Package 'scalpel'

October 14, 2022

Type Package

Title Processes Calcium Imaging Data

Version 1.0.3

Author Ashley Petersen

Maintainer Ashley Petersen <ashleyjpete@gmail.com>

Description Identifies the locations of neurons, and estimates their calcium concentrations over time using the SCALPEL method proposed in Petersen, Ashley; Simon, Noah; Witten, Daniela. SCALPEL: Extracting neurons from calcium imaging data. Ann. Appl. Stat. 12 (2018), no. 4, 2430--2456. <doi:10.1214/18-AOAS1159>. <https://projecteuclid.org/euclid.aoas/1542078051>.

License GPL (>= 2)

URL www.ajpete.com/software

Imports Matrix, R.matlab, protoclust, igraph, gam

LazyData TRUE

Encoding UTF-8

RoxygenNote 7.1.1

Suggests knitr, rmarkdown

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-02-03 05:30:02 UTC

R topics documented:

`scalpel-package` *scalpel: A package for processing calcium imaging data.*

Description

This package is called scalpel for "Segmentation, Clustering, and Lasso Penalties", which is a method for processing neuronal calcium imaging data that identifies the locations of neurons, and estimates their calcium concentrations over time. The main function is `scalpel`, which runs the entire SCALPEL pipeline. The pipeline involves several steps, each of which is described briefly in its corresponding function. See `scalpelStep0`, `scalpelStep1`, `scalpelStep2`, `scalpelStep3` for more details. Results can be summarized using `summary` and the main plotting function is `plotResults`, which displays the estimated spatial and temporal components. Full details for the SCALPEL method are provided in Petersen, Ashley; Simon, Noah; Witten, Daniela. SCALPEL: Extracting neurons from calcium imaging data. Ann. Appl. Stat. 12 (2018), no. 4, 2430–2456. doi:10.1214/18-AOAS1159. https://projecteuclid.org/euclid.aoas/1542078051

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software
```

```
#general example illustrating some of the main functions
#see the vignette for additional direction on using all of the functions
#and the help pages for the specific functions for details on using each function

#existing folder to save results (update this to an existing folder on your computer)
outputFolder = "scalpelResults"
#location on computer of raw data in R package to use
rawDataFolder = gsub("Y_1.rds", "", system.file("extdata", "Y_1.rds", package = "scalpel"))
#video height of raw data in R package
videoHeight = 30
#run SCALPEL pipeline
scalpelOutput = scalpel(outputFolder = outputFolder, rawDataFolder = rawDataFolder,
                        videoHeight = videoHeight)
#summarize each step
summary(scalpelOutput, step = 0)
summary(scalpelOutput, step = 1)
summary(scalpelOutput, step = 2)
summary(scalpelOutput, step = 3)

#plot the spatial and temporal components
plotResults(scalpelOutput = scalpelOutput)
#plot a summary of the video with the found neurons outlined
plotVideoVariance(scalpelOutput = scalpelOutput, neuronSet = "Afilter")
#plot the frames with the most fluorescence for each found neuron
plotBrightest(scalpelOutput = scalpelOutput, AfilterIndex = 1)
plotBrightest(scalpelOutput = scalpelOutput, AfilterIndex = 2)
plotBrightest(scalpelOutput = scalpelOutput, AfilterIndex = 3)

#if you want to use results from a previous session,
#use "getScalpel" to read in previous results
scalpelOutputCopy = getScalpel(outputFolder = outputFolder)


## End(Not run)
```

`getNeuronStatus` *Read in the manual classifications of neurons from SCALPEL.*

Description

This function allows the user to read in the manual classifications of neurons, based on the classifying done using `reviewNeurons` or `reviewNeuronsInteractive`.

Usage

```
getNeuronStatus(scalpelOutput, neuronSet)
```

Arguments

scalpelOutput	An object returned by one of the SCALPEL functions: scalpel, scalpelStep2, or scalpelStep3.
neuronSet	The set of neurons that should be reviewed: use "A" for those resulting from scalpelStep2 and saved as scalpelOutput$A, or use "Afilter" for those resulting from scalpelStep3 and saved as scalpelOutput$Afilter. This argument is ignored if the class of scalpelOutput is scalpelStep2.

Value

A vector of length equal to the number of columns in scalpelOutput$A if neuronSet="A" or scalpelOutput$Afilter if neuronSet="Afilter". The elements give the manual classifications of the neurons. The possible classifications are: "yes" if a neuron is to be kept, "no" if a neuron is to be discarded, "unsure" if a neuron needs to be reviewed further, and NA if a neuron has not yet been classified.

See Also

reviewNeurons, reviewNeuronsInteractive

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "updateNeurons" function

getNeuronStatus(scalpelOutput = scalpelOutput, neuronSet = "Afilter")

## End(Not run)
```

---

getScalpel *Read in results from SCALPEL.*

Description

This step allows the user to retrieve the object of class scalpel for results from a previous session.

Usage

```
getScalpel(
  outputFolder,
  version = NULL,
  cutoff = 0.18,
  omega = 0.2,
  lambdaMethod = "trainval",
```

```
  lambda = NULL,
  minClusterSize = 1,
  alpha = 0.9,
  removeBorder = FALSE,
  excludeReps = NULL
)
```

Arguments

outputFolder	The existing directory where the results that the user wishes to use are saved.
version	The 5-digit folder ID for the results that the user wishes to load. If NULL, automatically chooses the only version in outputFolder and if more than one version exists, returns an error.
cutoff	A value in [0,1] indicating the dendrogram cutpoint used. The default value is 0.18.
omega	A value in [0,1] indicating the dissimilarity metric weight used for clustering. The default value is 0.2.
lambdaMethod	How lambda was chosen: either "trainval" (default), "distn", or "user".
lambda	The value of lambda used to fit the sparse group lasso. If NULL, automatically chooses the only lambda in directory and if more than one lambda exists, returns an error.
minClusterSize	The minimum number of preliminary dictionary elements that a cluster must have contained to have been included in the sparse group lasso. The default value is 1.
alpha	The value of alpha used to fit the sparse group lasso. The default value is 0.9.
removeBorder	A logical scalar indicating whether the dictionary elements that contained pixels in the 10-pixel border of the video were removed prior to fitting the sparse group lasso. The default value is FALSE.
excludeReps	A vector giving the indices of which dictionary elements were excluded. The default value is NULL meaning no dictionary elements were manually excluded.

Value

An object of class scalpel, which can be used to rerun SCALPEL Steps 1-3 with new parameters using scalpelStep1, scalpelStep2, and scalpelStep3 or can be used with any of the plotting functions: plotFrame, plotThresholdedFrame, plotVideoVariance, plotCandidateFrame, plotCluster, plotResults, plotResultsAllLambda, plotSpatial, plotTemporal, and plotBrightest.

See Also

scalpel

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
```

```
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function
#folder where results were saved
outputFolder = "scalpelResults"

#read previous results in
#simplest example with default parameters:
out = getScalpel(outputFolder = outputFolder)
#note: if Step 1 has been run more than once, will need to specify 'version'

#example with optional parameters:
#need to enter if non-default options were used
out = getScalpel(outputFolder = outputFolder, omega = 0.2, cutoff = 0.18,
                 alpha = 0.9, minClusterSize = 1)

## End(Not run)
```

`getScalpelStep0` *Read in results from Step 0 of SCALPEL.*

Description

This step allows the user to retrieve the object of class `scalpelStep0` for results from a previous session.

Usage

```
getScalpelStep0(outputFolder)
```

Arguments

`outputFolder`	The existing directory where the results that the user wishes to use are saved.

Value

An object of class `scalpelStep0`, which can be used to run SCALPEL Step 1 using `scalpelStep1` or can be used with the plotting functions `plotFrame`, `plotThresholdedFrame`, and `plotVideoVariance`.

See Also

`scalpelStep0`, `scalpel`

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software
```

```
#assumes you have run the example for the "scalpel" function
#folder where results were saved
outputFolder = "scalpelResults"

#read previous results in
out = getScalpelStep0(outputFolder = outputFolder)

## End(Not run)
```

`getScalpelStep1` *Read in results from Step 1 of SCALPEL.*

Description

This step allows the user to retrieve the object of class `scalpelStep1` for results from a previous session.

Usage

```
getScalpelStep1(outputFolder, version = NULL)
```

Arguments

`outputFolder`	The existing directory where the results that the user wishes to use are saved.
`version`	The 5-digit folder ID for the results that the user wishes to load. If NULL, automatically chooses the only version in outputFolder and if more than one version exists, returns an error.

Value

An object of class `scalpelStep1`, which can be used to run SCALPEL Step 2 using `scalpelStep2` or can be used with the plotting function `plotCandidateFrame`.

See Also

`scalpelStep1`, `scalpel`

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function
#folder where results were saved
outputFolder = "scalpelResults"

#read previous results in
```

```
out = getScalpelStep1(outputFolder = outputFolder)
#note: if Step 1 has been run more than once, will need to specify 'version'

## End(Not run)
```

`getScalpelStep2` *Read in results from Step 2 of SCALPEL.*

Description

This step allows the user to retrieve the object of class `scalpelStep2` for results from a previous session.

Usage

```
getScalpelStep2(outputFolder, version = NULL, cutoff = 0.18, omega = 0.2)
```

Arguments

`outputFolder`	The existing directory where the results that the user wishes to use are saved.
`version`	The 5-digit folder ID for the results that the user wishes to load. If NULL, automatically chooses the only version in outputFolder and if more than one version exists, returns an error.
`cutoff`	A value in [0,1] indicating the dendrogram cutpoint used. The default value is 0.18.
`omega`	A value in [0,1] indicating the dissimilarity metric weight used for clustering. The default value is 0.2.

Value

An object of class `scalpelStep2`, which can be used to run SCALPEL Step 3 using `scalpelStep3` or can be used with the plotting functions `plotCluster` and `plotSpatial`.

See Also

`scalpelStep2`, `scalpel`

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function
#folder where results were saved
outputFolder = "scalpelResults"
```

```
#read previous results in
#simplest example with default parameters:
out = getScalpelStep2(outputFolder = outputFolder)
#note: if Step 1 has been run more than once, will need to specify 'version'

#example with optional parameters:
#need to enter if non-default options were used
out = getScalpelStep2(outputFolder = outputFolder, omega = 0.2, cutoff = 0.18)

## End(Not run)
```

`getScalpelStep3` *Read in results from Step 3 of SCALPEL.*

Description

This step allows the user to retrieve the object of class `scalpelStep3` for results from a previous session.

Usage

```
getScalpelStep3(
  outputFolder,
  version = NULL,
  cutoff = 0.18,
  omega = 0.2,
  lambdaMethod = "trainval",
  minClusterSize = 1,
  alpha = 0.9,
  lambda = NULL,
  removeBorder = FALSE,
  excludeReps = NULL
)
```

Arguments

`outputFolder`	The existing directory where the results that the user wishes to use are saved.
`version`	The 5-digit folder ID for the results that the user wishes to load. If NULL, automatically chooses the only version in outputFolder and if more than one version exists, returns an error.
`cutoff`	A value in [0,1] indicating the dendrogram cutpoint used. The default value is 0.18.
`omega`	A value in [0,1] indicating the dissimilarity metric weight used for clustering. The default value is 0.2.
`lambdaMethod`	How lambda was chosen: either `"trainval"` (default), `"distn"`, or `"user"`.

minClusterSize	The minimum number of preliminary dictionary elements that a cluster must have contained to have been included in the sparse group lasso. The default value is 1.
alpha	The value of alpha used to fit the sparse group lasso. The default value is 0.9.
lambda	The value of lambda used to fit the sparse group lasso. If NULL, automatically chooses the only lambda in directory and if more than one lambda exists, returns an error.
removeBorder	A logical scalar indicating whether the dictionary elements that contained pixels in the 10-pixel border of the video were removed prior to fitting the sparse group lasso. The default value is FALSE.
excludeReps	A vector giving the indices of which dictionary elements were excluded. The default value is NULL meaning no dictionary elements were manually excluded.

Value

An object of class scalpelStep3, which can be used with the plotting functions plotResults, plotResultsAllLambda, plotSpatial, plotTemporal, and plotBrightest.

See Also

scalpelStep3, scalpel

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function
#folder where results were saved
outputFolder = "scalpelResults"

#read previous results in
#simplest example with default parameters:
out = getScalpelStep3(outputFolder = outputFolder)
#note: if Step 1 has been run more than once, will need to specify 'version'

#example with optional parameters:
#need to enter if non-default options were used
out = getScalpelStep3(outputFolder = outputFolder, omega = 0.2, cutoff = 0.18,
                      alpha = 0.9, minClusterSize = 1)

## End(Not run)
```

getY *Read in Y matrix for SCALPEL.*

Description

This step allows the user to read in Y, the matrix of raw or processed video data, to use with several plotting functions.

Usage

```
getY(scalpelOutput, videoType = "processed", part = NULL)
```

Arguments

scalpelOutput	An object returned by one of the SCALPEL functions: scalpel, scalpelStep0, scalpelStep1, scalpelStep2, or scalpelStep3.
videoType	Specify whether to read in the processed data from Step 0 (default; videoType="processed") or raw data (videoType="raw").
part	The part of the video to read in, if it is split across multiple files. The default is NULL, which means that all parts will be read in and combined.

Value

An object of class scalpelY that can be provided as the Y argument in plotFrame, plotVideoVariance, plotBrightest, plotThresholdedFrame, and plotCandidateFrame. If would like to call these functions many times, this avoids reading the video into memory repeatedly.

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function
#read in the raw data
rawY = getY(scalpelOutput = scalpelOutput, videoType = "raw")
#read in the processed data from Step 0
processedY = getY(scalpelOutput = scalpelOutput, videoType = "processed")

## End(Not run)
```

plotBrightest *Plot the most active frames for a given neuron.*

Description

For a given neuron, we plot the frames with the highest estimated fluorescence, which results from fitting the sparse group lasso in Step 3 of SCALPEL.

Usage

```
plotBrightest(
  scalpelOutput,
  AfilterIndex,
  videoType = "processed",
  neuronsToOutline = "all",
  brightIndex = 1,
  shrinkLargest = FALSE,
  shrinkCutoff = NULL,
  title = NULL,
  Y = NULL
)
```

Arguments

scalpelOutput	An object returned by one of the SCALPEL functions: scalpel or scalpelStep3.
AfilterIndex	Scalar giving the neuron for which to plot the brightest frames. The index refers to which column of scalpelOutput$Afilter is of interest.
videoType	Specify whether to plot the processed data from Step 0 (default; videoType="processed") or raw data (videoType="raw"). This is ignored if Y is provided.
neuronsToOutline	Specify whether to plot outlines of all neurons (default; neuronsToOutline="all"), only the outline for neuron (neuronsToOutline="main"), outlines of only the neurons kept using a previous call to reviewNeurons or reviewNeuronsInteractive (neuronsToOutline="kept"), or none (neuronsToOutline="none").
brightIndex	Scalar giving which of the ordered brightest frames to plot. The default is 1, i.e., the brightest frame.
shrinkLargest	Logical value indicating whether the values above shrinkCutoff should be shrunk when plotting. Shrinking these values allows us to better visualize the areas with the largest fluorescence.
shrinkCutoff	The value above which pixel values will be shrunk. By default, this will be chosen as min(scalpelOutput$thresholdVec).
title	Label for the title. The default is frame number.
Y	An object of class scalpelY, which results from running the getY function. When not specified, Y is automatically read in, but specifying Y is recommended when the user would like to call this function many times, as this avoids reading the video into memory repeatedly.

Value

None

See Also

scalpel, scalpelStep3

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function
#simplest example with default parameters:
plotBrightest(scalpelOutput = scalpelOutput, AfilterIndex = 2)

#example with optional parameters:
#only outline neuron corresponding to frame, plot 5th brightest with raw data
plotBrightest(scalpelOutput = scalpelOutput, AfilterIndex = 2, videoType = "raw",
             neuronsToOutline = "main", brightIndex = 5)

#same plot but if you have video data read in already
#using 'getY' function, you can provide it
rawY = getY(scalpelOutput = scalpelOutput, videoType = "raw")
plotBrightest(scalpelOutput = scalpelOutput, AfilterIndex = 2, Y = rawY,
             neuronsToOutline = "main", brightIndex = 5)

## End(Not run)
```

---

plotCandidateFrame *Plot preliminary dictionary element from Step 1 of SCALPEL and its corresponding frame.*

Description

We plot the specified preliminary dictionary element, along with the frame of Y from which the component was derived in Step 1 of SCALPEL.

Usage

```
plotCandidateFrame(
  scalpelOutput,
  AzeroIndex = NULL,
  AIndex = NULL,
  AfilterIndex = NULL,
  member = NULL,
  videoType = "processed",
```

```
  shrinkLargest = FALSE,
  shrinkCutoff = NULL,
  Y = NULL
)
```

Arguments

scalpelOutput	An object returned by one of the SCALPEL functions: scalpel, scalpelStep1, scalpelStep2, or scalpelStep3.
AzeroIndex	The preliminary dictionary element of interest. The index refers to the column of scalpelOutput$Azero. Specify only one of the following: AzeroIndex, AIndex, or AfilterIndex.
AIndex	The dictionary element (i.e., cluster) of interest. The index refers to the column of scalpelOutput$A. Note that the class of scalpelOutput must be scalpel, scalpelStep2, or scalpelStep3 if specifying AIndex, and member must also be specified. Specify only one of the following: AzeroIndex, AIndex, or AfilterIndex.
AfilterIndex	The sparse group lasso component of interest. The index refers to the column of scalpelOutput$Afilter. Note that the class of scalpelOutput must be scalpel or scalpelStep3 if specifying AfilterIndex, and member must also be specified. Specify only one of the following: AzeroIndex, AIndex, or AfilterIndex.
member	Which member of the cluster corresponding to AIndex or AfilterIndex to plot. Ignored if AzeroIndex is specified.
videoType	Specify whether to plot the processed data from Step 0 (default; videoType="processed") or raw data (videoType="raw"). This is ignored if Y is provided.
shrinkLargest	Logical value indicating whether the values above shrinkCutoff should be shrunk when plotting. Shrinking these values allows us to better visualize the areas with the largest fluorescence.
shrinkCutoff	The value above which pixel values will be shrunk. By default, this will be chosen as min(scalpelOutput$thresholdVec).
Y	An object of class scalpelY, which results from running the getY function. When not specified, Y is automatically read in, but specifying Y is recommended when the user would like to call this function many times, as this avoids reading the video into memory repeatedly.

Value

None

See Also

scalpelStep1, scalpel

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
```

```
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function
#simplest example with default parameters:
plotCandidateFrame(scalpelOutput = scalpelOutput, AzeroIndex = 10)

#example with optional parameters:
#plot raw data instead of processed
plotCandidateFrame(scalpelOutput = scalpelOutput, AzeroIndex = 10, videoType = "raw")

#same plot but if you have video data read in already
#using 'getY' function, you can provide it
rawY = getY(scalpelOutput = scalpelOutput, videoType = "raw")
plotCandidateFrame(scalpelOutput = scalpelOutput, AzeroIndex = 10, Y = rawY)

## End(Not run)
```

`plotCluster` *Plot a summary of a given cluster from Step 2 of SCALPEL.*

Description

We plot the preliminary dictionary elements that correspond to a given dictionary element, derived during Step 2 of SCALPEL, or a given component included in the sparse group lasso of Step 3.

Usage

```
plotCluster(
  scalpelOutput,
  AIndex = NULL,
  AfilterIndex = NULL,
  pctTransp = 0.01
)
```

Arguments

`scalpelOutput`	An object returned by one of the SCALPEL functions: `scalpel`, `scalpelStep2`, or `scalpelStep3`.
`AIndex`	The dictionary element (i.e., cluster) of interest. The index refers to the column of `scalpelOutput$A`, which is part of the output from `scalpelStep2`. Specify `AIndex` or `AfilterIndex`, not both.
`AfilterIndex`	The refined dictionary element of interest. The index refers to the column of `scalpelOutput$Afilter`, which is part of the output from `scalpelStep3`. Note that the class of `scalpelOutput` must be `scalpel` or `scalpelStep3` if specifying `AfilterIndex`. Specify `AIndex` or `AfilterIndex`, not both.
`pctTransp`	The percent transparency (in [0,1]) for the colors used to plot the preliminary dictionary elements. The default value is 0.01.

Details

The left plot shows the dictionary element of interest in orange, with the other dictionary elements shown in blue. The middle plot shows all of the preliminary dictionary elements corresponding to the dictionary element plotted transparently. The right plot shows the dictionary element in orange, along with the union of all of the preliminary dictionary elements in gray. Note that the plots in the middle and on the right are zoomed-in, compared to the plot on the left that shows the entire field of view for the video.

Value

None

See Also

scalpelStep2, scalpel

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function
#plots the cluster for the 2nd dictionary element (i.e., scalpelOutput$A[,2])
plotCluster(scalpelOutput = scalpelOutput, AIndex = 2)
#plots the cluster for the 2nd component included in SGL (i.e., scalpelOutput$Afilter[,2])
plotCluster(scalpelOutput = scalpelOutput, AfilterIndex = 2)

## End(Not run)
```

---

plotFrame *Plot a frame of the video.*

Description

We plot a specified frame of the raw video that we began with in Step 0 of SCALPEL, or the processed video that results from Step 0 of SCALPEL.

Usage

```
plotFrame(
  scalpelOutput,
  frame,
  videoType = "processed",
  shrinkLargest = FALSE,
  shrinkCutoff = NULL,
  title = NULL,
```

```
  col = grDevices::grey(seq(0, 1, length = 256)),
  addToPlot = FALSE,
  Y = NULL
)
```

Arguments

scalpelOutput	An object returned by one of the SCALPEL functions: scalpel, scalpelStep0, scalpelStep1, scalpelStep2, or scalpelStep3.
frame	The frame to plot.
videoType	Specify whether to plot the processed data from Step 0 (default; videoType="processed") or raw data (videoType="raw"). This is ignored if Y is provided.
shrinkLargest	Logical value indicating whether the values above shrinkCutoff should be shrunk when plotting. Shrinking these values allows us to better visualize the areas with the largest fluorescence.
shrinkCutoff	The value above which pixel values will be shrunk. By default, this will be chosen as scalpelOutput$lowThreshold if class(scalpelOutput)=="scalpelStep0" or min(scalpelOutput$thresholdVec) otherwise.
title	Label for the title. By default, it is the frame number.
col	Vector of colors to use, which by default is grayscale.
addToPlot	Logical value indicating whether to add to the current plot.
Y	An object of class scalpelY, which results from running the getY function. When not specified, Y is automatically read in, but specifying Y is recommended when the user would like to call this function many times, as this avoids reading the video into memory repeatedly.

Value

None

See Also

scalpelStep0, scalpel

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function

#simplest example with default parameters:
plotFrame(scalpelOutput = scalpelOutput, frame = 100)

#example with optional parameters:
#plot raw data instead of processed
```

```
plotFrame(scalpelOutput = scalpelOutput, frame = 100, videoType = "raw")

#same plot but if you have video data read in already
#using 'getY' function, you can provide it
rawY = getY(scalpelOutput = scalpelOutput, videoType = "raw")
plotFrame(scalpelOutput = scalpelOutput, frame = 100, Y = rawY)

## End(Not run)
```

`plotResults` *Plot both the spatial and temporal components from Step 3 of SCALPEL.*

Description

We plot the temporal components, displaying the estimated fluorescence over time for each spatial component, along with a map of the spatial components.

Usage

```
plotResults(
  scalpelOutput,
  neuronsToDisplay = NULL,
  colVec = NULL,
  titleA = "",
  titleZ = "",
  ylabZ = "",
  fileName = NULL,
  pctTransp = 0.7,
  number = TRUE,
  border = FALSE
)
```

Arguments

`scalpelOutput`	An object returned by one of the SCALPEL functions: `scalpel` or `scalpelStep3`.
`neuronsToDisplay`	Vector giving which neurons' spatial and temporal components to plot. The indices refer to which columns of `scalpelOutput$Afilter` to plot. By default, all components are plotted. Users may also specify `"kept"`, which will exclude all dictionary elements discarded using a previous call to `reviewNeurons` or `reviewNeuronsInteractive`.
`colVec`	Vector of colors to use, which are chosen automatically if the default value of NULL is used.
`titleA`	Label for the title of the spatial components plot.
`titleZ`	Label for the title of the temporal components plot.
`ylabZ`	Label for the y-axis of the temporal components plot.

fileName	If provided, the plot will be saved to the specified location.
pctTransp	The percent transparency (in [0,1]) for the colors used to plot the neurons. The default value is 0.7.
number	Logical value indicating whether the neurons should be numbered.
border	Logical value indicating whether only the borders of the neurons should be plotted.

Details

If lambdaIndex is NULL, each temporal component is scaled by its largest value. If lambdaIndex is specified, each temporal component is scaled by its largest value across all of the lambda values. Temporal components that were zeroed out in the sparse group lasso are omitted from the plot, and their corresponding spatial components are shown in gray.

Value

None

See Also

scalpelStep3, scalpel, plotSpatial, plotTemporal

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function

#simplest example with default parameters:
plotResults(scalpelOutput = scalpelOutput)

#example with optional parameters:
#plot only two of the neurons, do not number neurons, draw the outlines of the neurons
plotResults(scalpelOutput = scalpelOutput, neuronsToDisplay = c(1,2),
           number = FALSE, border = TRUE)

## End(Not run)
```

---

plotResultsAllLambda *Plot both the spatial and temporal components for the sequence of lambda values from Step 3 of SCALPEL.*

Description

We plot the temporal components, displaying the estimated fluorescence over time for each spatial component, along with a map of the spatial components for a whole sequence of lambda values.

Usage

```
plotResultsAllLambda(
  scalpelOutput,
  neuronsToDisplay = NULL,
  colVec = NULL,
  titleA = "",
  ylabZ = "",
  fileName = NULL,
  pctTransp = 0.7,
  number = TRUE,
  border = FALSE
)
```

Arguments

scalpelOutput	An object returned by one of the SCALPEL functions: scalpel or scalpelStep3.
neuronsToDisplay	Vector giving which neurons' spatial and temporal components to plot. The indices refer to which columns of scalpelOutput$Afilter to plot. By default, all components are plotted. Users may also specify "kept", which will exclude all dictionary elements discarded using a previous call to reviewNeurons or reviewNeuronsInteractive.
colVec	Vector of colors to use, which are chosen automatically if the default value of NULL is used.
titleA	Label for the title of the spatial components plot.
ylabZ	Label for the y-axis of the temporal components plot.
fileName	If provided, the plot will be saved to the specified location.
pctTransp	The percent transparency (in [0,1]) for the colors used to plot the neurons. The default value is 0.7.
number	Logical value indicating whether the neurons should be numbered.
border	Logical value indicating whether only the borders of the neurons should be plotted.

Details

Temporal components that were zeroed out in the sparse group lasso and their corresponding spatial components are shown in gray for both plots.

Value

None

See Also

scalpelStep3, scalpel, plotSpatial, plotTemporal

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function

#simplest example with default parameters:
plotResultsAllLambda(scalpelOutput = scalpelOutput)

#example with optional parameters:
#plot only two of the neurons, do not number neurons, draw the outlines of the neurons
plotResultsAllLambda(scalpelOutput = scalpelOutput, neuronsToDisplay = c(1,2),
                     number = FALSE, border = TRUE)

## End(Not run)
```

`plotSpatial` *Plot spatial components from Steps 2 or 3 of SCALPEL.*

Description

We plot the dictionary elements obtained from Step 2 of SCALPEL, or the filtered dictionary elements from Step 3 of SCALPEL.

Usage

```
plotSpatial(
  scalpelOutput = NULL,
  neuronSet = "",
  neuronsToDisplay = NULL,
  colVec = NULL,
  title = "",
  fileName = NULL,
  pctTransp = 0.7,
  number = TRUE,
  addToPlot = FALSE,
  border = FALSE,
  zoom = FALSE,
  A = NULL,
  videoHeight = NULL
)
```

Arguments

`scalpelOutput`	An object returned by one of these SCALPEL functions: `scalpel`, `scalpelStep2`, or `scalpelStep3`.

neuronSet	Which set of neurons should be plotted: use "A" for the dictionary elements resulting from scalpelStep2 and saved as scalpelOutput$A, or use "Afilter" for the filtered dictionary elements resulting from scalpelStep3 and saved as scalpelOutput$Afilter.
neuronsToDisplay	Vector giving which neurons' spatial components to plot. The indices refer to which columns to plot of scalpelOutput$Afilter (if neuronSet="Afilter"), or scalpelOutput$A (if neuronSet="A"). By default, all components are plotted. Users may also specify "kept", which will exclude all dictionary elements discarded using a previous call to reviewNeurons or reviewNeuronsInteractive.
colVec	Vector of colors to use, which are chosen automatically if the default value of NULL is used.
title	Label for the title.
fileName	If provided, the plot will be saved to the specified location.
pctTransp	The percent transparency (in [0,1]) for the colors used to plot the neurons. The default value is 0.7.
number	Logical value indicating whether the neurons should be numbered.
addToPlot	Logical value indicating whether these neurons should be plotted to an existing plot.
border	Logical value indicating whether only the borders of the neurons should be plotted.
zoom	Logical value indicating whether the plot should be zoomed in to exclude any area not containing neurons.
A	Optional advanced user argument: A matrix containing the spatial components to plot, where the ith column of A is a vector of 1's and 0's, indicating whether each pixel is contained in the ith spatial component. By default, this argument is ignored and the dictionary elements saved in scalpelOutput$A or scalpelOutput$Afilter are plotted. If A is provided, scalpelOutput will be ignored and neuronsToDisplay will refer to the columns of A.
videoHeight	The height of the video (in pixels). This only needs to be specified if the user is plotting A.

Details

When neuronSet="Afilter", spatial components corresponding to temporal components that were zeroed out in the sparse group lasso are plotted in gray, unless colVec is specified by the user.

Value

None

See Also

scalpelStep2, scalpelStep3, scalpel, plotResults

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function

#simplest example with default parameters:
plotSpatial(scalpelOutput = scalpelOutput, neuronSet = "Afilter")

#example with optional parameters:
#plot only two of the neurons, add a title, do not number neurons,
#and draw the outlines of the neurons
plotSpatial(scalpelOutput = scalpelOutput, neuronsToDisplay = c(1,2), neuronSet = "Afilter",
           title = "First two neurons", number = FALSE, border = TRUE)

## End(Not run)
```

`plotTemporal` *Plot temporal components from Step 3 of SCALPEL.*

Description

We plot the temporal components, displaying the estimated fluorescence over time for each spatial component, which result from running Step 3 of SCALPEL.

Usage

```
plotTemporal(
  scalpelOutput,
  neuronsToDisplay = NULL,
  colVec = NULL,
  ylab = "",
  title = "",
  fileName = NULL,
  lambdaIndex = NULL
)
```

Arguments

`scalpelOutput`	An object returned by one of these SCALPEL functions: `scalpel` or `scalpelStep3`.
`neuronsToDisplay`	Vector giving which neurons' temporal components to plot. The indices refer to which rows of `scalpelOutput$Zhat` to plot. By default, all components are plotted. Users may also specify `"kept"`, which will exclude all dictionary elements discarded using a previous call to `reviewNeurons` or `reviewNeuronsInteractive`.

colVec	Vector of colors to use, which are chosen automatically if the default value of NULL is used.
ylab	Label for the y-axis.
title	Label for the title.
fileName	If provided, the plot will be saved to the specified location.
lambdaIndex	Optional advanced user argument: Index of lambda value for which results will be plotted. Default is to use lambda value of scalpelOutput$lambda but specifying this will use the lambda value of scalpelOutput$lambdaSeq[lambdaIndex].

Details

If lambdaIndex is NULL, each temporal component is scaled by its largest value. If lambdaIndex is specified, each temporal component is scaled by its largest value across all of the lambda values. Temporal components that were zeroed out in the sparse group lasso are omitted from the plot.

Value

None

See Also

scalpelStep3, scalpel, plotResults

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function

#simplest example with default parameters:
plotTemporal(scalpelOutput = scalpelOutput)

#example with optional parameters:
#plot only two of the neurons and add a title
plotTemporal(scalpelOutput = scalpelOutput, neuronsToDisplay = c(1,2),
             title = "First two neurons")

## End(Not run)
```

plotThresholdedFrame *Plot a frame of the video with shading.*

Description

We plot a specified frame of the processed video, which results from Step 0 of SCALPEL, with shading to indicate values above a specified threshold.

Usage

```
plotThresholdedFrame(
  scalpelOutput,
  frame,
  threshold,
  shrinkLargest = FALSE,
  shrinkCutoff = NULL,
  title = NULL,
  col = "yellow",
  Y = NULL
)
```

Arguments

scalpelOutput	An object returned by one of the SCALPEL functions: scalpel, scalpelStep0, scalpelStep1, scalpelStep2, or scalpelStep3.
frame	The frame to plot.
threshold	Value above which pixels will be shaded.
shrinkLargest	Logical value indicating whether the values above shrinkCutoff should be shrunk when plotting. Shrinking these values allows us to better visualize the areas with the largest fluorescence.
shrinkCutoff	The value above which pixel values will be shrunk. By default, this will be chosen as scalpelOutput$lowThreshold if class(scalpelOutput)=="scalpelStep0" or min(scalpelOutput$thresholdVec) otherwise.
title	Label for the title. By default, it gives the threshold value.
col	Color of shading to use, which is yellow by default.
Y	An object of class scalpelY, which results from running the getY function. When not specified, Y is automatically read in, but specifying Y is recommended when the user would like to call this function many times, as this avoids reading the video into memory repeatedly.

Value

None

See Also

scalpelStep0, scalpel

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function

#simplest example with default parameters:
plotThresholdedFrame(scalpelOutput = scalpelOutput, frame = 100,
                     threshold = scalpelOutput$thresholdVec[1])

#example with optional parameters:
#change shading to purple and add a title
plotThresholdedFrame(scalpelOutput = scalpelOutput, frame = 100, col = "purple",
                    threshold = scalpelOutput$thresholdVec[2])

#if you have video data read in already using 'getY' function, you can provide it
processedY = getY(scalpelOutput = scalpelOutput, videoType = "processed")
plotThresholdedFrame(scalpelOutput = scalpelOutput, frame = 100,
                     threshold = scalpelOutput$thresholdVec[1], Y = processedY)

## End(Not run)
```

---

plotVideoVariance *Plot a summary of the fluorescence in the video.*

Description

We plot a heat map of the variance of each pixel across the frames.

Usage

```
plotVideoVariance(
  scalpelOutput,
  neuronSet = "",
  videoType = "processed",
  neuronsToOutline = "all",
  shrinkLargest = FALSE,
  shrinkQuantile = 0.95,
  title = "",
  Y = NULL
)
```

Arguments

scalpelOutput	An object returned by one of the SCALPEL functions: scalpel, scalpelStep0, scalpelStep1, scalpelStep2, or scalpelStep3.
neuronSet	This argument is ignored unless the class of scalpelOutput is scalpel or scalpelStep3, and neuronsToOutline is not "none". It gives which set of neurons should be plotted: use "A" for those resulting from scalpelStep2 and saved as scalpelOutput$A, or use "Afilter" for those resulting from scalpelStep3 and saved as scalpelOutput$Afilter.
videoType	Specify whether to plot the processed data from Step 0 (default; videoType="processed") or raw data (videoType="raw"). This is ignored if Y is provided.
neuronsToOutline	Specify whether to plot outlines of all neurons (default; neuronsToOutline="all"), none of the neurons (neuronsToOutline="none"), or outlines of only the neurons kept using a previous call to reviewNeurons or reviewNeuronsInteractive (neuronsToOutline="kept"). If scalpelOutput is not of the class scalpel, scalpelStep2, or scalpelStep3, this argument is ignored.
shrinkLargest	Logical value indicating whether the values above shrinkQuantile should be shrunk when plotting. Shrinking these values allows us to better visualize the areas with the highest variance fluorescence.
shrinkQuantile	The quantile value above which pixel values will be shrunk. By default, this is the 95th quantile.
title	Label for the title.
Y	An object of class scalpelY, which results from running the getY function. When not specified, Y is automatically read in, but specifying Y is recommended when the user would like to call this function many times, as this avoids reading the video into memory repeatedly.

Value

None

See Also

scalpelStep0, scalpel

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function

#simplest example with default parameters:
plotVideoVariance(scalpelOutput = scalpelOutput, neuronSet = "Afilter")
```

```
#example with optional parameters:
#previous showed summary of processed data,
#can instead show raw data, not outline the neurons found, and add a title
plotVideoVariance(scalpelOutput = scalpelOutput, videoType = "raw",
                  neuronsToOutline = "none", title = "Raw Data")

#if you have video data read in already using 'getY' function, you can provide it
rawY = getY(scalpelOutput = scalpelOutput, videoType = "raw")
plotVideoVariance(scalpelOutput = scalpelOutput, neuronSet = "Afilter", Y = rawY)

## End(Not run)
```

`reviewNeurons` *Manually classify the identified neurons from SCALPEL.*

Description

We save plots that will be used to review the set of identified neurons that result from either Step 2 or 3 of SCALPEL in order to manually classify them according to whether they appear to be real neurons or not. To do this, the plot of the frame from which the dictionary element was derived is saved. The user can then sort these saved plot into folders indicating whether the neuron is real or not, or indicate that additional frames are needed to make the classification, in which case the `reviewNeuronsMoreFrames` function can subsequently be used. After finishing this sorting process, `updateNeurons` should be called. A similar manual classification can be done interactively using `reviewNeuronsInteractive`.

Usage

```
reviewNeurons(
  scalpelOutput,
  neuronSet,
  keepClusterSize = NULL,
  discardZeroedOut = FALSE
)
```

Arguments

`scalpelOutput`	An object returned by one of the SCALPEL functions: `scalpel`, `scalpelStep2`, or `scalpelStep3`.
`neuronSet`	The set of neurons that should be reviewed: use `"A"` for those resulting from `scalpelStep2` and saved as `scalpelOutput$A`, or use `"Afilter"` for those resulting from `scalpelStep3` and saved as `scalpelOutput$Afilter`. This argument is ignored if the class of `scalpelOutput` is `scalpelStep2`.
`keepClusterSize`	Neurons corresponding to clusters with at least `keepClusterSize` members will be automatically classified as real neurons. The default value is `NULL`, which means that none of the neurons will be automatically kept based on cluster size.

`discardZeroedOut`
: Logical value indicating whether neurons zeroed out in the sparse group lasso problem should automatically be discarded. This argument is ignored when `neuronSet` is `"A"`, and has a default value of `FALSE`.

Details

Plots are saved for each of the neurons under consideration in a certain folder. Also within that folder, there will be folders called 'keep', 'discard', and 'unsure'. After running this function, the plots for each of the neurons should be moved into the appropriate folder. After completing this sorting, call `updateNeurons` in order to update the classification of the neurons. Any plots that are missing or that remain in the original folder will be classified as not having been sorted yet.

Value

None

See Also

After sorting the plots saved by this function, the user should call `updateNeurons`. For other functions useful in the classification process, see `reviewNeuronsMoreFrames` and `reviewOverlappingNeurons`. Once classification is finished, the argument `neuronsToOutline="kept"` can be used with `plotBrightest` and `plotVideoVariance`, and the argument `neuronsToDisplay="kept"` can be used with `plotResults`, `plotResultsAllLambda`, `plotTemporal`, and `plotSpatial`. Finally, the argument `excludeReps="discarded"` allows the discarded dictionary elements to be excluded from the sparse group lasso model when running `scalpelStep3`.

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function

#we review the set of spatial components from Step 3,
#which are contained in scalpelOutput$Afilter
reviewNeurons(scalpelOutput = scalpelOutput, neuronSet = "Afilter")

## End(Not run)
```

`reviewNeuronsInteractive` *Manually classify the identified neurons from SCALPEL.*

Description

We interactively review the set of identified neurons that result from either Step 2 or 3 of SCALPEL in order to manually classify them according to whether they appear to be real neurons or not. To do this, the frame from which the dictionary element was derived is plotted. The user can manually classify the neuron as real or not, or indicate that additional frames are needed to make the classification, in which case the reviewNeuronsMoreFrames function can subsequently be used. A similar manual classification can be done non-interactively using reviewNeurons.

Usage

```
reviewNeuronsInteractive(scalpelOutput, neuronSet)
```

Arguments

scalpelOutput	An object returned by one of the SCALPEL functions: scalpel, scalpelStep2, or scalpelStep3.
neuronSet	The set of neurons that should be reviewed: use "A" for those resulting from scalpelStep2 and saved as scalpelOutput$A, or use "Afilter" for those resulting from scalpelStep3 and saved as scalpelOutput$Afilter. This argument is ignored if the class of scalpelOutput is scalpelStep2.

Value

None

See Also

For other functions useful in the classification process, see reviewNeuronsMoreFrames, reviewOverlappingNeurons, and updateNeuronsInteractive. Once classification is finished, the argument neuronsToOutline="kept" can be used with plotBrightest and plotVideoVariance, and the argument neuronsToDisplay="kept" can be used with plotResults, plotResultsAllLambda, plotTemporal, and plotSpatial. Finally, the argument excludeReps="discarded" allows the discarded dictionary elements to be excluded from the sparse group lasso model when running scalpelStep3.

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function

#we review the set of spatial components from Step 2,
#which are contained in scalpelOutput$A
reviewNeuronsInteractive(scalpelOutput = scalpelOutput, neuronSet = "A")
#enter "Y" for the first neuron and then "Q"
#entering "Q" allows us to finish manually classifying later using the same command
#this time there are fewer left to review
reviewNeuronsInteractive(scalpelOutput = scalpelOutput, neuronSet = "A")
```

```
#enter "N" for the first and "?" for the second this time
#note that once a neuron is classified as "N", it disappears from the plot

## End(Not run)
```

`reviewNeuronsMoreFrames` *Save additional frames for manually classifying the identified neurons from SCALPEL.*

Description

We use this function after running `reviewNeurons` or `reviewNeuronsInteractive` to plot additional frames for neurons whose classification was unclear from the single frame plotted. The additional frames are saved, and the classification for the neurons can then be updated using `updateNeurons` or `updateNeuronsInteractive`.

Usage

```
reviewNeuronsMoreFrames(scalpelOutput, neuronSet, numFrames = 10)
```

Arguments

`scalpelOutput`	An object returned by one of the SCALPEL functions: `scalpel`, `scalpelStep2`, or `scalpelStep3`.
`neuronSet`	The set of neurons that should be reviewed: use `"A"` for those resulting from `scalpelStep2` and saved as `scalpelOutput$A`, or use `"Afilter"` for those resulting from `scalpelStep3` and saved as `scalpelOutput$Afilter`. This argument is ignored if the class of `scalpelOutput` is `scalpelStep2`.
`numFrames`	The maximum number of frames that should be saved for each neuron being considered. Each neuron has a number of frames equal to the number of members in that neuron's cluster that can be plotted. All frames will be saved when the total number of available frames for the neuron is less than `numFrames`. The default value is 10.

Value

None

See Also

`reviewNeurons`, `updateNeurons`, `reviewNeuronsInteractive`, `updateNeuronsInteractive`

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "reviewNeuronsInteractive" function

#we save frames for the neurons previously classified
#as "?" using the "reviewNeuronsInteractive" function
reviewNeuronsMoreFrames(scalpelOutput = scalpelOutput, neuronSet = "A")

## End(Not run)
```

---

reviewOverlappingNeurons

Save additional frames for overlapping neurons from SCALPEL.

Description

We use this function after running `reviewNeurons` or `reviewNeuronsInteractive` to plot additional frames for neurons that overlap with others. These frames are saved, and the classification for the neurons can then be updated using `updateNeurons` or `updateNeuronsInteractive`.

Usage

```
reviewOverlappingNeurons(scalpelOutput, neuronSet, numFrames = 10)
```

Arguments

`scalpelOutput`	An object returned by one of the SCALPEL functions: `scalpel`, `scalpelStep2`, or `scalpelStep3`.
`neuronSet`	The set of neurons that should be reviewed: use `"A"` for those resulting from `scalpelStep2` and saved as `scalpelOutput$A`, or use `"Afilter"` for those resulting from `scalpelStep3` and saved as `scalpelOutput$Afilter`. This argument is ignored if the class of `scalpelOutput` is `scalpelStep2`.
`numFrames`	The maximum number of frames that should be saved for each neuron being considered. Each neuron has a number of frames equal to the number of members in that neuron's cluster that can be plotted. All frames will be saved when the total number of available frames for the neuron is less than `numFrames`. The default value is 10.

Value

None

See Also

reviewNeurons, updateNeurons, reviewNeuronsInteractive, updateNeuronsInteractive

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "reviewNeuronsInteractive" function

reviewOverlappingNeurons(scalpelOutput = scalpelOutput, neuronSet = "A")

## End(Not run)
```

---

scalpel *Perform entire SCALPEL pipeline.*

Description

Segmentation, Clustering, and Lasso Penalties (SCALPEL) is a method for neuronal calcium imaging data that identifies the locations of neurons, and estimates their calcium concentrations over time. The pipeline involves several steps, each of which is described briefly in its corresponding function. See scalpelStep0, scalpelStep1, scalpelStep2, scalpelStep3 for more details. Full details for the SCALPEL method are provided in Petersen, A., Simon, N., and Witten, D. (Forthcoming). SCALPEL: Extracting Neurons from Calcium Imaging Data.

Usage

```
scalpel(
  outputFolder,
  rawDataFolder,
  videoHeight,
  minClusterSize = 1,
  lambdaMethod = "trainval",
  lambda = NULL,
  cutoff = 0.18,
  omega = 0.2,
  fileType = "R",
  processSeparately = TRUE,
  minSize = 25,
  maxSize = 500,
  maxWidth = 30,
  maxHeight = 30,
  removeBorder = FALSE,
  alpha = 0.9,
```

```
  thresholdVec = NULL,
  maxSizeToCluster = 3000
)
```

Arguments

`outputFolder`	Step 0 parameter: The existing directory where the results should be saved.
`rawDataFolder`	Step 0 parameter: The directory where the raw data version of Y is saved. The data should be a PxT matrix, where P is the total number of pixels per image frame and T the number of frames of the video, for which the (i,j)th element contains the fluorescence of the ith pixel in the jth frame. To create Y, you should vectorize each 2-dimensional image frame by concatenating the columns of the image frame. If the data is saved in a single file, it should be named "Y_1.mat", "Y_1.rds", "Y_1.txt", or "Y_1.txt.gz" (depending on `fileType`), and if the data is split over multiple files, they should be split into chunks of the columns and named consecutively ("Y_1.mat", "Y_2.mat", etc.; "Y_1.rds", "Y_2.rds", etc.; "Y_1.txt", "Y_2.txt", etc.; or "Y_1.txt.gz", "Y_2.txt.gz", etc.).
`videoHeight`	Step 0 parameter: The height of the video (in pixels).
`minClusterSize`	Step 3 parameter: The minimum number of preliminary dictionary elements that a cluster must contain in order to be included in the sparse group lasso.
`lambdaMethod`	Step 3 parameter: A description of how lambda should be chosen: either `"trainval"` (default), `"distn"`, or `"user"`. A value of `"trainval"` means lambda will be chosen using a training/validation set approach. A value of `"distn"` means lambda will be chosen as the negative of the 0.1% quantile of elements of active pixels (i.e., those contained in at least one dictionary element) of Y. Using `"distn"` is computationally faster than `"trainval"`. Alternatively with `"user"`, the value of lambda can be directly specified using `lambda`.
`lambda`	Step 3 parameter: The value of lambda to use when fitting the sparse group lasso. By default, the value is automatically chosen using the approach specified by `lambdaMethod`. If a value is provided for `lambda`, then `lambdaMethod` will be ignored.
`cutoff`	Step 2 parameter: A value in [0,1] indicating where to cut the dendrogram that results from hierarchical clustering of the preliminary dictionary elements. The default value is 0.18.
`omega`	Step 2 parameter: A value in [0,1] indicating how to weight spatial vs. temporal information in the dissimilarity metric used for clustering. If `omega=1`, only spatial information is used. The default value is 0.2.
`fileType`	Step 0 parameter: Indicates whether raw data is an .rds (default value; `fileType="R"`), .mat (`fileType="matlab"`), .txt (`fileType="text"`), or .txt.gz (`fileType="zippedText"`) file. Any text files should not have row or column names.
`processSeparately`	Step 0 parameter: Logical scalar giving whether the multiple raw data files should be processed individually, versus all at once. Processing the files separately may be preferable for larger videos. Default value is `TRUE`; this argument is ignored if the raw data is saved in a single file.

`minSize, maxSize`
: Step 1 parameter: The minimum and maximum size, respectively, for a preliminary dictionary element with default values of 25 and 500, respectively.

`maxWidth, maxHeight`
: Step 1 parameter: The maximum width and height, respectively, for a preliminary dictionary element with default values of 30.

`removeBorder`
: Step 3 parameter: A logical scalar indicating whether the dictionary elements containing pixels in the 10-pixel border of the video should be removed prior to fitting the sparse group lasso. The default value is `FALSE`.

`alpha`
: Step 3 parameter: The value of alpha to use when fitting the sparse group lasso. The default value is 0.9.

`thresholdVec`
: Optional advanced user argument: Step 1 parameter: A vector with the desired thresholds to use for image segmentation. If not specified, the default is to use the negative of the minimum of the processed Y data, the negative of the 0.1% quantile of the processed Y data, and the mean of these. If there were multiple raw data files that were processed separately, these values are calculated on only the first part of data, and then these thresholds are used for the remaining parts.

`maxSizeToCluster`
: Optional advanced user argument: Step 2 parameter: The maximum number of preliminary dictionary elements to cluster at once. We attempt to cluster each overlapping set of preliminary dictionary elements, but if one of these sets is very large (e.g., >10,000), memory issues may result. Thus we perform a two-stage clustering in which we first cluster together random sets of size approximately equaling `maxSizeToCluster` and then cluster together the representatives from the first stage. Finally, we recalculate the representatives using all of the preliminary dictionary elements in the final clusters. The default value is 3000. If `maxSizeToCluster` is set to `NULL`, single-stage clustering is done, regardless of the size of the overlapping sets. Memory issues may result when using this option to force single-stage clustering if the size of the largest overlapping set of preliminary dictionary elements is very large (e.g., >10,000).

Details

Several files containing data from the pipeline, as well as summaries of each step, are saved in various subdirectories of "outputFolder".

Value

An object of class `scalpel`, which can be summarized using `summary`, used to rerun SCALPEL Steps 1-3 with new parameters using `scalpelStep1`, `scalpelStep2`, and `scalpelStep3`, or can be used with any of the plotting functions: `plotFrame`, `plotThresholdedFrame`, `plotVideoVariance`, `plotCandidateFrame`, `plotCluster`, `plotResults`, `plotResultsAllLambda`, `plotSpatial`, `plotTemporal`, and `plotBrightest`. The individual elements are described in detail in the documentation for the corresponding step: `scalpelStep0`, `scalpelStep1`, `scalpelStep2`, and `scalpelStep3`.

See Also

The individual steps in the pipeline can be run using the `scalpelStep0`, `scalpelStep1`, `scalpelStep2`, and `scalpelStep3` functions. Results can be summarized using `summary`, loaded at a later time

using getScalpel, and plotted using plotResults, plotSpatial, plotTemporal, plotCluster, plotVideoVariance, plotFrame, plotThresholdedFrame, plotCandidateFrame, and plotBrightest.

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#existing folder to save results (update this to an existing folder on your computer)
outputFolder = "scalpelResults"
#location on computer of raw data in R package to use
rawDataFolder = gsub("Y_1.rds", "", system.file("extdata", "Y_1.rds", package = "scalpel"))
#video height of raw data in R package
videoHeight = 30
#run SCALPEL pipeline
scalpelOutput = scalpel(outputFolder = outputFolder, rawDataFolder = rawDataFolder,
                        videoHeight = videoHeight)
#summarize each step
summary(scalpelOutput, step = 0)
summary(scalpelOutput, step = 1)
summary(scalpelOutput, step = 2)
summary(scalpelOutput, step = 3)

## End(Not run)
```

---

scalpelStep0 *Perform Step 0 of SCALPEL.*

Description

This step involves data pre-processing. We read in the raw data version of Y and perform standard pre-processing techniques in order to smooth the data both temporally and spatially, remove the bleaching effect, and calculate a standardized fluorescence.

Usage

```
scalpelStep0(
  outputFolder,
  rawDataFolder,
  videoHeight,
  fileType = "R",
  processSeparately = TRUE
)
```

Arguments

`outputFolder`	The existing directory where the results should be saved.
`rawDataFolder`	The directory where the raw data version of Y is saved. The data should be a PxT matrix, where P is the total number of pixels per image frame and T the number of frames of the video, for which the (i,j)th element contains the fluorescence of the ith pixel in the jth frame. To create Y, you should vectorize each 2-dimensional image frame by concatenating the columns of the image frame. If the data is saved in a single file, it should be named "Y_1.mat", "Y_1.rds", "Y_1.txt", or "Y_1.txt.gz" (depending on `fileType`), and if the data is split over multiple files, they should be split into chunks of the columns and named consecutively ("Y_1.mat", "Y_2.mat", etc.; "Y_1.rds", "Y_2.rds", etc.; "Y_1.txt", "Y_2.txt", etc.; or "Y_1.txt.gz", "Y_2.txt.gz", etc.).
`videoHeight`	The height of the video (in pixels).
`fileType`	Indicates whether raw data is an .rds (default value; `fileType="R"`), .mat (`fileType="matlab"`), .txt (`fileType="text"`), or .txt.gz (`fileType="zippedText"`) file. Any text files should not have row or column names.
`processSeparately`	Logical scalar giving whether the multiple raw data files should be processed individually, versus all at once. Processing the files separately may be preferable for larger videos. The default value is `TRUE`; this argument is ignored if the raw data is saved in a single file.

Details

Several files containing data from this step and a summary of the step are saved in "outputFolder".

Value

An object of class `scalpelStep0`, which can be summarized using `summary`, used to run SCALPEL Step 1 using `scalpelStep1`, or can be used with the plotting functions `plotFrame`, `plotThresholdedFrame`, and `plotVideoVariance`.

- `minRaw`, `maxRaw`, `minDeltaf`, `maxDeltaf`: Minimum and maximum values for the raw and processed videos.
- `partsRaw`, `partsDeltaf`: Vectors indicating the indices of the raw and processed data files, respectively.
- `nFramesRaw`, `nFramesDeltaf`: The number of frames in each part of the raw and processed data.
- `lowThreshold`, `highThreshold`: The default lowest and highest threshold values for image segmentation that may be used in Step 1.
- Other elements: As specified by the user.

See Also

The entire SCALPEL pipeline can be implemented using the `scalpel` function. The other steps in the pipeline can be run using the `scalpelStep1`, `scalpelStep2`, `scalpelStep3` functions. Results from this step can be summarized using `summary`, loaded at a later time using `getScalpelStep0`, and plotted using `plotFrame`, `plotThresholdedFrame`, and `plotVideoVariance`.

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#existing folder to save results (update this to an existing folder on your computer)
outputFolder = "scalpelResultsStepByStep"
#location on computer of raw data in R package to use
rawDataFolder = gsub("Y_1.rds", "", system.file("extdata", "Y_1.rds", package = "scalpel"))
#video height of raw data in R package
videoHeight = 30
#run Step 0 of SCALPEL
Step0Out = scalpelStep0(outputFolder = outputFolder,
                        rawDataFolder = rawDataFolder, videoHeight = videoHeight)
summary(Step0Out)

## End(Not run)
```

`scalpelStep1` *Perform Step 1 of SCALPEL.*

Description

This step involves constructing a spatial component dictionary. We apply a simple image segmentation procedure to each frame of the video in order to derive a dictionary of preliminary dictionary elements. Ideally, this dictionary is a superset of the true spatial components.

Usage

```
scalpelStep1(
  step0Output,
  minSize = 25,
  maxSize = 500,
  maxWidth = 30,
  maxHeight = 30,
  thresholdVec = NULL
)
```

Arguments

`step0Output`	An object of class `scalpel` or `scalpelStep0`, which result from running the `scalpel` or `scalpelStep0` functions, respectively.
`minSize, maxSize`	The minimum and maximum size, respectively, for a preliminary dictionary element with default values of 25 and 500, respectively.

`maxWidth, maxHeight`
: The maximum width and height, respectively, for a preliminary dictionary element with default values of 30.

`thresholdVec`
: Optional advanced user argument: A vector with the desired thresholds to use for image segmentation. If not specified, the default is to use the negative of the minimum of the processed Y data (i.e., `step0Output$highThreshold`), the negative of the 0.1% quantile of the processed Y data (i.e., `step0Output$lowThreshold`), and the mean of these. These automatically chosen thresholds can also be updated using `updateThreshold`.

Details

Several files containing data from this step and a summary of the step are saved in "outputFolder/Step1_version" where `version` is a 5-digit unique identifier that is automatically generated.

Value

An object of class `scalpelStep1`, which can be summarized using `summary`, used to run SCALPEL Step 2 using `scalpelStep2`, or can be used with the plotting function `plotCandidateFrame`.

- `Azero`: A matrix containing the preliminary dictionary elements, where the ith column of `Azero` is a vector of 1's and 0's, indicating whether each pixel is contained in the ith preliminary dictionary element.
- `AzeroFrames`: A vector whose ith element gives the video frame from which the preliminary dictionary element in the ith column of `Azero` was derived.
- `AzeroThreshold`: A vector whose ith element gives the threshold used to obtain the preliminary dictionary element in the ith column of `Azero`.
- `pixelsUse`: A vector with the pixels (i.e., indices of the rows of `Azero`) that are contained in at least one preliminary dictionary element.
- `version`: A 5-digit unique identifier for the output folder name that is automatically generated in this step.
- Other elements: As specified by the user or returned from a previous step.

See Also

The entire SCALPEL pipeline can be implemented using the `scalpel` function. The other steps in the pipeline can be run using the `scalpelStep0`, `scalpelStep2`, `scalpelStep3` functions. Results from this step can be summarized using `summary`, loaded at a later time using `getScalpelStep1`, and plotted using `plotCandidateFrame`.

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpelStep0" function
```

```
#run Step 1 of SCALPEL
Step1Out = scalpelStep1(step0Output = Step0Out)
summary(Step1Out)

## End(Not run)
```

`scalpelStep2` *Perform Step 2 of SCALPEL.*

Description

This step involves refinement of the spatial component dictionary from Step 1. We eliminate redundancy in the spatial component dictionary by clustering together preliminary dictionary elements that represent the same neuron, based on spatial and temporal information.

Usage

```
scalpelStep2(step1Output, cutoff = 0.18, omega = 0.2, maxSizeToCluster = 3000)
```

Arguments

`step1Output`	An object of class `scalpel` or `scalpelStep1`, which result from running the `scalpel` or `scalpelStep1` functions, respectively.
`cutoff`	A value in [0,1] indicating where to cut the dendrogram that results from hierarchical clustering of the preliminary dictionary elements. The default value is 0.18.
`omega`	A value in [0,1] indicating how to weight spatial vs. temporal information in the dissimilarity metric used for clustering. If `omega=1`, only spatial information is used. The default value is 0.2.
`maxSizeToCluster`	Optional advanced user argument: The maximum number of preliminary dictionary elements to cluster at once. We attempt to cluster each overlapping set of preliminary dictionary elements, but if one of these sets is very large (e.g., >10,000), memory issues may result. Thus we perform a two-stage clustering in which we first cluster together random sets of size approximately equaling `maxSizeToCluster` and then cluster together the representatives from the first stage. Finally, we recalculate the representatives using all of the preliminary dictionary elements in the final clusters. The default value is 3000. If `maxSizeToCluster` is set to `NULL`, single-stage clustering is done, regardless of the size of the overlapping sets. Memory issues may result when using this option to force single-stage clustering if the size of the largest overlapping set of preliminary dictionary elements is very large (e.g., >10,000).

Details

Several files containing data from this step and a summary of the step are saved in the folder "outputFolder/Step1_version/Step2_omega_omega_cutoff_cutoff" where `version` is a 5-digit unique identifier that is automatically generated in Step 1 and `omega` and `cutoff` are the user-supplied parameters.

Value

An object of class scalpelStep2, which can be summarized using summary, used to run SCALPEL Step 3 using scalpelStep3, or can be used with the plotting functions plotCluster and plotSpatial.

- A: A matrix containing the dictionary elements, where the ith column of A is a vector of 1's and 0's, indicating whether each pixel is contained in the ith dictionary element.
- repComps: A vector where the ith element indicates which preliminary dictionary element is the ith representive component. That is, A[,i]=step1Output$Azero[,repComps[i]].
- clusterID: A vector whose ith element indicates which of the dictionary elements in A is the representative for the ith preliminary dictionary element.
- overlapSetID: A vector indicating which preliminary dictionary elements overlap, with the ith element giving the group index for the ith preliminary dictionary element.
- treeList: A list of length max(overlapSetID) with the ith element containing an object of class protoclust corresponding to prototype clustering for the preliminary dictionary elements with overlapSetID=i. If two-stage clustering was done for a particular set, then the element will be NULL.
- Other elements: As specified by the user or returned from a previous step.

See Also

The entire SCALPEL pipeline can be implemented using the scalpel function. The other steps in the pipeline can be run using the scalpelStep0, scalpelStep1, scalpelStep3 functions. Results from this step can be summarized using summary, loaded at a later time using getScalpelStep2, and plotted using plotCluster and plotSpatial.

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpelStep1" function

#run Step 2 of SCALPEL
Step2Out = scalpelStep2(step1Output = Step1Out)
summary(Step2Out)

## End(Not run)
```

---

scalpelStep3 *Perform Step 3 of SCALPEL.*

Description

This step involves spatial component selection and temporal component estimation. We estimate the temporal components corresponding to the dictionary elements from Step 2 by solving a sparse group lasso problem with a non-negativity constraint.

Usage

```
scalpelStep3(
  step2Output,
  lambdaMethod = "trainval",
  lambda = NULL,
  minClusterSize = 1,
  alpha = 0.9,
  removeBorder = FALSE,
  excludeReps = NULL
)
```

Arguments

step2Output	An object of class scalpel or scalpelStep2, which result from running the scalpel or scalpelStep2 functions, respectively.
lambdaMethod	A description of how lambda should be chosen: either "trainval" (default), "distn", or "user". A value of "trainval" means lambda will be chosen using a training/validation set approach. A value of "distn" means lambda will be chosen as the negative of the 0.1% quantile of elements of active pixels (i.e., those contained in at least one dictionary element) of Y. Using "distn" is computationally faster than "trainval". Alternatively with "user", the value of lambda can be directly specified using lambda.
lambda	The value of lambda to use when fitting the sparse group lasso. By default, the value is automatically chosen using the approach specified by lambdaMethod. If a value is provided for lambda, then lambdaMethod will be ignored.
minClusterSize	The minimum number of preliminary dictionary elements that a cluster must contain in order to be included in the sparse group lasso. The default value is 1 (i.e., all possible dictionary elements are included).
alpha	The value of alpha to use when fitting the sparse group lasso. The default value is 0.9.
removeBorder	A logical scalar indicating whether the dictionary elements containing pixels in the 10-pixel border of the video should be removed prior to fitting the sparse group lasso. The default value is FALSE.
excludeReps	A vector giving the indices of which dictionary elements to exclude, where the indices refer to the columns of step2Output$A. The default value is NULL and no dictionary elements are excluded. Users may also specify "discarded", which will exclude all dictionary elements discarded using a previous call to reviewNeurons or reviewNeuronsInteractive.

Details

To solve the sparse group lasso problem in this step, we minimize the following over Z with all non-negative elements:

```
0.5*sum((Y - AfilterTilde %*% Z)^2) + lambda*alpha*sum(Z)
+ lambda*(1-alpha)*sum(sqrt(rowSums(Z^2)))
```

where AfilterTilde is a scaled version of Afilter.

Several files containing data from this step and a summary of the step are saved in the folder "outputFolder/Step1_version/Step2_omega_omega_cutoff_cutoff/Step3_lambdaMethod_lambdaMethod_ minClusterSize_minClusterSize_alpha_alpha_removeBorder_removeBorder" where `version` is a 5-digit unique identifier that is automatically generated in Step 1, `omega` and `cutoff` are the user-supplied parameters from Step 2, and `lambdaMethod`, `minClusterSize`, `alpha`, and `removeBorder` are the user-supplied parameters from this step. If dictionary elements were manually excluded using `excludeReps`, this is appended to the folder name.

Value

An object of class `scalpelStep3`, which can be summarized using `summary` and used with the plotting functions `plotResults`, `plotResultsAllLambda`, `plotSpatial`, `plotTemporal`, and `plotBrightest`.

- `Afilter`: A matrix containing the filtered dictionary elements, where the ith column of `Afilter` is a vector of 1's and 0's, indicating whether each pixel is contained in the ith filtered dictionary element. Note that `Afilter` is equivalent to `A` after removing the components excluded due to being on the border (if `removeBorder=TRUE`) or having fewer preliminary dictionary elements in their cluster than `minClusterSize`.
- `Zhat`: A matrix containing the estimated temporal components, where the ith row of `Zhat` is the estimated calcium trace corresponding to the ith spatial component (i.e., the ith column of `Afilter`).
- `lambda`: The value of lambda used in fitting the sparse group lasso.
- `ZhatList`: A list of matrices containing the estimated temporal components for alternative values of `lambda` specified in `lambdaSeq`. These can be plotted using `plotResultsAllLambda`.
- `lambdaSeq`: A vector with length equaling the length of `ZhatList`, where the ith element indicates the value of lambda corresponding to the temporal components in `ZhatList[[i]]`.
- `clustersUse`: A vector with length equaling the number of columns of `Afilter`, where the ith element indicates which column of `step2Output$A` the ith column of `Afilter` equals.
- Other elements: As specified by the user or returned from a previous step.

See Also

The entire SCALPEL pipeline can be implemented using the `scalpel` function. The other steps in the pipeline can be run using the `scalpelStep0`, `scalpelStep1`, `scalpelStep2` functions. Results from this step can be summarized using `summary`, loaded at a later time using `getScalpelStep3`, and plotted using `plotSpatial`, `plotTemporal`, `plotResults`, and `plotBrightest`.

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpelStep2" function

#run Step 3 of SCALPEL
Step3Out = scalpelStep3(step2Output = Step2Out)
summary(Step3Out)
```

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

---

summary *Summarize results from SCALPEL pipeline.*

Description

Prints the parameters used and a summary of results for a specified step of SCALPEL.

Usage

```
## S3 method for class 'scalpelStep0'
summary(object, ...)

## S3 method for class 'scalpelStep1'
summary(object, ...)

## S3 method for class 'scalpelStep2'
summary(object, ...)

## S3 method for class 'scalpelStep3'
summary(object, ...)

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

Arguments

object	An object returned by one of the SCALPEL functions: scalpel, scalpelStep0, scalpelStep1, scalpelStep2, or scalpelStep3.
...	Additional arguments to be passed, which are ignored in this function.
step	The SCALPEL step (0, 1, 2, or 3) that you wish to summarize. This is only needed if summarizing an object of class scalpel.

Value

None

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the examples for the "scalpelStep0" and "scalpel" functions
summary(Step0Out)
```

```
 #summarize each step
 summary(scalpelOutput, step = 0)
 summary(scalpelOutput, step = 1)
 summary(scalpelOutput, step = 2)
 summary(scalpelOutput, step = 3)

## End(Not run)
```

---

updateNeurons *Update the classifications of neurons from SCALPEL.*

Description

This function allows the user to update the classifications of neurons, based on manual sorting of plots saved as a result of running reviewNeurons.

Usage

```
updateNeurons(scalpelOutput, neuronSet)
```

Arguments

scalpelOutput	An object returned by one of the SCALPEL functions: scalpel, scalpelStep2, or scalpelStep3.
neuronSet	The set of neurons that should be reviewed: use "A" for those resulting from scalpelStep2 and saved as scalpelOutput$A, or use "Afilter" for those resulting from scalpelStep3 and saved as scalpelOutput$Afilter. This argument is ignored if the class of scalpelOutput is scalpelStep2.

Value

None

See Also

reviewNeurons, reviewNeuronsMoreFrames, reviewOverlappingNeurons

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "reviewNeurons" function

updateNeurons(scalpelOutput = scalpelOutput, neuronSet = "Afilter")

## End(Not run)
```

`updateNeuronsInteractive` *Update the classifications of specified neurons from SCALPEL.*

Description

This function allows the user to update the classifications of neurons, which were reviewed previously using `reviewNeuronsInteractive`. Typically, this function is used after running `reviewNeuronsMoreFrames`.

Usage

```
updateNeuronsInteractive(scalpelOutput, neuronSet)
```

Arguments

`scalpelOutput`	An object returned by one of the SCALPEL functions: `scalpel`, `scalpelStep2`, or `scalpelStep3`.
`neuronSet`	The set of neurons that should be reviewed: use `"A"` for those resulting from `scalpelStep2` and saved as `scalpelOutput$A`, or use `"Afilter"` for those resulting from `scalpelStep3` and saved as `scalpelOutput$Afilter`. This argument is ignored if the class of `scalpelOutput` is `scalpelStep2`.

Value

None

See Also

`reviewNeuronsInteractive`, `reviewNeuronsMoreFrames`, `reviewOverlappingNeurons`

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "reviewNeuronsInteractive" function

updateNeuronsInteractive(scalpelOutput = scalpelOutput, neuronSet = "A")
#you will be prompted for the changes you wish to make

## End(Not run)
```

updateThreshold *Review and update the chosen threshold for image segmentation in Step 1 of SCALPEL.*

Description

We plot random frames from the video processed in Step 0 of SCALPEL with shading to indicate the smallest of the automatically chosen thresholds that will be used to perform image segmentation in Step 1 of SCALPEL. The user is given the option to try out different thresholds and if desired, update the threshold to use.

Usage

```
updateThreshold(step0Output)
```

Arguments

step0Output	An object of class scalpel or scalpelStep0, which result from running the scalpel or scalpelStep0 functions, respectively.

Value

An object identical to step0Output, except it may (depending on the user's decision) have lowThreshold updated.

See Also

scalpelStep0

Examples

```
## Not run:
### many of the functions in this package are interconnected so the
### easiest way to learn to use the package is by working through the vignette,
### which is available at ajpete.com/software

#assumes you have run the example for the "scalpel" function

#update the smallest threshold used for image segmentation in Step 1
scalpelOutput = updateThreshold(step0Output = scalpelOutput)

## End(Not run)
```

Index