The ssWavelets Package

May 22, 2017

R topics documented:

ssWavelets-package 1
befp34 4
covMODWT 5
hfsMODWT 7
hfsPlot 8
palMODWT 10
plotLevel2D 11
plotMODWT2D 13
ssCovMODWT 15
ssCovMODWT-class 16
ssCovMODWT-methods 18
ssEnergy 18
ssMODWT 19
ssMODWT-class 20
ssMODWT-methods 21
ssReflect 22
ssToroid 23
ssWavelet-class 25
varPlot 26

Index **28**

`ssWavelets-package` *\packageTitlessWavelets*

Description

This package adds several classes, generics and associated methods as well as a few various functions to help with wavelet decomposition of sampling surfaces generated using **sampSurf**. As such, it can be thought of as an extension to **sampSurf** for wavelet analysis.

At present, the maximal overlap discrete wavelet transform (MODWT) as implemented in the R package **waveslim** is currently supported. Because objects of class "`sampSurf`" primarily contain results that are simply raster images, the two-dimensional routines from **waveslim** are used exclusively (though the 2-D pyramid algorithm employs the 1-D routines behind the scenes). Also, because one is interested in working with the full image rather than compression, methods such as the discrete wavelet transform (DWT) seem to make little sense on these data. However, other

wavelet filters may prove useful in the future, and the package has been designed for easy extension to other methods.

Much of the package is geared towards decomposition of the sample variance; that is, the total variance of the sampSurf raster object. The idea is to decompose the sample (sampSurf) variance by scale (distance) to see how sampling methods compare at different scales. This distance-based decomposition gives an idea of the variation at scales that is unavailable in the overall sampSurf variance.

The "raw" wavelet decomposition is available in the objects, but its use seems limited in the simulated sampling context at this point. Plotting methods are available for both, but analysis functions are geared towards variance decomposition.

A rudimentary S4 class structure has been established for extending the current classes should one desire to do so.

Details

Package:	ssWavelets
Type:	Package
Version:	0.1
Date:	12-Jan-2017
License:	GPL (>= 3)
LazyData:	TRUE

A list of the main resources are described in the following sections. Please refer to the links below for more information. The package vignette provides further details and examples on how to use this package.

Classes For Use In ssWavelets...

The following S4 classes are available.

The "ssWavelet" class...:

Objects of the non-virtual classes below can be created using constructor functions of the same name, see the *Object Constructors* section for details.

`ssWavelet`	Virtual base class for the following
`ssMODWT`	A class for MODWT decomposition
`ssCovMODWT`	A class for covariance decomposition

Object Constructors

For each of the non-virtual classes defined in the table above, we must be able to create objects that can be used in R. This is done using class-specific "constructor" methods that take the drudgery away from creating what can often be somewhat complicated (with all the graphical components) new object instances. Eventually, there may be more than one constructor for a given class of object, and these are differentiated by the method signature; see the links provided below for more details.

"ssWavelet" class constructors...:

`ssMODWT`	Constructor for individual `ssMODWT` objects
`ssCovMODWT`	Constructor for individual `ssCovMODWT` objects

Summary and Plotting Methods

The objects created above have graphical content made possible by using classes from the sp or raster packages. Therefore, functions have been developed to allow for graphical display of these objects to show different aspects of the wavelet decomposition (i.e., raw coefficients, variance, covariance).

In addition, summary and show methods are also available (sometimes producing the same result) for printing information within the object...

plotMODWT2D	Plots the MODWT object at a given level and anisotropic decomposition
plotLevel2D	Plots the set of MODWT object decompositions at chosen level
show	To succinctly print one of the package objects
summary	To summarize one of the package objects

Note that plot3D is used to plot "raster" and "sampSurf" objects using package **rgl**. However, it can also be used for "ssWavelet" object images by first 'casting' the desired matrix representation to an object of class "RasterLayer". Casting is necessary as there is no method written specifically for objects of class "ssWavelets", since the number of possible images in these objects is large. Examples of this procedure are given in the vignette.

Other Functions

A few other functions that will not normally be required listed below. These include some plotting methods in addition to those listed above...

covMODWT	Much like the R **stats** routine cov, but for wavelets
hfsMODWT	For comparing different sampling methods by scale
hfsPlot	Plots the results of hfsMODWT: An H. F. Smith plot
varPlot	A variance plot using results from hfsMODWT
palMODWT	A nice palette function for display
ssEnergy	Average "energy" for a "sampSurf" object
ssToroid	Returns the toroidal boundary correction object
ssReflect	Returns the reflection (periodization) boundary correction object

Data Sets

Data sets included in the package...

befp34	Mapped tree data from the Bartlett Experimental Forest.

Author(s)

Author & Maintainer: Jeffrey H. Gove, <jgove@fs.fed.us>

References

Gove, J. H. 2017. A User's Guide to the ssWavelets Package. http://sswavelets.r-forge.r-project.org/.

See Also

sampSurf

---

befp34 *Plot 34 Bartlett Density Study*

Description

This data set comes from 1989 remeasurements on the 'Density Study' at the Bartlett Experimental Forest, Bartlett, NH, USA. The Bartlett Experimental Forest is part of the White Mountain National Forest.

Usage

```
data("befp34")
```

Format

The format is: chr "befp34"

Details

The 'Density Study' is one of many silvicultural treatments in the Bartlett Experiment Forest, Bartlett, NH, USA. The study was designed with the idea of applying different treatments to existing even-aged stands with the intent of eventual conversion to uneven-aged. More details may be found in Leak and Solomon (1975) or Leak and Gove (2008). The original treatment for plot 34 was cutting to 40 square feet of basal area per acre with 30 percent of the residual basal area in sawtimber trees over 10.5 inches dbh.

Note that this study was conducted in 'English' (U.S. Customary) units, therefore all locations and measurements are in this system. It is simple enough to convert to metric if desired. These data have left it in 'English' because it gives better detail in the "sampSurf" and subsequent "ssMODWT" image results.

The columns or variables in the data frame are those required to create an object of class standingTrees, plus a few extras...

plotNo The plot number.

treeNo The tree number within this plot.

x,y The tree location coordinates in feet. The northwest corner of the physical plot is at $(0, 0)$.

spp.num The species number corresponding to the codes below.

species The species code. Legal species codes are...

'HE'	Eastern hemlock (*Tsuga canadensis*)
'BE'	American beech (*Fagus grandifolia*)
'PB'	Paper birch (*Betula papyrifera*)
'RM'	Red maple (*Acer rubrum*)
'RS'	Red spruce (*Picea rubens*)
'SM'	Sugar maple (*Acer saccharum*)
'WA'	White ash (*Fraxinus americana*)
'YB'	Yellow birch (*Betula alleghaniensis*)
'OT'	Other species

dbh Diameter at breast height in inches.

height A modeled height in feet; the 'all species' equation in Fast and Ducey (2011) was used.

solidType A randomly generated solid type in the range $[2, 4]$; see the details for the slot of the same name in `standingTree`.

topDiam The top diameter in inches. These should all be zero, tapering to a 'tip' for all species.

Source

The full data set will eventually be available online. It is currently available from the package author.

References

Please refer to the package vignette for examples of the use of this data set.

The Bartlett Experimental Forest.

The White Mountain National Forest, Eastern Region, R9.

Leak, W. B. and D. S. Solomon. 1975. Influence of Residual Stand Density on Regeneration of Northern Hardwoods. USDA Forest Service Research Paper NE-310. Northeastern Forest Experiment Station, Upper Darby, PA. https://www.treesearch.fs.fed.us/pubs/15417.

Leak, W. B. and J. H. Gove. 2008. Growth of Northern Hardwoods in New England: a 25-Year Update. *Northern Journal of Applied Forestry* **25**(2):103–105.

Fast, A. J. and M. J. Ducey. 2011. Height-Diameter Equations for Select New Hampshire Tree Species. *Northern Journal of Applied Forestry* **28**(3):157–160.

Examples

```
#
#LazyData, so just use it; e.g.,...
#
dim(befp34)
head(befp34)
```

`covMODWT` *Function to Calculate (Co-) Variances for MODWT Decompositions*

Description

This routine calculates various variances *for* objects of class "`ssMODWT`", or covariances *for* objects of class "`ssCovMODWT`". It is rather complex, so please see the *Details* section and the vignette referenced below for more information. Please also see the caveat in *Details* concerning the general use of this function.

Usage

```
covMODWT(ss.modwt.x, ss.modwt.y, ...)
```

Arguments

ss.modwt.x	An list object as returned from waveslim::modwt.2d.
ss.modwt.y	A second list object from waveslim::modwt.2d for covariance calculations. If variance calculations are desired, then leave this argument out as the routine tests for this argument as missing and makes a second copy of it from the first argument for variance calculation.
...	Gobbled.

Details

This routine acts much like the stats::cov routine in the sense that if the second argument, ss.modwt.y is missing, then it calculates the variance instead of the covariance. The return list object from this routine is quite long and complicated and is explained in adequate detail in the vignette.

Please note that this routine is an integral part of the constructor methods for both "ssMODWT" and "ssCovMODWT" objects, where the results are automatically incorporated into the correct slot in these objects. Because of this, there is very little need to call this routine on your own, in fact, it is discouraged.

Value

This is a very succinct summary, please see the vignette and/or code for details. A list with components...

isCovariance:	TRUE: covariances; FALSE: variances
summary:	a list of marginal total (co-) variances (vectors)
total:	a list of surface decomposition total (co-) variances (scalars) as well as the smooth (surface) mean
image:	a list of matrix/image (co-) variances

Note

Again, there should be little need to use this outside of the constructor functions. If one is extending the classes in this package and requires the calculation of (co-) variances, then please see the extant constructors for detailed examples. For this reason, no examples are presented below.

Author(s)

Jeffrey H. Gove

See Also

"ssMODWT", "ssCovMODWT".

hfsMODWT *Prepare "ssMODWT" Objects for Further Summary Analysis*

Description

This function takes a number of "ssMODWT" result objects and sets up a data frame with the isotropic variances for each level in the decomposition. It also includes total variance, average energy and surface mean from the decompositions, among others.

This is helpful in further analysis of the H. F. Smith type (using the companion `hfsPlot`), either by model estimation, plotting, or both.

Usage

```
hfsMODWT(..., ids = NA, long = TRUE, runQuiet = TRUE)
```

Arguments

`...`	Any number of "ssMODWT" objects.
`ids`	A character vector of simplified ids for each of the "..." objects. The names of the objects will be used by default (`ids = NA`). The identifying names (`ids`) should correspond to some measure of the size of the inclusion zone; e.g., for circular plots, these might be `ids = c('cp.8', 'cp.10')` for radii of 8 and 10m (ft), respectively.
`long`	`TRUE`: The results are a data frame in "long" format (more useful for, e.g., plotting with **lattice**); `FALSE`: the results are returned in "wide" format.
`runQuiet`	`TRUE`: No feedback when run; `FALSE`: Some summary results.

Details

Normally one run of this routine might correspond to a given sampling method, but with different design parameters like plot size or basal area factor. The reason for this is that then multiple methods (data frames from this function) are passed to one of the plotting methods (see below) and are assigned secondary identifiers to distinguish between sets of sampling methods (i.e., circular plots, horizontal point sampling, etc.). This seemed to be the simplest way to structure it at the time.

Note that one must not mix units of measure ("English" and "metric"') in the "ssMODWT" objects passed. No formal check is made for this, so be good.

The average inclusion zone area is returned in the data frame as the unit of 'effort' corresponding to the various variances. For fixed-area plots, this is constant, for inclusion zones with area that is variable based on some size attribute of the standing trees or downed logs, this is the average over all inclusion areas in the population.

Details of its use are given in the vignette along with examples, which are somewhat lengthy to set up, therefore there are no examples presented below.

Value

A data frame in either wide or long format. The columns for both options are discussed in the vignette.

Author(s)

Jeffrey H. Gove

References

H. F. Smith. 1938. An empirical law describing heterogeneity in the yields of agricultural crops. *Journal of Agricultural Science*, **28**:1–23

See Also

hfsPlot, varPlot

Examples

```
#
# please see the examples in the vignette.
#
```

---

hfsPlot *H. F. Smith Plot*

Description

This routine will generate a lattice plot for the marginal isotropic wavelet variance in the form of an 'H. F. Smith' plot; that is, variance versus the average inclusion zone area increasing on the x-axis for the different sampling methods.

The routine assumes the existence of one or more data frames from hfsMODWT, which takes an "ssMODWT" object set and exports the appropriate data frame—in this case the hfsMODWT output should be in the *long* form.

Usage

```
hfsPlot(...,
        sampMeth = c("Meth.A", "Meth.B"),
        conditionOn = c("iso.j", "tau.j", "j"),
        units = c("metric", "English"),
        showPlot = TRUE,
        fileName = "",
        ylab = "Isotropic Variance",
        xlab = "Average Inclusion Zone Area",
        scales = "same",
        type = "b",
        pch = 19,
        as.table = TRUE,
        theme = c("plain", "custom", "ggplot", "economist"),
        runQuiet = FALSE
       )
```

Arguments

...	Any number of *long-form* data frames as generated from hfsMODWT.
sampMeth	A character vector of sampling method identifiers that will give a label to each of the data frame objects in '...' that is useful for grouping. For example, if the first data frame is from fixed-area plot sampling and the second from critial height sampling, one might use sampMeth = c("FAP","CHS") for the identifiers of these methods.
conditionOn	One of the factor or character columns in the data frame; for columns tau.j and j, they will be converted appropriately.
units	The appropriate units that match the underlying sampling surface. Please be mindful that no checking is possible for the correct units as there is no connection between the original "sampSurf" objects and the data frames used for input.
showPlot	TRUE: display the plot; FALSE: no display.
fileName	The file name for a 'hard copy'; ``'' for no hard copy file of the plot.
ylab	See par.
xlab	See par.
scales	See xyplot.
type	See par.
pch	See par.
as.table	See xyplot.
theme	One of the lattice themes listed in the argument default; see the styles in **latticeExtra**.
runQuiet	TRUE: No feedback when run; FALSE: Some summary results.

Details

The sampMeth argument is helpful for distinguishing between sampling methods if more than one different method is used in the data frame(s) passed. Simply use a vector of blank strings if nothing is desired.

A hard copy to a file may be printed if hardcopyLattice is available (it resides in another of the author's packages and is available on request). Otherwise, simply print/plot the plt component of the return list to a trellis.device as usual for "lattice" objects.

Details of its use are given in the vignette along with examples, which are somewhat lengthy to set up, therefore there are no examples presented below.

Value

A list with...

df:	The concatenated data frames from "...".
plt:	The "lattice" plot object.

Author(s)

Jeffrey H. Gove

References

H. F. Smith. 1938. An empirical law describing heterogeneity in the yields of agricultural crops. *Journal of Agricultural Science*, **28**:1–23

See Also

hfsMODWT, varPlot

Examples

```
#
# please see the examples in the vignette.
#
```

---

palMODWT *Simple Palette Function for "ssWavelets" and "sampSurf"*

Description

This simply gives a pleasing palette with reds for negative (if any) and blues up to brownish for positive; zero should always be white-ish, which was chosen for the background.

Usage

```
palMODWT(n, bias = 5, range = NA, ...)
```

Arguments

n	The length of the palette vector returned.
bias	Please see colorRampPalette.
range	The range of the variable in question to be displayed.
...	Passed to colorRampPalette.

Value

A vector of length n of colors in the palette.

Author(s)

Jeffrey H. Gove

See Also

colorRampPalette

Examples

```
#
# display a simple palette...
#
n = 100
pie(rep(1,n), col=palMODWT(n, range=c(0,100)))
pie(rep(1,n),
    labels = as.character(round(seq(-10, 10, len = n), 1)),
    col = palMODWT(n, range = c(-10, 10)))
```

`plotLevel2D` *Plot a Full Set of Images for a "sampSurf" Wavelet Decomposition*

Description

This routine will plot the full set of isotropic and anisotropic images for a given `level` of decomposition either as the raw wavelet coefficients or various wavelet variances. Both "ssMODWT" and covariance objects, "ssCovMODWT", are supported.

The `level = j` decompositions are plotted in a 2x2 matrix with...

1st row:	Horizontal | Diagonal
2nd row:	Isotropic | Vertical

At `level = J0`, there are five subfigures as the smooth, 'LLJ', is also included centered in the 3rd row.

Usage

```
plotLevel2D(ssMODWT,
            level = 1,
            decompType = c("modwt", "mra"),
            type = c("raw", "var"),
            isoSmooth = TRUE,
            showPlot = TRUE,
            showIZs = TRUE,
            col = NA,
            boxCol = "gray",
            title = NA,
            showLegend = TRUE,
            runQuiet = FALSE,
            ...
           )
```

Arguments

`ssMODWT`	An object of class "ssMODWT" for raw or variance display, or an object of class "ssCovMODWT" for covariance display.
`level`	The decomposition level desired, $j = 1, \ldots, J_0$.

decompType	Either "MODWT" or "MRA" decomposition may be displayed for `type = 'raw'`. Only the former is available for variance or covariance displays.
type	'raw' corresponds to the raw wavelet coefficients. 'var' is for a plot of the variance.
isoSmooth	TRUE: include 'LLJ' in the result for isotropic at level J (i.e., J_0); FALSE: do not include the smooth component.
showPlot	TRUE: display the plot; FALSE: no display.
showIZs	TRUE: display the inclusion zones for the "Stem" objects; FALSE: no display.
col	A vector palette of colors for the surface; if this is NA, the default, then palMODWT is used.
boxCol	Outline color for perimeter box.
title	Add a title to the plot if desired. The default (NA) is a title (and subtitles) constructed from the arguments passed. Otherwise, one can pass their own title or `title = ''` for no title.
showLegend	TRUE: Show the 'legend' identifiers around the outside of the maps; FALSE: suppress the legend.
runQuiet	TRUE: No feedback when run; FALSE: Some summary results.
...	Gobbled.

Details

This routine uses plotMODWT2D to generate each of the individual subfigures, which can be altered using the arguments above that are in common to plotMODWT2D.

Please see the details in plotMODWT2D and the vignette for more information and examples.

Value

The final "RasterLayer" object invisibly.

Author(s)

Jeffrey H. Gove

See Also

plotMODWT2D

Examples

```
#
# 1. creates a sampSurf object with horizontal point sampling
# 2. creates a J_0 = 3-level MODWT decomposition object
# 3. a basic plot...
#
tr = Tract(c(x = 64, y = 64), cellSize = 1) #square tract ~0.5ha
btr = bufferedTract(10, tr)
ag3m = angleGauge(3)        #metric BAF
sshps = sampSurf(10, btr, iZone = 'horizontalPointIZ', angleGauge = ag3m,
        topDiam = c(0,0), startSeed = 123)
modwt.hps = ssMODWT(sshps, J = 3)
## Not run:
```

```
r2 = plotLevel2D(modwt.hps, type = 'var')

## End(Not run)
```

`plotMODWT2D` *Plot Two-Dimensional Wavelet Decomposition on "sampSurf" Objects*

Description

This routine will plot any of the individual "ssMODWT" decompositions below (`waveType`) at any `level` either as the raw wavelet coefficients or various wavelet variances (see Details for some restrictions). Covariance objects, "ssCovMODWT", are also supported.

Usage

```
plotMODWT2D(ssMODWT,
            level = 1,
            waveType = c("ISO", "LH", "HL", "HH", "LL"),
            decompType = c("modwt", "mra"),
            type = c("raw", "var"),
            isoSmooth = TRUE,
            showPlot = TRUE,
            showIZs = TRUE,
            addLattice = FALSE,
            col = NA,
            boxCol = "gray",
            title = NA,
            ...
           )
```

Arguments

`ssMODWT`	An object of class "ssMODWT" for raw or variance display, or an object of class "ssCovMODWT" for covariance display.
`level`	The decomposition level desired, $j = 1, \ldots, J_0$. Note that for `waveType = 'ISO'` and `type = 'var'`, it is legal to specify `j = 0`; please see the details below for an explanation.
`waveType`	'ISO' is for isotropic; the next three are the anisotropic decompositions: 'LH' horizontal; 'HL' vertical; 'HH' diagonal. Finally, the 'LL' is the smooth at the highest scale (level J_0). Please see the vignette for more details.
`decompType`	Either "MODWT" or "MRA" decomposition may be displayed for `type = 'raw'`. Only the former is available for variance or covariance displays.
`type`	'raw' corresponds to the raw wavelet coefficients. 'var' is for a plot of the variance.
`isoSmooth`	`TRUE`: include 'LLJ' in the result for isotropic at level J (i.e., J_0); `FALSE`: do not include the smooth component.
`showPlot`	`TRUE`: display the plot; `FALSE`: no display.
`showIZs`	`TRUE`: display the inclusion zones for the "Stem" objects; `FALSE`: no display.

addLattice	TRUE: create a **lattice** version of the graph (see details); FALSE: no lattice version.
col	A vector palette of colors for the surface; if this is NA, the default, then palMODWT is used.
boxCol	Outline color for perimeter box.
title	Add a title to the plot if desired. The default (NA) is a title (and subtitles) constructed from the arguments passed. Otherwise, one can pass their own title or title = ``'' for no title.
...	Gobbled.

Details

This routine has a fair bit of functionality and it is the basis for the plotLevel2D function as well. The vignette has a number of examples that can be reviewed for more information.

Note that not all combinations of argument options are available. For example, variance is calculated only for decompType = ``MODWT'' as noted above. Also, level = 0 is only meaningful at this point for waveType = ``ISO'' and type = ``var'', which presupposes decompType = ``MODWT''. Also, only type = ``var'' is allowed when the ssMODWT argument passed is an object of class "ssCovMODWT". This latter restriction is because there the covariance is composed of two "ssMODWT" objects; therefore, type = `raw' would be ambiguous. One can always plot the raw wavelets for the individual "ssMODWT" objects that were used to create the covariance object.

With addLattice = TRUE, the actual graphical object is created using levelplot in the **rasterVis** package. Therefore, the latter must be available to use this option. The "lattice" graph is returned and may be plotted as usual.

Value

A list with...

r:	The final "RasterLayer" object.
plt:	The "lattice" plot object if desired; NULL otherwise.

Author(s)

Jeffrey H. Gove

See Also

plotLevel2D

Examples

```
#
# 1. creates a sampSurf object with horizontal point sampling
# 2. creates a J_0 = 3-level MODWT decomposition object
# 3. a basic plot...
#
tr = Tract(c(x = 64, y = 64), cellSize = 1) #square tract ~0.5ha
btr = bufferedTract(10, tr)
ag3m = angleGauge(3)        #metric BAF
sshps = sampSurf(10, btr, iZone = 'horizontalPointIZ', angleGauge = ag3m,
        topDiam = c(0,0), startSeed = 123)
modwt.hps = ssMODWT(sshps, J = 3)
## Not run:
```

```
r1 = plotMODWT2D(modwt.hps, waveType = 'ISO', type = 'var', level = 1)

## End(Not run)
```

---

ssCovMODWT *Generate Objects of Class "ssCovMODWT"*

Description

This is the generic function for class "ssCovMODWT". Please see the associated method in ssCovMODWT-methods for more details.

Usage

```
ssCovMODWT(ssMODWT.a, ssMODWT.b, ...)
```

Arguments

ssMODWT.a	First signature object. Because there is only one extant method, this argument name is derived from the class "ssMODWT," to which it applies.
ssMODWT.b	Second signature object. Again, this argument name is derived from the class "ssMODWT," to which it applies.
...	See the method(s) for details.

Details

Only one method is available for this generic function as noted above. Please see the link above for other arguments that are possible in the constructor.

Value

A valid object of class "ssCovMODWT."

Note

The example below creates an object of class "ssCovMODWT" in such a way that the two sampling methods used are comparable. For example, the same tract and population of standing trees is used for both sampSurf runs. Both the horizontal point and critical height methods use the same angle gauge, though this is not necessary, but nice for illustration. In addition, to make sure the inclusion zones for critical height sampling match those for horizontal point sampling, notice that the reference height is requested to be at DBH rather than ground level (the default), though again this is not strictly necessary, depending on the intent of the simulations. Finally, the ssMODWT decompositions share the same total decomposition level J. The points above that should be routinely followed in all comparisons are to use *(i)* the same tract, *(ii)* the same population of trees and *(iii)* the same decomposition level. Note that it is perfectly acceptable to compare two different basal area factors in HPS, for example, or two different plot sizes for fixed area plot sampling. The results of such comparisons might surprise you.

Author(s)

Jeffrey H. Gove

See Also

"ssMODWT", "sampSurf".

Examples

```
#
# creates a sampSurf object with horizontal point sampling BAF 5,
# then creates a J_0 = 4-level MODWT decomposition object...
#
tr = Tract(c(x = 64, y = 64), cellSize = 1) #square tract ~0.5ha
btr = bufferedTract(10, tr)
ag5m = angleGauge(5)
strees = standingTrees(10, btr, dbhs=c(15,25), topDiam=c(0,0), startSeed = 123)
strees.hps = standingTreeIZs(strees, 'horizontalPointIZ', angleGauge = ag5m)
ss.hps = sampSurf(strees.hps, btr)
modwt.hps = ssMODWT(ss.hps, J = 4)
#
# creates a sampSurf object with critical height sampling BAF 5,
# then creates a J_0 = 4-level MODWT decomposition object...
#
strees.chs = standingTreeIZs(strees, 'criticalHeightIZ', angleGauge = ag5m,
                             referenceHeight='dbh')
ss.chs = sampSurf(strees.chs, btr)
modwt.chs = ssMODWT(ss.chs, J = 4)
#
# now both HPS and CHS are to the same reference height with the
# same population on the same tract, so we can decompose
# the covariance...
#
modwt.cov = ssCovMODWT(modwt.hps, modwt.chs)
#
# take a look at the level j=3 surface...
#
## Not run:
plotMODWT2D(modwt.cov, level=3, type='var')

## End(Not run)
```

---

ssCovMODWT-class *Class* "ssCovMODWT"

Description

This class encompasses the essentials of two ssMODWT classes that share the same decomposition attributes as found in the `levels` slot. The two surfaces produce a set of covariances that are also contained in the object.

The "ssMODWT" objects used to create the class must be from the same underlying tract and population of trees or logs in the sense of a **sampSurf** simulation. The difference in the two lies in applying different sampling methods or, say, plot sizes for circular plot sampling as an example. The wavelet covariance decomposition is then based on the two wavelet decompositions of the "sampSurf" objects.

Examples and information concerning this class are found in the package vignette in the references below. This vignette should be consulted for more details on creating and using objects from this class.

Objects from the Class

Objects are created using the the constructor method ssCovMODWT.

Slots

Two "ssMODWT" objects are used to construct this object. They are variously referred to as the primary or first object, which appears in the "default" fields corresponding to those in "ssMODWT" below, and the second(ary) object. However, the primary often appears with an appended 'a' in the code to distinguish it from the second object, which always appears with an appended 'b' as seen below. I have tried to use [a,b] to distinguish the objects in the code and have stayed away from using [x,y] so as not confuse the latter's use in referring to the dimensions of the images; the exception being in covMODWT and some of the text in the vignette.

ss.b: Object of class "sampSurf": The second sampling surface.

ss.modwt.b: Object of class "list": The raw wavelet coefficient results from running a 2-D MODWT wavelet transformation using modwt.2d in package **waveslim** on the ss.b surface.

ss.mra.b: Object of class "list": The results of applying a 2-D multi-resolution analysis (MRA) to the ss.modwt.b decomposition.

covStats: Object of class "list": A succinct summary of the overall covariances and correlations. These are used when showing or performing a summary on the object.

ss.modwt: Object of class "list": See "ssMODWT"; this is for the first surface in the ss slot.

vars.modwt: Object of class "list": The covariance results. This is a rather long list containing marginal summary, total, and image covariances. Details are provided in the vignette. Please note that this slot does not contain variances as in "ssMODWT", though the two have the same structure.

ss.mra: Object of class "list": See "ssMODWT"; this is for the primary object..

levels: Object of class "list": See "ssMODWT". The information here should match exactly as checked in the constructor for both decomposition objects and is thus the shared result.

description: Object of class "character": See "ssWavelet".

wfName: Object of class "character": See "ssWavelet".

ss: Object of class "sampSurf": See "ssWavelet".

Extends

Class "ssMODWT", directly.
Class "ssWavelet", by class "ssMODWT", distance 2.

Methods

No methods defined with class "ssCovMODWT" in the signature. However the 'summary' method is available as a subclass of "ssMODWT".

Author(s)

Jeffrey H. Gove

See Also

"ssWavelet", "ssMODWT", and package references: ssWavelets and sampSurf

Examples

```
showClass("ssCovMODWT")
```

`ssCovMODWT-methods` *Methods for "ssCovMODWT" object construction in Package* **ssWavelets**

Description

This is the single method for construction of "ssCovMODWT" objects via the generic ssCovMODWT. Simply pass two valid "ssMODWT" objects to the method with other appropriate arguments listed below as desired. If the first and second objects are `identical` one should simply get the variance results back (as a check).

Methods

`signature(ssMODWT.a = "ssMODWT", ssMODWT.b = "ssMODWT")`

usage...

```
ssCovMODWT(ssMODWT.a,
           ssMODWT.b,
         description = 'sampSurf Covariance MODWT wavelet decomposition object',
           runQuiet = FALSE,
           ...)
```

- `ssMODWT.a`: The first object of class "ssMODWT" for the covariance decomposition.
- `ssMODWT.b`: The second object of class "ssMODWT" for the covariance decomposition.
- `description`: A character description as desired.
- `runQuiet`: `TRUE`: no feedback; `FALSE`: some results printed.
- `...`: Arguments currently gobbled.

`ssEnergy` *Calcluate the "Average Energy" from a "sampSurf" Image*

Description

This will calculate the "average energy" surface for the "sampSurf" object slot in an "ssMODWT" or "ssCovMODWT" object. In the case case of covariance, this is simply the average covariance energy.

Usage

```
ssEnergy(ssMODWT, showPlot = TRUE, showIZs = TRUE, boxCol = "gray", col = NA, title = NA, ...)
```

Arguments

`ssMODWT`	Either a valid "ssMODWT" or "ssCovMODWT" object.
`showPlot`	`TRUE`: display the surface; `FALSE`: no display.
`showIZs`	`TRUE`: display the inclusion zones; `FALSE`: don't display.
`boxCol`	Outline color for perimeter box.
`col`	A vector palette of colors for the surface; see, e.g., palMODWT.
`title`	If `showPlot`, a title, or `NA` for nothing.
`...`	Gobbled.

Details

The "average energy" in a scene is essentially the mean sums of squares that is probably more familiar to most users. The former terminology is used by physicists.

The package vignette in the references below gives more information on this subject.

Value

A "Raster" object corresponding to the average energy.

Author(s)

Jeffrey H. Gove

See Also

covMODWT

Examples

```
#
# 1. creates a sampSurf object with horizontal point sampling
# 2. creates a J_0 = 3-level MODWT decomposition object
# 3. calculates the average energy of the surface...
#
tr = Tract(c(x = 64, y = 64), cellSize = 1) #square tract ~0.5ha
btr = bufferedTract(10, tr)
ag3m = angleGauge(3)        #metric BAF
sshps = sampSurf(10, btr, iZone = 'horizontalPointIZ', angleGauge = ag3m,
        topDiam = c(0,0), startSeed = 123)
modwt.hps = ssMODWT(sshps, J = 3)
ae = ssEnergy(modwt.hps, showPlot = FALSE) #showPlot=TRUE to display
```

---

ssMODWT *Generate Objects of Class "ssMODWT"*

Description

This is the generic function for class "ssMODWT". Please see the associated method in ssMODWT-methods for more details.

Usage

```
ssMODWT(ss, ...)
```

Arguments

ss	Signature object. Because there is only one extant method, this argument name is derived from the class "sampSurf," to which it applies.
...	See the method(s) for details.

Details

Only one method is available for this generic function as noted above. Please see the link above for other arguments that are possible in the constructor.

Value

A valid object of class "ssMODWT."

Author(s)

Jeffrey H. Gove

See Also

"ssMODWT", "sampSurf".

Examples

```
#
# creates a sampSurf object with horizontal point sampling
# then creates a J_0 = 5-level MODWT decomposition object...
#
tr = Tract(c(x = 64, y = 64), cellSize = 1) #square tract ~0.5ha
btr = bufferedTract(10, tr)
ag3m = angleGauge(3)        #metric BAF
sshps = sampSurf(10, btr, iZone = 'horizontalPointIZ', angleGauge = ag3m,
        topDiam = c(0,0), startSeed = 123)
modwt.hps = ssMODWT(sshps, J = 5)
```

---

ssMODWT-class *Class* "ssMODWT"

Description

A subclass of virtual class "ssWavelet" that can be used to create objects of that class. These objects contain the results from a two-dimensional maximal overlap discrete wavelet transform (MODWT) that has been applied to the "sampSurf" object stored in the object's ss slot.

Examples and information concerning this class are found in the package vignette in the references below. This vignette should be consulted for more details on creating and using objects from this class.

Objects from the Class

Objects are created using the the constructor method ssMODWT.

Slots

In addition to the slots provided by the virtual superclass "ssWavelet", the following slots are represented...

ss.modwt: Object of class "list": The raw wavelet coefficient results from running a 2-D MODWT wavelet transformation using modwt.2d in package **waveslim**.

`vars.modwt`: Object of class `"list"`: The variance results. This is a rather long list containing marginal summary, total, and image variances. Details are provided in the vignette.

`ss.mra`: Object of class `"list"`: The results of applying a 2-D multi-resolution analysis (MRA) to the `ss.modwt` decomposition.

`levels`: Object of class `"list"`: A list with information on the image dimensions and levels of the decomposition. Details are provided in the vignette.

`description`: Object of class `"character"`: See "ssWavelet".

`wfName`: Object of class `"character"`: See "ssWavelet".

`ss`: Object of class `"sampSurf"`: See "ssWavelet".

Extends

Class `"ssWavelet"`, directly.

Methods

Methods other than ssMODWT for object creation that are applicable to this class include...

ssCovMODWT `signature(ssMODWT.a = "ssMODWT", ssMODWT.b = "ssMODWT")`: Covariance analysis of two "ssMODWT" wavelet decompositions.

summary `signature(object = "ssMODWT")`: Summary of the object.

Author(s)

Jeffrey H. Gove

See Also

"ssWavelet", "ssCovMODWT", and package references: ssWavelets and sampSurf

Examples

```
showClass("ssMODWT")
```

`ssMODWT-methods` *Methods for "ssMODWT" object construction in Package* **ssWavelets**

Description

This is the single method for construction of "ssMODWT" objects via the generic ssMODWT. Simply pass a valid "sampSurf" object to the method with other appropriate arguments listed below as desired.

Methods

signature(ss = "sampSurf")

usage...

```
ssMODWT(ss,
        J = NA,
        wfName = c('haar'),
        reflect = FALSE,
        shift = TRUE,
        trimRight = TRUE,
        description = 'sampSurf MODWT wavelet decomposition object',
        runQuiet = FALSE,
        ...)
```

- ss: An object of class "sampSurf" on which the MODWT decomposition is to be made.
- J: Desired maximum level for the MODWT decomposition, often denoted as J_0.
- wfName:The wavelet filter name. Ostensibly, this should match one of the filters allowed in waveslim::wave.filter. See the class definition for those currently supported.
- reflect: TRUE: reflect the sampSurf results (note: this is in addition to periodic correction, which is always done); FALSE: don't reflect.
- shift: TRUE: apply waveslim::shift.2d to rectify/realign the MODWT images with the original at all levels; FALSE: leave the alignment as is.
- trimRight: When shift=TRUE occasionally the image has come out larger at the first decomposition level that the original image, this will trim the offending extra on the right side and bottom; FALSE: leave as is.
- description: A character description as desired.
- runQuiet: TRUE: no feedback; FALSE: some results printed.
- ...: Arguments currently gobbled.

---

ssReflect *Reflection boundary correction for "sampSurf" Objects*

Description

This routine takes a "sampSurf" object and makes a larger "Raster" object with copies of the original all around it using reflection.

Reflection is specifically used for correction of "edge effect" in the application of a MODWT wavelet filter as discussed in detail in Lark and Webster (2004); see details below.

Usage

```
ssReflect(ss, ...)
```

Arguments

ss	An object of class "sampSurf".
...	Gobbled.

Details

This routine may be used stand-alone, but its main use it to be called from within ssMODWT when constructing objects of class "ssMODWT". In this case, the filter is applied to the extended raster scene encompassing the reflected replicates and then clipped back to the original extents of the "sampSurf" image for further processing.

It should be kept in mind that the ssMODWT constructor uses the **waveslim** code for the decomposition. This code *always* applies a periodic correction to the image. Therefore, if one chooses the reflection as well, the periodic correction is applied to the extended relected image set in the decomposition.

Value

A "Raster" image padded in all directions using the reflection correction.

Author(s)

Jeffrey H. Gove

References

R. M. Lark and R. Webster. 2004. Analysing soil variation in two dimensions with the discrete wavelet transform. *European Journal of Soil Science*, **55**:777–797.

See Also

ssToroid

Examples

```
#
# creates a sampSurf object with horizontal point sampling
# then applies reflection...
#
tr = Tract(c(x = 64, y = 64), cellSize = 1) #square tract ~0.5ha
btr = bufferedTract(10, tr)
ag3m = angleGauge(3)        #metric BAF
sshps = sampSurf(10, btr, iZone = 'horizontalPointIZ', angleGauge = ag3m,
        topDiam = c(0,0), startSeed = 123)
refhps = ssReflect(sshps)    #raster object
## Not run:
plot(refhps, col = palMODWT(100, range=cellStats(refhps, range)))
plot(perimeter(sshps), add = TRUE)

## End(Not run)
```

---

ssToroid *Toroidal boundary correction for "sampSurf" Objects*

Description

This routine takes a "sampSurf" object and makes a larger "Raster" object with copies of the original all around it using toroidal correction, more commonly called periodization or circular correction for the boundary.

Toroidal correction is specifically used for correction of "edge effect" in the application of a MODWT wavelet filter as discussed in detail in Lark and Webster (2004); see details below.

Usage

```
ssToroid(ss, ...)
```

Arguments

ss	An object of class "sampSurf".
...	Gobbled.

Details

This method is meant to be used stand-alone as an illustration for what happens in periodic correction. An example is given below, and a more extensive discussion can be found in the package vignette.

It should be kept in mind that the ssMODWT constructor uses the **waveslim** code for the decomposition. This code *always* applies a periodic correction to the image. Therefore, there is no need to use this routine for this type of correction when creating objects of class "ssMODWT".

I suppose this could be used in **sampSurf** proper as a boundary correction technique for inclusion zones. Perhaps some version of it will make its way into that package in the future to compliment those boundary correction methods already available.

Value

A "Raster" image padded in all directions using the toriodal correction.

Author(s)

Jeffrey H. Gove

References

R. M. Lark and R. Webster. 2004. Analysing soil variation in two dimensions with the discrete wavelet transform. *European Journal of Soil Science*, **55**:777–797.

See Also

ssReflect

Examples

```
#
# creates a sampSurf object with horizontal point sampling
# then does a toroidal correction...
#
tr = Tract(c(x = 64, y = 64), cellSize = 1) #square tract ~0.5ha
btr = bufferedTract(10, tr)
```

```
ag3m = angleGauge(3)        #metric BAF
sshps = sampSurf(10, btr, iZone = 'horizontalPointIZ', angleGauge = ag3m,
        topDiam = c(0,0), startSeed = 123)
torhps = ssToroid(sshps)    #raster object
## Not run:
plot(torhps, col = palMODWT(100, range=cellStats(torhps, range)))
plot(perimeter(sshps), add = TRUE)

## End(Not run)
```

---

ssWavelet-class *Class* "ssWavelet"

Description

This is a virtual class that is the base class for other wavelet classes based on **sampSurf**. It is very simply by design, and can easily be extended to subclasses.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`description`**:** Object of class `"character"`: Some form of identifier.

`wfName`**:** Object of class `"character"`: The wavelet filter name. This should be a legal name from `waveslim::wave.filter`; though for now it is restricted to "haar" wavelets.

`ss`**:** Object of class `"sampSurf"`: the decomposition is based on this object.

Methods

Virtual class: No methods defined with class "ssWavelet" in the signature.

Author(s)

Jeffrey H. Gove

Examples

```
showClass("ssWavelet")
```

varPlot *Summary Wavelet Variance Plot*

Description

This routine creates a plot of the marginal isotropic variances against distance/scale conditioned on either *(i)* each of the different sampling methods, one panel for each method+size (i.e., baf, plot size) combination: `groupSM = FALSE`; or *(ii)* just the `sampMethod` level with each of the 'sizes' grouped: `groupSM = TRUE`. Note that using *(i)* could conceivably get rather busy panel-wise.

Usage

```
varPlot(...,
        sampMeth = c("Meth.A", "Meth.B"),
        groupSM = TRUE,
        units = c("metric", "English"),
        isoSmooth = TRUE,
        showPlot = TRUE,
        fileName = "",
        ylab = "Isotropic Variance",
        xlab = "Distance",
        scales = "same",
        type = "b",
        pch = 19,
        as.table = TRUE,
        theme = c("plain", "custom", "ggplot", "economist"),
        runQuiet = FALSE
       )
```

Arguments

`...`	Any number of *long-form* data frames as generated from `hfsMODWT`.
`sampMeth`	A character vector of sampling method identifiers that will give a label to each of the data frame objects in '`...`' that is useful for grouping. For example, if the first data frame is from fixed-area plot sampling and the second from critial height sampling, one might use `sampMeth = c("FAP","CHS")` for the identifiers of these methods.
`groupSM`	`TRUE`: condition on `sampMeth` and group on the `id` column in the data frames; `FALSE`: condition on `id` and do not group.
`units`	The appropriate units that match the underlying sampling surface. Please be mindful that no checking is possible for the correct units as there is no connection between the original "sampSurf" objects and the data frames used for input.
`isoSmooth`	`TRUE`: include 'LLJ' (smooth) in the result for the isotropic variance at level J; `FALSE`: do not include the smooth component.
`showPlot`	`TRUE`: display the plot; `FALSE`: no display.
`fileName`	The file name for a 'hard copy'; `''` for no hard copy file of the plot.
`ylab`	See `par`.
`xlab`	See `par`.

scales	See xyplot.
type	See par.
pch	See par.
as.table	See xyplot.
theme	One of the lattice themes listed in the argument default; see the styles in **latticeExtra**.
runQuiet	TRUE: No feedback when run; FALSE: Some summary results.

Details

The sampMeth argument is helpful for distinguishing between sampling methods if more than one different method is used in the data frame(s) passed. Simply use a vector of blank strings if nothing is desired.

A hard copy to a file may be printed if hardcopyLattice is available (it resides in another of the author's packages and is available on request). Otherwise, simply print/plot the plt component of the return list to a trellis.device as usual for "lattice" objects.

Note that if groupSM = TRUE then we will have as many panels as sampling methods; if it is FALSE, then the number of panels is as above in *(i)*.

If we want to look at a plot of just the isotropic wavelet variances without the smooth/scale component (LL.var in the "ssMODWT" object) at level J_0, then one can use isoSmooth = FALSE.

Details of its use are given in the vignette along with examples, which are somewhat lengthy to set up, therefore there are no examples presented below.

Value

A list with...

df:	The concatenated data frames from "...".
plt:	The "lattice" plot object.

Author(s)

Jeffrey H. Gove

See Also

hfsMODWT, hfsPlot

Examples

```
#
# please see the examples in the vignette.
#
```

Index

∗Topic **classes**
ssCovMODWT-class, 16
ssMODWT-class, 20
ssWavelet-class, 25
∗Topic **datasets**
befp34, 4
∗Topic **methods**
ssCovMODWT-methods, 18
ssMODWT-methods, 21
∗Topic **package**
ssWavelets-package, 1

befp34, *3*, 4

colorRampPalette, *10*
cov, *3*
covMODWT, *3*, 5, *17*, *19*

hfsMODWT, *3*, 7, *8–10*, *26, 27*
hfsPlot, *3*, *8*, 8, *27*

levelplot, *14*

modwt.2d, *17*, *20*

palMODWT, *3*, 10, *12*, *14*, *18*
par, *9*, *26, 27*
plot3D, *3*
plotLevel2D, *3*, 11, *14*
plotMODWT2D, *3*, *12*, 13

Raster, *19*, *22*, *24*
raster, *3*
RasterLayer, *3*, *12*, *14*

sampSurf, *1*, *4*, *16, 17*, *19–22*, *24*
show, *3*
sp, *3*
ssCovMODWT, *2*, *5, 6*, *11*, *13*, *15*, 15, *17, 18*, *21*
ssCovMODWT,ssMODWT,ssMODWT-method *(*ssCovMODWT-methods*)*, 18
ssCovMODWT,ssMODWT,ssMODWT-method *(*ssMODWT-class*)*, 20
ssCovMODWT-class, 16
ssCovMODWT-methods, 18
ssEnergy, *3*, 18
ssMODWT, *2*, *5–8*, *11*, *13*, *15–19*, 19, *20, 21*, *23*
ssMODWT,sampSurf-method *(*ssMODWT-methods*)*, 21
ssMODWT-class, 20
ssMODWT-methods, 21
ssReflect, *3*, 22, *24*
ssToroid, *3*, *23*, 23
ssWavelet, *2*, *17*, *20, 21*
ssWavelet-class, 25
ssWavelets, *17*, *21*
ssWavelets *(*ssWavelets-package*)*, 1
ssWavelets-package, 1
standingTree, *5*
standingTrees, *4*
summary, *3*, *17*

trellis.device, *9*, *27*

varPlot, *3*, *8*, *10*, 26

xyplot, *9*, *27*