples are not run on CRAN because they take too long.
## You can still try them for yourself.

img <- ijtiff::read_tif(system.file("extdata", "bleached.tif",
                            package = "detrendr")
corrected <- img_detrend_rh(img)
corrected10 <- img_detrend_boxcar(img, "auto", purpose = "fcs", parallel = 2)
corrected50 <- img_detrend_boxcar(img, 10, purpose = "fcs", parallel = 2)
corrected10 <- img_detrend_boxcar(img, 50, purpose = "fcs", parallel = 2)
corrected <- img_detrend_exp(img, "auto", purpose = "ffs", parallel = 2)
corrected10 <- img_detrend_exp(img, 10, purpose = "ffs", parallel = 2)
corrected50 <- img_detrend_exp(img, 50, purpose = "ffs", parallel = 2)
corrected <- img_detrend_polynom(img, "auto", purpose = "ffs", parallel = 2)
corrected2 <- img_detrend_polynom(img, 2, purpose = "ffs", parallel = 2)
```
## detrendr

*detrendr: Image detrending in R.*

### Description

The `detrendr` package gives functions for detrending images, most often used for preprocessing in fluorescence fluctuation and correlation spectroscopy (FFS and FCS).

### References


---

### mean_frames

*Get the sums/means of frames in a 3-dimensional array.*

### Description

Frame $i$ of a 3-dimensional array `arr3d` is defined as `arr3d[, , i]`.

### Usage

```r
mean_frames(arr3d, na.rm = FALSE)
sum_frames(arr3d, na.rm = FALSE)
```

### Arguments

- **arr3d**
  - A 3-dimensional numeric array.
- **na_rm**
  - Do you want NA values to be excluded from calculations?

### Value

A numeric vector.

### Examples

```r
a <- array(seq_len(2^3), dim = rep(2, 3))
sum_frames(a)
mean_frames(a)
```
pillar-stats

Get the sums/means/medians/variances of pillars of an ijtiff_img-style array.

Description

For an ijtiff_img-style array img (indexed as img[y, x, channel, frame]), pillar xy of channel ch is defined as img[y, x, ch, ] for each channel.

Usage

```
sum_pillars(img, parallel = FALSE)
mean_pillars(img, parallel = FALSE)
median_pillars(img, parallel = FALSE)
var_pillars(img, parallel = FALSE)
```

Arguments

- `img` A 4-dimensional array in the style of an ijtiff_img (indexed by img[y, x, channel, frame]) or a 3-dimensional array which a single channel of an ijtiff_img (indexed by img[y, x, frame]).
- `parallel` Would you like to use multiple cores to speed up this function? If so, set the number of cores here, or to use all available cores, use parallel = TRUE.

Value

An ijtiff_img-style array arr with one frame. arr[y, x, ch, 1] is equal to mean(img[y, x, ch, ]), median(img[y, x, ch, ]), or var(img[y, x, ch, ]).

Examples

```
aaa <- array(seq_len(2^4), dim = rep(2, 4)) # a 2-channel, 2-frame array
sum_pillars(aaa)
mean_pillars(aaa)
median_pillars(aaa)
var_pillars(aaa)
```
rtoboxes

Randomly draw balls from boxes.

Description

Given a number of boxes with a specified number of balls in each, randomly draw a number of balls from these boxes, recording how many balls was drawn from each. An empty box cannot be drawn from.

Usage

rtfromboxes(n, balls, weights = NULL)

Arguments

n
A natural number. The number of balls to draw.
balls
A vector of natural numbers. The number of balls in each box to begin with.
weights
A non-negative numeric vector the same length as balls. The relative probabilities of drawing a ball from each box. Default is each box is equally likely to be drawn from.

Value

A vector of natural numbers with the same length as balls. The number of balls drawn from each box.

See Also

rtoboxes

Examples

balls <- 1:10
rtfromboxes(40, balls)
rtfromboxes(40, balls, weights = c(rep(1, 9), 0))

rtoboxes

Randomly place balls in boxes.

Description

Given a number of boxes, randomly distribute \( n \) balls into these boxes.

Usage

rtoboxes(n, boxes, weights = NULL, capacities = NULL)
rtoboxes

Arguments

\begin{itemize}
\item \texttt{n} A natural number. The number of balls to put into the boxes.
\item \texttt{boxes} A natural number. The number of boxes.
\item \texttt{weights} A non-negative numeric vector. The relative probabilities of putting a ball in each box. Default is each box is equally likely.
\item \texttt{capacities} A vector of natural numbers. The capacity of each box. Default is each box has infinite capacity.
\end{itemize}

Value

A vector of natural numbers with the same length as \texttt{boxes}. The number of balls placed in each box.

See Also

rfromboxes

Examples

\begin{verbatim}
rtoboxes(30, 7)
rtoboxes(30, 7, capacities = c(rep(1, 3), rep(7, 4)))
rtoboxes(30, 7,
  capacities = c(rep(1, 3), rep(70, 4)),
  weights = c(rep(0.1, 6), 1)
)
\end{verbatim}
Index

apply_on_pillars, 2
autothresholdr::mean_stack_thresh(), 9
best_degree, 3
best_l, 4
best_swaps, 5
best_tau, 6
brightness_pillars, 7
detrend-directory, 8
detrended_img, 10, 10, 13
detrending, 11, 11
detrendr, 14
detrendr-package (detrendr), 14
detrendr-package (detrendr), 14
detrendr-package (detrendr), 14
detrendr-package (detrendr), 14
detrendr-package (detrendr), 14
ijtiff_img, 3–7, 10–12, 15
img_detrend_boxcar, 10
img_detrend_boxcar (detrending), 11
img_detrend_exp, 10
img_detrend_exp (detrending), 11
img_detrend_polynom, 10
img_detrend_polynom (detrending), 11
img_detrend_robinhood, 11
img_detrend_robinhood (detrending), 11
mean_frames, 14
mean_pillars (pillar-stats), 15
median_pillars (pillar-stats), 15
pillar-stats, 15
rfromboxes, 16
rtoboxes, 16
sum_frames (mean_frames), 14
sum_pillars (pillar-stats), 15
var_pillars (pillar-stats), 15
var_pillars (pillar-stats), 15