

Package ‘scbursts’

July 23, 2025

Version 1.6

Date 2019-06-24

Title Single Channel Bursts Analysis

Description Provides tools to import and export from several existing pieces of ion-channel analysis software such as 'TAC', 'QUB', 'SCAN', and 'Clampfit', implements procedures such as dwell-time correction and defining bursts with a critical time, and provides tools for analysis of bursts, such as tools for sorting and plotting.

Author Blair Drummond [aut], Mathieu Dextraze [ctb]

Maintainer Blair Drummond <blair.robert.drummond@gmail.com>

License LGPL-2.1

LazyData true

RoxygenNote 6.1.1

Suggests knitr, rmarkdown, tinytex

VignetteBuilder knitr

Imports readxl, tibble

NeedsCompilation no

Repository CRAN

Date/Publication 2019-07-06 15:30:03 UTC

Contents

bursts.check_subconductance	3
bursts.conductance_states	3
bursts.copy	4
bursts.defined_by_tcrit	4
bursts.get_gaps	5
bursts.impose_deadtime	6
bursts.modify_conductance	7
bursts.pclosed	8
bursts.popens	8
bursts.recombine	9

bursts.remove_first_and_last	10
bursts.select	10
bursts.sort	11
bursts.space_out	12
bursts.start_times_update	13
bursts.subconductance_as	13
clampfit.read	14
cplot.conductance_hist	15
cplot.log_root_axes	15
cplot.pclosed_ts	16
cplot.popen_ts	17
dwt.read	17
dwt.write	18
evt.extract_header	19
evt.from_dwells	19
evt.read	20
evt.to_dwells	21
evt.write	21
hst.extract_header	22
hst.read	23
hst.write	23
risetime.correct_gaussian	24
scan.read	25
segment.check_subconductance	26
segment.closed_dwells	26
segment.conductance_states	27
segment.consecutives_to_dwells	28
segment.copy	28
segment.count_closed	29
segment.count_dwells	29
segment.count_open	30
segment.create	31
segment.duration	32
segment.dwells_by_conductance	32
segment.dwells_by_conductance_range	33
segment.impose_deadtime	34
segment.modify_conductance	34
segment.name	35
segment.open_dwells	36
segment.pclosed	37
segment.pconductance	37
segment.pconductance_range	38
segment.popen	39
segment.seg	39
segment.start_time	40
segment.subconductance_as	41
segment.verify	41
util.basename	42

`bursts.check_subconductance`
Check if segment contains subconductive states

Description

Check if segment contains subconductive states

Usage

```
bursts.check_subconductance(bursts)
```

Arguments

`bursts` The list of all bursts

Value

True if it contains an conductance other than 0 or 1, False otherwise.

Examples

```
infile <- system.file("extdata", "example4.dwt", package = "scbursts")
dwell <- dwt.read(infile)
dwell_c <- risetime.correct_gaussian(Tr=35.0052278, dwell, units="us")
bursts <- bursts.defined_by_tcrit(dwell_c, 100, units="ms")

bursts.check_subconductance(bursts)
```

`bursts.conductance_states`
Return a list of all the (sub)conductance states.

Description

Return a list of all the (sub)conductance states.

Usage

```
bursts.conductance_states(bursts)
```

Arguments

`bursts` The list of all bursts

Value

a list of all the (sub)conductance states.

Examples

```
infile <- system.file("extdata", "example4.dwt", package = "scbursts")
dwells <- dwt.read(infile)
dwells_c <- risetime.correct_gaussian(Tr=35.0052278, dwells, units="us")
bursts <- bursts.defined_by_tcrit(dwells_c, 100, units="ms")
```

```
bursts.conductance_states(bursts)
```

bursts.copy	<i>Copy a list of bursts (by value)</i>
-------------	---

Description

Copy a list of bursts (by value)

Usage

```
bursts.copy(bursts)
```

Arguments

bursts	bursts to copy
--------	----------------

Value

A copy of the bursts.

bursts.defined_by_tcrit	<i>Divide a recording into bursts defined by a critical time.</i>
-------------------------	---

Description

Split segment at long pauses, dividing the segment into multiple -shorter- segments (which are the bursts), Along with the interburst closings, which are referred to as "gaps". (Default time units are seconds)

Usage

```
bursts.defined_by_tcrit(segments, t_crit, units = "s")
```

Arguments

segments	A segment or multiple segments with \$states and \$dwells. NOTE: separate segments will remain split, regardless of why they were originally divided.
t_crit	Critical time at which to divide bursts (in seconds by default)
units	what unit the critical time is in ('s','ms','us', or 'ns')

Value

bursts. Which is a list of segments starting and ending in 1 states (open dwell)

Examples

```
infile <- system.file("extdata", "example1_tac.evt", package = "scbursts")
transitions <- evt.read(infile)
dwells <- evt.to_dwells(transitions)
dwells_c <- risetime.correct_gaussian(Tr=35.0052278, dwells, units="us")

bursts <- bursts.defined_by_tcrit(dwells_c, 100, units="ms")
head(bursts[[1]])
```

bursts.get_gaps *Get the gaps between bursts.*

Description

Extract vector of gaps from the bursts. This is done using the start_time attribute, which is mostly hidden in the data. (The gaps at the ends may have length 0)

Usage

```
bursts.get_gaps(bursts)
```

Arguments

bursts	The list of segments
--------	----------------------

Value

A vector of N+1 gaps for N bursts times

Examples

```
infile <- system.file("extdata", "example1_tac.evt", package = "scbursts")
transitions <- evt.read(infile)
dwells <- evt.to_dwells(transitions)
dwells_c <- risetime.correct_gaussian(Tr=35.0052278, dwells, units="us")

bursts <- bursts.defined_by_tcrit(dwells_c, 100, units="ms")
gaps <- bursts.get_gaps(bursts)

head(gaps)
```

bursts.impose_deadtime

Imposes a deadtime to each segment in a burst.

Description

The user specifies a deadtime in microseconds. The function applies `segment.impose_deadtime` to each segment in the burst. (See `segment.impose_deadtime` for details.)

Usage

```
bursts.impose_deadtime(bursts, deadtime)
```

Arguments

`bursts` a burst containing segments of dwells and states.
`deadtime` the briefest possible event in microseconds.

Value

A modified copy of the original burst

Examples

```
infile <- system.file("extdata", "example4.dwt", package = "scbursts")
dwells <- dwt.read(infile)
dwells_c <- risetime.correct_gaussian(Tr=35.0052278, dwells, units="us")
bursts <- bursts.defined_by_tcrit(dwells_c, 100, units="ms")

bursts_d <- bursts.impose_deadtime(bursts, deadtime=0.01)
```

bursts.modify_conductance

Transform the conductance states according to a user-defined function of conductance level.

Description

Transform the conductance states according to a user-defined function of conductance level.

Usage

```
bursts.modify_conductance(bursts, fun)
```

Arguments

bursts	the list of segments
fun	a function on conductance levels

Value

A modified copy of the original bursts

Examples

```
infile <- system.file("extdata", "example4.dwt", package = "scbursts")
dwells <- dwt.read(infile)
dwells_c <- risetime.correct_gaussian(Tr=35.0052278, dwells, units="us")
bursts <- bursts.defined_by_tcrit(dwells_c, 100, units="ms")

### Collapse into three subconductance states
fun <- function(amp) {
  if (amp < 0.3)
    return(0)
  else if (amp >= 0.3 && amp < 0.6)
    return(0.5)
  else
    return(1)
}

bursts_d <- bursts.modify_conductance(bursts, fun)
```

bursts.pclosededs *Return pclosededs of every burst.*

Description

Return pclosededs of every burst.

Usage

```
bursts.pclosededs(bursts)
```

Arguments

bursts The list of all bursts

Value

The pclosed values

Examples

```
infile <- system.file("extdata", "example1_qub.dwt", package = "scbursts")
dwells <- dwt.read(infile)
dwells_c <- risetime.correct_gaussian(Tr=35.0052278, dwells, units="us")
bursts <- bursts.defined_by_tcrit(dwells_c, 100, units="ms")

pclosededs <- bursts.popens(bursts)
hist(pclosededs)
```

bursts.popens *Return popens of every burst.*

Description

Return popens of every burst.

Usage

```
bursts.popens(bursts)
```

Arguments

bursts The list of all bursts

Value

The popen values

Examples

```
infile <- system.file("extdata", "example1_qub.dwt", package = "scbursts")
dwell <- dwt.read(infile)
dwell_c <- risetime.correct_gaussian(Tr=35.0052278, dwell, units="us")

bursts <- bursts.defined_by_tcrit(dwell_c, 100, units="ms")

popen <- bursts.popen(bursts)
hist(popen)
```

bursts.recombine	<i>Combine bursts into one recording (with obvious spaces between them).</i>
------------------	--

Description

From a list of segments, return the concatenated segment containing all bursts. Inverse of functions like `bursts.defined_by_tcrit`

Usage

```
bursts.recombine(bursts)
```

Arguments

`bursts` The list of all bursts

Value

The segment containing all bursts.

Examples

```
infile <- system.file("extdata", "example1_qub.dwt", package = "scbursts")
dwell <- dwt.read(infile)
dwell_c <- risetime.correct_gaussian(Tr=35.0052278, dwell, units="us")

bursts <- bursts.defined_by_tcrit(dwell_c, 100, units="ms")

# This is a single segment!
record <- bursts.recombine(bursts)

# Which means you can do stuff like this
open_dwell <- segment.open_dwell(bursts.recombine(bursts))
```

```
bursts.remove_first_and_last
```

Remove the first and last burst from the list.

Description

Remove the first and last burst from the list.

Usage

```
bursts.remove_first_and_last(bursts)
```

Arguments

bursts The list of all bursts

Value

A shorter list of bursts

Examples

```
infile <- system.file("extdata", "example1_tac.evt", package = "scbursts")
transitions <- evt.read(infile)
dwellings <- evt.to_dwellings(transitions)
dwellings_c <- risetime.correct_gaussian(Tr=35.0052278, dwellings, units="us")

bursts <- bursts.defined_by_tcrit(dwellings_c, 100, units="ms")

# If there seem to be bad bursts at the ends
bursts <- bursts.remove_first_and_last(bursts)
```

```
bursts.select
```

From a list of bursts, extract those that interest you by passing a selecting function.

Description

From a list of bursts, extract those that interest you by passing a selecting function.

Usage

```
bursts.select(bursts, func, one_file = FALSE)
```

Arguments

bursts	The list of all bursts
func	A function of a segment that returns either TRUE or FALSE
one_file	TRUE or FALSE: Return a single file to write to disk, or a list of bursts. The one_file will return a file with all unselected bursts zeroed out.

Value

A shorter list of bursts OR if one_file is passed one segment with zeros where the other bursts might have been originally. Defaults to FALSE.

Examples

```
high_popen <- function (seg) {
  segment.popen(seg) > 0.7
}

infile <- system.file("extdata", "example1_qub.dwt", package = "scbursts")
dwellings <- dwt.read(infile)
dwellings_c <- risetime.correct_gaussian(Tr=35.0052278, dwellings, units="us")

bursts <- bursts.defined_by_tcrit(dwellings_c, 100, units="ms")

subset <- bursts.select(bursts, high_popen)

# To export to one .dwt file
subset_f <- bursts.select(bursts, high_popen, one_file=TRUE)
```

bursts.sort	<i>Order a list of bursts by some function. For instance, popen.</i>
-------------	--

Description

Order a list of bursts by some function. For instance, popen.

Usage

```
bursts.sort(bursts, func, reverse = FALSE)
```

Arguments

bursts	The list of all bursts
func	A function of a segment that returns a numeric value
reverse	By default, return in ascending order. Use reverse=TRUE to change that.

Value

A list sorted by func. By default in ascending order (unless reversed)

Examples

```
infile <- system.file("extdata", "example1_qub.dwt", package = "scbursts")
dwell <- dwt.read(infile)
dwell_c <- risetime.correct_gaussian(Tr=35.0052278, dwell, units="us")
bursts <- bursts.defined_by_tcrit(dwell_c, 100, units="ms")

# A sorted list of bursts.
sorted <- bursts.sort(bursts, segment.popen)

# You can also write your own functions. If you want P(Open) ~ P(Closed)
variance_fun <- function (seg) {
  # Any function that maps a segment to a number works.
  return( segment.popen(seg) * segment.pclosed(seg) )
}

weird_sort <- bursts.sort(bursts, variance_fun)
```

bursts.space_out	<i>Artificially add amount of time between bursts (in absence of recording information).</i>
------------------	--

Description

Given a list of segments separated by an unknown amount of time, one may want to space the segments by some amount of time, so that they can be plotted. This function takes a separating factor, and splits up the segments by either that factor (in seconds), or that many multiples of the largest observed dwell.

Usage

```
bursts.space_out(segments, sep_factor = 1000)
```

Arguments

segments	The segments to space out
sep_factor	the factor by which to separate the segments. Either the factor in seconds, or a multiple of the longest observed dwell.

Value

The segments again, but with modified meta-data.

Examples

```
infile <- system.file("extdata", "example_multiple_segments.dwt", package = "scbursts")
dwells <- dwt.read(infile)

# Still a list, but the meta-data is fixed
spaced_records <- bursts.space_out(dwells, sep_factor=1000)

# Combine them, and they'll be nicely spaced out.
single_record <- bursts.recombine(spaced_records)

# You can now plot that single_record using one of the plot functions.
```

```
bursts.start_times_update
```

(DON'T USE THIS) Fix meta-data of bursts.

Description

YOU PROBABLY WON'T EVER HAVE TO CALL THIS DIRECTLY. Attach the meta-data to each segment saying when it began. It interleaves the durations of the bursts and gaps, and assigns the sum of those durations up to a point as the starting time.

Usage

```
bursts.start_times_update(bursts, gaps)
```

Arguments

bursts	List of segments
gaps	vector of gap times.

Value

A list of segments, one per burst, with updated start_times

```
bursts.subconductance_as
```

Imposes a fixed conductance level (0 or 1) to all dwells with subconductance levels to each segment in a burst

Description

The user specifies the desired level ('open' or 'closed'). The function applies segment.subconductance_as to each segment in the burst. (See segment.subconductance_as for details.)

Usage

```
bursts.subconductance_as(bursts, level)
```

Arguments

```
bursts      the list of segments
level      either 'open' or 'closed'
```

Value

A modified copy of the original burst

Examples

```
infile <- system.file("extdata", "example4.dwt", package = "scbursts")
dwells <- dwt.read(infile)
dwells_c <- risetime.correct_gaussian(Tr=35.0052278, dwells, units="us")
bursts <- bursts.defined_by_tcrit(dwells_c, 100, units="ms")

bursts_d <- bursts.subconductance_as(bursts, "open")
```

clampfit.read	<i>Read a .xlsx file output from clampfit</i>
---------------	---

Description

Read a .xlsx file output from clampfit. Result is a list of "segments", which is a dataframe extra data. See "segment" for more details. Converts millisecond dwells to seconds.

Usage

```
clampfit.read(filename, separating_factor = 1000, header = FALSE)
```

Arguments

```
filename      Filename to read from
separating_factor
               In lieu of a known time between segments, separate with a multiple of the longest
               dwell.
header        Does the file include a header?
```

Value

A list of bursts (possibly a singleton)

Examples

```
infile <- system.file("extdata", "example1_clampfit.xlsx", package = "scbursts")
dwells <- clampfit.read(infile)
head(dwells)
```

```
cplot.conductance_hist
```

Histogram of Conductance States

Description

Histogram of Conductance States

Usage

```
cplot.conductance_hist(bursts, ...)
```

Arguments

bursts	List of multiple segments
...	other arguments passed to histogram

Examples

```
infile <- system.file("extdata", "example4.dwt", package = "scbursts")
dwells <- dwt.read(infile)
dwells_c <- risetime.correct_gaussian(Tr=35.0052278, dwells, units="us")
bursts <- bursts.defined_by_tcrit(dwells_c, 100, units="ms")

cplot.conductance_hist(bursts, main="example4.dwt conductance state histogram")
```

```
cplot.log_root_axes
```

Add log-root axes to histogram plot

Description

Add log-root axes to histogram plot

Usage

```
cplot.log_root_axes(points)
```

Arguments

points	The data to plot
--------	------------------

Examples

```
infile <- system.file("extdata", "example1_qub.dwt", package = "scbursts")
dwells <- dwt.read(infile)
dwells_c <- risetime.correct_gaussian(Tr=35.0052278, dwells, units="us")
bursts <- bursts.defined_by_tcrit(dwells_c, 100, units="ms")

open_dwells <- segment.open_dwells(bursts.recombine(bursts))
hist(log10(open_dwells), axes=FALSE, breaks=30)
cplot.log_root_axes(open_dwells)
```

cplot.pclosed_ts *Plot Time Series (ts) of P(Closed).*

Description

Plot Time Series (ts) of P(Closed).

Usage

```
cplot.pclosed_ts(bursts, main = "P(Closed) Time Series", ...)
```

Arguments

bursts	List of multiple segments
main	The title of the plot.
...	Options to pass to plot

Examples

```
infile <- system.file("extdata", "example1_qub.dwt", package = "scbursts")
dwells <- dwt.read(infile)
dwells_c <- risetime.correct_gaussian(Tr=35.0052278, dwells, units="us")
bursts <- bursts.defined_by_tcrit(dwells_c, 100, units="ms")

cplot.pclosed_ts(bursts, main="P(Closed) Time Series, 2018-09-20")
```

cplot.popen_ts	<i>Plot Time Series (ts) of P(Open).</i>
----------------	--

Description

Plot Time Series (ts) of P(Open).

Usage

```
cplot.popen_ts(bursts, main = "P(Open) Time Series", ...)
```

Arguments

bursts	List of multiple segments
main	The title of the plot.
...	Options to pass to plot

Examples

```
infile <- system.file("extdata", "example1_qub.dwt", package = "scbursts")
dwells <- dwt.read(infile)
dwells_c <- risetime.correct_gaussian(Tr=35.0052278, dwells, units="us")
bursts <- bursts.defined_by_tcrit(dwells_c, 100, units="ms")

cplot.popen_ts(bursts, "P(Open) Time Series, 2018-09-20")
```

dwt.read	<i>Read a .dwt file.</i>
----------	--------------------------

Description

Read a .dwt file. Result is a list of "segments", which is a dataframe extra data. See "segment" for more details. Converts millisecond dwells to seconds.

Usage

```
dwt.read(filename, separating_factor = 1000)
```

Arguments

filename	Filename to read from
separating_factor	In lieu of a known time between segments, separate with a multiple of the longest dwell.

Value

A list of bursts (possibly a singleton)

Examples

```
infile <- system.file("extdata", "example1_tac.evt", package = "scbursts")
transitions <- evt.read(infile)
dwells <- evt.to_dwells(transitions)

dwt.write(dwells, file=file.path(tempdir(), "dwells.dwt"))

# Quit R, come back the next day
## Not run:
dwells <- dwt.read("dwells.dwt")

## End(Not run)
```

dwt.write	<i>Write a dwt file to disk. Writes DOS line endings. Dwells are in milliseconds</i>
-----------	--

Description

Write a dwt file to disk. Writes DOS line endings. Dwells are in milliseconds

Usage

```
dwt.write(segments, file = "", seg = 1, append = FALSE)
```

Arguments

segments	A segment or multiple segments with \$dwells and \$states
file	Filename to write to
seg	Segment number to write in .dwt header.
append	Add ot the end of a file or overwrite? (defaults to false)

Examples

```
infile <- system.file("extdata", "example1_tac.evt", package = "scbursts")
transitions <- evt.read(infile)
dwells <- evt.to_dwells(transitions)

dwt.write(dwells, file=file.path(tempdir(), "dwells.dwt"))
```

evt.extract_header *Extract header from evt file.*

Description

Extract header from evt file.

Usage

evt.extract_header(filename)

Arguments

filename The filename

Value

A string containing the header

Examples

```
infile <- system.file("extdata", "example1_tac.evt", package = "scbursts")

# Get Dwells
transitions <- evt.read(infile)
dwells <- evt.to_dwells(transitions)
dwells_c <- risetime.correct_gaussian(Tr=35.0052278, dwells, units="us")

# Get Header
header <- evt.extract_header(infile)

evt.write(dwells_c, header=header, file=file.path(tempdir(), "fixed_example1_tac.evt"))
```

evt.from_dwells *Converts dwell durations to absolute transition times.*

Description

Converts dwell durations to absolute transition times.

Usage

evt.from_dwells(segments)

Arguments

segments A segment or multiple segments

Value

A dataframe or multiple dataframes of states and transition times

Examples

```
dwells_file <- system.file("extdata", "example1_qub.dwt", package = "scbursts")
dwells <- dwt.read(dwells_file)

transitions <- evt.from_dwells(dwells)
```

evt.read

Read a .evt file to a table. Times are in seconds

Description

Read a .evt file to a table. Times are in seconds

Usage

```
evt.read(filename)
```

Arguments

filename The filename

Value

A list of tables with columns "states" and "times". Each table corresponds to a contiguous segment from a recording.

Examples

```
# import some of the data included with the package
infile <- system.file("extdata", "example1_tac.evt", package = "scbursts")
transitions <- evt.read(infile)

head(transitions[[1]])
```

evt.to_dwells *Calculate pulse lengths. Converts transition times to dwell durations.*

Description

Calculate pulse lengths. Converts transition times to dwell durations.

Usage

```
evt.to_dwells(tables)
```

Arguments

tables Either a single table or a list of tables with columns "states" and "times"

Value

A segment or a list of segments with one less row, where each row represents pulse in state 0 (closed dwell) of duration 0.51231, instead of the time at which the state transitioned.

Examples

```
infile <- system.file("extdata", "example1_tac.evt", package = "scbursts")
transitions <- evt.read(infile)
dwells <- evt.to_dwells(transitions)
head(dwells[[1]])
```

evt.write *Write bursts to a .evt file.*

Description

Write bursts to a .evt file.

Usage

```
evt.write(segments, filename = "", header = NULL)
```

Arguments

segments A segment or list of segments to write to filename
filename The filename
header The header information for the evt file, if available

Examples

```
infile <- system.file("extdata", "example1_tac.evt", package = "scbursts")

# Get Dwells
transitions <- evt.read(infile)
dwells <- evt.to_dwells(transitions)
dwells_c <- risetime.correct_gaussian(Tr=35.0052278, dwells, units="us")

# Get Header
header <- evt.extract_header(infile)

evt.write(dwells_c, header=header, file=file.path(tempdir(), "fixed_example1_tac.evt"))
```

hst.extract_header	<i>Extract header from hst file.</i>
--------------------	--------------------------------------

Description

Extract header from hst file.

Usage

```
hst.extract_header(filename)
```

Arguments

filename	The filename
----------	--------------

Value

A string containing the header

Examples

```
# import some of the data included with the package
infile <- system.file("extdata", "example1_hst.hst", package = "scbursts")

open_table <- hst.read(infile, extract="open")
closed_table <- hst.read(infile, extract="closed")
header <- hst.extract_header(infile)

# Make adjustments to the histogram, if you wish
hst.write(open_table, closed_table, file=file.path(tempdir(), "output_hist.hst"), header=header)
```

hst.read	<i>Read a MIL ".hst" file to a table.</i>
----------	---

Description

Read a MIL ".hst" file to a table. By default these files are in $\log_{10}(\text{Milliseconds})\text{-}\sqrt{\text{Freq}}$, but unless "raw" is set to TRUE, this function returns a table containing Seconds-Freq

Usage

```
hst.read(filename, extract = "open", raw = FALSE)
```

Arguments

filename	The filename
extract	Extract either "open" or "closed" histogram
raw	Data is given as $\log_{10}(\text{milliseconds})\text{-}\sqrt{\text{Freq}}$. Setting raw=FALSE yields output as Seconds-Frequency

Value

A tables with columns "bin", "freq" and "fit".

Examples

```
# import some of the data included with the package
infile <- system.file("extdata", "example1_hst.hst", package = "scbursts")
open_hst <- hst.read(infile, extract="open")
closed_hst <- hst.read(infile, extract="closed")

head(open_hst)
head(closed_hst)
```

hst.write	<i>Write bursts to a $\log_{10}(\text{ms})\text{-}\sqrt{\text{Frequency}}$.hst file from open and closed tables.</i>
-----------	--

Description

Write bursts to a $\log_{10}(\text{ms})\text{-}\sqrt{\text{Frequency}}$.hst file from open and closed tables.

Usage

```
hst.write(open_hist, closed_hist, file = "", header = NULL,
          fromraw = FALSE)
```

Arguments

open_hist	The table (bin,freq,fit) for open times
closed_hist	The table (bin,freq,fit) for closed times
file	The filename
header	The header info
fromraw	Unless FALSE, assume we need to write a $\log_{10}(\text{milliseconds})\text{-}\sqrt{\text{Frequency}}$ plot

Examples

```
infile <- system.file("extdata", "example1_hst.hst", package = "scbursts")

open = hst.read(infile, extract="open")
closed = hst.read(infile, extract="closed")
header = hst.extract_header(infile)

### Do stuff
hst.write(open, closed, file=file.path(tempdir(), "new_histogram.hst"), header=header)
```

```
risetime.correct_gaussian
```

Undo the effect of the gaussian filter.

Description

Undo the effect of the gaussian filter. See section 4.1.1 of Colquhoun and Sigworth, "Fitting and Analysis of Single-Channel segments". NOTE: This is potentially problematic, in that this unfiltering lengthens every dwell. A less naive algorithm would take into account the influence of the surroundings, as they impact the effects of the filter.

Usage

```
risetime.correct_gaussian(Tr, segments, units = "s")
```

Arguments

Tr	Rise time of the filter in (us)
segments	A segment or multiple segments with \$states and \$dwells to correct.
units	What unit the risetime is input in (defaults to seconds)

Value

A Segment or multiple segments with corrected risetimes.

Examples

```
infile <- system.file("extdata", "example1_tac.evt", package = "scbursts")
transitions <- evt.read(infile)
dwells <- evt.to_dwells(transitions)

dwells_c <- risetime.correct_gaussian(Tr=35.0052278, dwells, units="us")
```

scan.read	<i>Read a scan results text file. scan.read returns a 1 segment list Reads in scan results and puts them in the same format as the output of dwt.read. See 'dwt', and 'segment' for more information.</i>
-----------	---

Description

Data is in seconds.

Usage

```
scan.read(filename, separating_factor = 1000)
```

Arguments

filename the file name to read from.

separating_factor In lieu of a known time between segments, seperate with a multiple of the longest dwell.

Value

A list of recording segments from the scan file

Examples

```
infile <- system.file("extdata", "example1_scan.txt", package = "scbursts")
record <- scan.read(infile)
head(record)
```

```
segment.check_subconductance
```

Check if segment contains subconductive states

Description

Check if segment contains subconductive states

Usage

```
segment.check_subconductance(segment)
```

Arguments

segment The dwells and states table

Value

True if it contains an conductance other than 0 or 1, False otherwise.

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 0.2, 0, 1, 0, 0.5, 0, 0.7, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

segment.check_subconductance(my_burst)
```

```
segment.closed_dwells Extract closed dwells.
```

Description

Extract closed dwells.

Usage

```
segment.closed_dwells(segment)
```

Arguments

segment the segment object

Value

the closed dwells

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

closed_dwells <- segment.closed_dwells(my_burst)
head(closed_dwells)
```

segment.conductance_states

Return a list of all the (sub)conductance states.

Description

Return a list of all the (sub)conductance states.

Usage

```
segment.conductance_states(segment)
```

Arguments

segment The dwells and states table

Value

a list of all the (sub)conductance states.

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 0.2, 0, 1, 0, 0.5, 0, 0.7, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

segment.conductance_states(my_burst)
```

```
segment.consecutives_to_dwells
```

Collapses a segment into dwells with alternating conductance levels.

Description

Segments may contain consecutive dwells with the same conductance level. `consecutives_to_dwells` sums together all consecutive dwells with the same conductance level. The result is a segment containing dwells that alternate in conductance level (i.e. 1,0,1,0,1,...)

Usage

```
segment.consecutives_to_dwells(segment)
```

Arguments

segment	The dwells and states table
---------	-----------------------------

Value

A modified copy of the original segment

```
segment.copy
```

Copy a segment

Description

This is a low-level function, mostly for use internally by other functions. There aren't many reasons to use this.

Usage

```
segment.copy(segment)
```

Arguments

segment	The segment to copy
---------	---------------------

Value

A duplicate identical content.

segment.count_closed *Extract number of closed dwells. In the case of subconductive states, a dwell is only closed if the conductance is exactly zero.*

Description

Extract number of closed dwells. In the case of subconductive states, a dwell is only closed if the conductance is exactly zero.

Usage

```
segment.count_closed(segment)
```

Arguments

segment the segment object

Value

number of closed dwells

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

segment.count_closed(my_burst)
```

segment.count_dwells *Extract number of dwells in segment.*

Description

Extract number of dwells in segment.

Usage

```
segment.count_dwells(segment)
```

Arguments

segment the segment object

Value

number of dwells

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

segment.count_dwells(my_burst)
```

segment.count_open	<i>Extract number of open dwells. In the case of subconductive states, count the number of non-zero states.</i>
--------------------	---

Description

Extract number of open dwells. In the case of subconductive states, count the number of non-zero states.

Usage

```
segment.count_open(segment)
```

Arguments

segment the segment object

Value

number of open dwells

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

segment.count_open(my_burst)
```

segment.create	Create a "segment" object
----------------	---------------------------

Description

This is a low-level function, mostly for use internally by other functions. There aren't many reasons to use this. Create object containing table data and metadata. The object can be used as a dataframe, and the metadata can be accessed with the functions: `segment.seg`, `segment.start_time`, `segment.filename`

Usage

```
segment.create(states, dwells, seg = 1, start_time = 0,  
              name = "burst", ignore_errors = FALSE)
```

Arguments

<code>states</code>	a vector of states
<code>dwells</code>	a vector of dwell durations (same length as states)
<code>seg</code>	The segment number. Defaults to 1
<code>start_time</code>	When the dwells began. Defaults to 0
<code>name</code>	Suffix-less version of the original filename. 60uM.dwt -> '60uM'
<code>ignore_errors</code>	Do not report faulty segments (not many reasons to do this)

Value

The segment object: A dataframe with extra metadata.

Examples

```
# It's more likely that you created states or dwells with some function  
states <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1)  
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)  
my_burst <- segment.create(states, dwells, seg=1, start_time=0, name="example_segment")  
  
segment.name(my_burst)
```

segment.duration	<i>Get duration of a segment.</i>
------------------	-----------------------------------

Description

Get duration of a segment.

Usage

```
segment.duration(segment)
```

Arguments

segment	the segment object
---------	--------------------

Value

the duration

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

segment.duration(my_burst)
```

segment.dwells_by_conductance	<i>Extract dwells in conductance range. lower <= x <= upper</i>
-------------------------------	---

Description

Extract dwells in conductance range. lower <= x <= upper

Usage

```
segment.dwells_by_conductance(segment, level)
```

Arguments

segment	the segment object
level	The conductance to extract

Value

the dwells in a given range

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 0.2, 0, 1, 0, 0.5, 0, 0.7, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

half_open <- segment.dwells_by_conductance(my_burst, 0.5)
head(half_open)
```

```
segment.dwells_by_conductance_range
```

Extract dwells in conductance range. lower <= x <= upper

Description

Extract dwells in conductance range. lower <= x <= upper

Usage

```
segment.dwells_by_conductance_range(segment, lower = 0, upper = Inf)
```

Arguments

segment	the segment object
lower	lower bound on conductance (defaults to 0)
upper	upper bound on conductance (defaults to infinity)

Value

the dwells in a given range

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 0.2, 0, 1, 0, 0.5, 0, 0.7, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

half_open <- segment.dwells_by_conductance_range(my_burst, lower=0.2, upper=0.7)
head(half_open)
```

`segment.impose_deadtime`

Imposes a deadtime to a segment by removing any dwell that is shorter than the deadtime.

Description

The user specifies a deadtime in microseconds. The function effectively undoes the work of the event detection algorithm by reverting the conductance level (of the brief dwell) back to the previous conductance level in the time sequence. The function then returns a collapsed segment containing alternating dwells.

Usage

```
segment.impose_deadtime(segment, deadtime)
```

Arguments

`segment` the segment containing dwells and states.
`deadtime` the briefest possible event in microseconds.

Value

A modified copy of the original segment

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 0.2, 0, 1, 0, 0.5, 0, 0.7, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

my_burst_d <- segment.impose_deadtime(my_burst, deadtime=0.3)
```

`segment.modify_conductance`

Transform the conductance states according to a user-defined function of conductance level.

Description

Transform the conductance states according to a user-defined function of conductance level.

Usage

```
segment.modify_conductance(segment, fun)
```

Arguments

segment the segment containing dwells and states.
 fun a function on conductance levels (states)

Value

A modified copy of the original segment

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 0.2, 0, 1, 0, 0.5, 0, 0.7, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

### Collapse into three subconductance states
fun <- function(amp) {
  if (amp < 0.3)
    return(0)
  else if (amp >= 0.3 && amp < 0.6)
    return(0.5)
  else
    return(1)
}

my_burst_d <- segment.modify_conductance(my_burst, fun)
```

segment.name	<i>Extract name from segment.</i>
--------------	-----------------------------------

Description

Extract name from segment.

Usage

```
segment.name(segment)
```

Arguments

segment the segment object

Value

Segment name (string)

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

segment.name(my_burst)
```

segment.open_dwells *Extract open dwells. (Any conductance greater than zero)*

Description

Extract open dwells. (Any conductance greater than zero)

Usage

```
segment.open_dwells(segment)
```

Arguments

segment the segment object

Value

the open dwells

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

open_dwells <- segment.open_dwells(my_burst)
head(open_dwells)
```

segment.pclosed	<i>Calculate empirical P(Closed) of a segment.</i>
-----------------	--

Description

Calculate empirical P(Closed) of a segment. NOTE: Assuming that burst starts and ends with 1

Usage

```
segment.pclosed(segment)
```

Arguments

segment	The dwells and states table
---------	-----------------------------

Value

The ratio of closed time to total time

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

# P(Closed) of this burst
segment.pclosed(my_burst)
```

segment.pconductance	<i>Calculate empirical P(Lower <= Conductance <= Upper) of a segment.</i>
----------------------	---

Description

Calculate empirical P(Lower <= Conductance <= Upper) of a segment.

Usage

```
segment.pconductance(segment, level)
```

Arguments

segment	the segment object
level	conductance level

Value

The probability of being in this conductance state

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 0.2, 0, 1, 0, 0.5, 0, 0.7, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

segment.pconductance(my_burst, 0.5)
```

segment.pconductance_range

Calculate empirical $P(\text{Lower} \leq \text{Conductance} \leq \text{Upper})$ of a segment.

Description

Calculate empirical $P(\text{Lower} \leq \text{Conductance} \leq \text{Upper})$ of a segment.

Usage

```
segment.pconductance_range(segment, lower = 0, upper = Inf)
```

Arguments

segment	the segment object
lower	lower bound on conductance (defaults to 0)
upper	upper bound on conductance (defaults to infinity)

Value

The probability of being in these conductance states

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 0.2, 0, 1, 0, 0.5, 0, 0.7, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

segment.pconductance_range(my_burst, lower=0.5, upper=0.5)
```

segment.popen	<i>Calculate empirical P(Open) of a segment. (A state is considered open if the conductance is non-zero)</i>
---------------	--

Description

Calculate empirical P(Open) of a segment. NOTE: Assuming that burst starts and ends with 1

Usage

```
segment.popen(segment)
```

Arguments

segment	The dwells and states table
---------	-----------------------------

Value

The ratio of open time to total time

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

# P(Open) of this burst
segment.popen(my_burst)
```

segment.seg	<i>Extract segment number from segment.</i>
-------------	---

Description

Extract segment number from segment.

Usage

```
segment.seg(segment)
```

Arguments

segment	the segment object
---------	--------------------

Value

Segment number (integer)

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=0, name="example_segment")

segment.seg(my_burst)
```

segment.start_time *Extract start_time from segment.*

Description

Extract start_time from segment.

Usage

```
segment.start_time(segment)
```

Arguments

segment the segment object

Value

Segment start_time (float)

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

segment.start_time(my_burst)
```

 segment.subconductance_as

Imposes a fixed conductance level (0 or 1) to all dwells with subconductance levels.

Description

The user specifies the desired level ('open' or 'closed'). The function will modify any subconductance level (that is not 0 or 1) to be the desired level 1 for 'open' or 0 for 'closed'. The function then returns a collapsed segment containing alternating dwells. (See `segment.consecutives_to_dwells` for details about the collapsed segment.)

Usage

```
segment.subconductance_as(segment, level)
```

Arguments

segment	the segment containing dwells and states.
level	either 'open' or 'closed'

Value

A modified copy of the original segment

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 0.2, 0, 1, 0, 0.5, 0, 0.7, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

my_burst_d <- segment.subconductance_as(my_burst, "open")
```

 segment.verify

Detect misrecorded data.

Description

Segments should have a very specific shape, but recordings can produce errors that make nonsensical segments. In particular, ones contain multiple consecutive states of equal conductance, or end in closings. This function detects whether a segment satisfies the constraint that the segment conductances are not the same from one dwell to the next, and begin and end with a closing.

Usage

```
segment.verify(segment)
```

Arguments

```
segment          The dwells and states table
```

Value

True if a valid segment, False otherwise

Examples

```
# It's more likely that you created states or dwells with some function
states <- c(0, 1, 0, 1, 0, 1, 0, 1, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.0, 1.1, 0.6, 1.1, 0.8, 1.1)
my_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="example_segment")

segment.verify(my_burst)

# Now, a bad burst with two adjacent open dwells
states <- c(0, 1, 0, 1, 1, 0, 1, 0, 1)
dwells <- c(0.1, 1.1, 0.5, 0.2, 1.1, 0.6, 1.1, 0.8, 1.1)

# This will issue a warning
faulty_burst <- segment.create(states, dwells, seg=1, start_time=3.14159, name="faulty_segment")

# This will differentiate good and faulty bursts
segment.verify(faulty_burst)

# If you have a list of bursts, you can select the good ones with
# vbursts <- bursts.select(bursts, segment.verify)
```

```
util.basename
```

```
Remove suffix and path from filename.
```

Description

Remove suffix and path from filename.

Usage

```
util.basename(filename)
```

Arguments

```
filename          string to extract basename from
```

Value

Name with suffix and path removed

Examples

```
util.basename("bursts/60uM-2017-08-18-16-32/60uM-712.dwt")
```

Index

bursts.check_subconductance, 3
bursts.conductance_states, 3
bursts.copy, 4
bursts.defined_by_tcrit, 4
bursts.get_gaps, 5
bursts.impose_deadtime, 6
bursts.modify_conductance, 7
bursts.pclosed, 8
bursts.popens, 8
bursts.recombine, 9
bursts.remove_first_and_last, 10
bursts.select, 10
bursts.sort, 11
bursts.space_out, 12
bursts.start_times_update, 13
bursts.subconductance_as, 13

clampfit.read, 14
cplot.conductance_hist, 15
cplot.log_root_axes, 15
cplot.pclosed_ts, 16
cplot.popen_ts, 17

dwt.read, 17
dwt.write, 18

evt.extract_header, 19
evt.from_dwells, 19
evt.read, 20
evt.to_dwells, 21
evt.write, 21

hst.extract_header, 22
hst.read, 23
hst.write, 23

risetime.correct_gaussian, 24

scan.read, 25
segment.check_subconductance, 26
segment.closed_dwells, 26
segment.conductance_states, 27
segment.consecutives_to_dwells, 28
segment.copy, 28
segment.count_closed, 29
segment.count_dwells, 29
segment.count_open, 30
segment.create, 31
segment.duration, 32
segment.dwells_by_conductance, 32
segment.dwells_by_conductance_range, 33
segment.impose_deadtime, 34
segment.modify_conductance, 34
segment.name, 35
segment.open_dwells, 36
segment.pclosed, 37
segment.pconductance, 37
segment.pconductance_range, 38
segment.popen, 39
segment.seg, 39
segment.start_time, 40
segment.subconductance_as, 41
segment.verify, 41

util.basename, 42