A User's Guide to the ssWavelets Package

J. H. Gove[a]

[a] *USDA Forest Service, Northern Research Station, 271 Mast Road, Durham, NH 03824 USA*
(603) 868-7667; e-mail: jgove@fs.fed.us

Wednesday 24th May, 2017
1:53pm

Contents

1 Introduction

ssWavelets is an R package that is meant to be used in conjunction with the sampSurf package (Gove, 2012) to perform wavelet decomposition on the results of a sampling surface simulation. In general, the *wavelet filter* decomposes the sampSurf simulation results by scale (distance), with each scale corresponding to a different level of the decomposition. This results in a set of anisotropic *wavelet coefficients* for each scale or level, one each for horizontal, vertical and diagonal components;

the sum of these yields an isotropic decomposition for that scale. Residual *smooth* coefficients for the coarsest level are also available from applying the *scaling filter*, and its mean corresponds to the sample mean of the sampling surface from `sampSurf`. Interesting as these 'raw wavelet' decompositions may be, the main interest of the wavelet analysis from a sampling perspective has to do with the sample variance, which is the sampling surface variance from a simulation run of `sampSurf`. The different wavelet decomposition variances corresponding to the raw wavelet coefficients can be easily constructed at each level from the coefficients as we will see in § 2.2 & 4.3. These wavelet variances also correspond to a decomposition of the sample variance, or, its total or average 'energy' (§ 2.2), depending upon the particular level, filter, and use. Covariance analysis, which may prove useful in the comparison of sampling methods at different scales, is discussed in § 2.3 & 4.6.

There are a number of good texts on wavelet analysis. Most cover time series applications, and a few do cover images, but in varying degrees of completeness (actually, most are quite terse compared to the 1-D application in time series/signals). For example, Percival and Walden (2000) is a very good reference, even though they do not cover images at all. Fortunately the **R** `waveslim` package (Whitcher) is used in this book, as well as in another good reference Gençay et al. (2002). Other sources are discussed in the material that follows, many of which are papers rather than books.

The rest of this vignette documents both the main classes and methods available in the `ssWavelets` package, and presents a number of examples illustrating their use on `sampSurf` results. Sections 1.1 and 2 review the notation and some of the basics of wavelet analysis, with special emphasis on the variance. This manual is not a manual on wavelet analysis; therefore, only a brief review is presented here. The main intent is to demonstrate the link between wavelet filtering and its possible uses in the analysis of variance in simulation of sampling methods using `sampSurf` objects. The latter are particularly amenable to this analysis, because wavelet filtering requires raster 'images,' which was inherent in the design of `sampSurf` (though not for this reason).

The `ssWavelets` package is made available to one's session either using `library` or by...

```
R> require(ssWavelets, quietly= TRUE)
```

1.1 Notation

The notation used in the wavelet field is not standardized. I have adopted the notation from Percival and Walden (2000) (& used in Geilhufe et al. (2013) for images) and where applicable some other sources, especially in the context of 2D applications, which these authors do not consider. To being, from Percival and Walden...

J = the largest level available in the discrete wavelet transform (DWT) for the sample size: $N = 2^J$. Note that we may play a little loose with this notation and call J the maximal partial maximal overlap discrete wavelet transform (MODWT) in places rather then J_0.

J_0 = the last level for a partial DWT or MODWT.

j = the level index for the scaling filter, with scale given by τ_j.

L = the width of the wavelet or scaling filter (unit scale). For the Haar wavelet, $L = 2$.

L_j = the width of the jth level wavelet or scaling filter; $L_j \equiv (2^j - 1)(L - 1) + 1$ (Percival and Walden, 2000, p. 96). For the Haar wavelet, this becomes $L_j \equiv (2^j - 1)(2 - 1) + 1 = 2^j$.

M = the number of rows, or cells in the y direction in an image.

N = for signals, this is the (diadic) length of the series. For images, it is the number of columns, or cells in the x direction. Normally we will deal with square images so that $N \equiv M$.

$\tau_j = 2^{j-1}$ is the unitless standardized scale of the jth level wavelet coefficients ($j \geq 1$). Geilhufe et al. (2013) use $\tau_j = 2^{j-1}$ for the wavelet filter and $2\tau_j = 2 \cdot 2^{j-1} = 2^j$ for the scaling filter. In time series analysis, the distance between observations must also be considered, i.e., Δt. In practical applications then τ_j corresponds to a physical scale of $\tau_j \Delta t$, which has meaningful units (Percival and Walden, 2000, p. 59). In image analysis, the physical scale would correspond to distance rather than time (Geilhufe et al., 2013); thus, we would have $\tau'_j = \tau_j \Delta x$, where Δx is the cell extent in the x direction and likewise for y. Again, since `sampSurf` uses square cells, we may write $\tau'_j = \tau_j \Delta xy$ if desired since they are both the same. In what follows, we will use τ_j interchangeably, such that the context will dictate the interpretation. Also, what is used in summaries and figures is τ_j from the wavelet filter.

$M_j = M - L_j + 1$ is the number of non-boundary jth-level coefficients in y.

$N_j = N - L_j + 1$ is the number of non-boundary jth-level coefficients in x.

Note that for both N and M, the references given above begin the image index (or time series) at zero, so that the index ranges to $N - 1$ (for 1D, § 2.2.1) and $M - 1$ (for 2D, § 2.2.2). This convention works quite well with **C** code, since arrays begin at index zero. But **R** does not support indexes beginning at zero, so in **R**, they run from $1, \ldots, N$ and $1, \ldots, M$ depending on the case. Note that Walker (2008, Chapter 4) also uses this latter indexing; his indexing begins with the first row and column at the bottom left of the image, which corresponds well with the `raster` package (Hijmans, 2016), where the origin is at the lower left.

2 Overview: MODWT Wavelet Decomposition

This section presents a very simplified overview of the MODWT wavelet decomposition. Please see the references below for more details. There are many wavelet filters available in `waveslim`, as well as a number of different decompositions. The full list of available filters may be found in the code for the `waveslim::wave.filter` function; the full list of decompositions can be found in the `waveslim` help. Currently `ssWavelets` allows Haar wavelet filters to be used. This restriction

could easily be lifted to any supported in `waveslim`, but right now there seems to be little need for anything more complicated for the analysis of sampling surfaces.

2.1 Matrix approach: Haar wavelets

The matrix approach is used only for illustration, it is not what is used in `waveslim`, and thus this vignette for the decomposition. But it illustrates most simply what is going on in the actual pyramid algorithm used in `waveslim::modwt.2d`.

The Haar wavelet is of length $L = 2$, with $l = 0, \ldots, L-1$ in general. The wavelet filter coefficient vector, $\boldsymbol{h}$, for level $j = 1$ of the decomposition has $2\tau_1 = 2$ coefficients: $\boldsymbol{h}_1 = (h_{1,0}, h_{1,1})' = (-1/\sqrt{2}, 1/\sqrt{2})'$. The corresponding scaling filter coefficient vector, $\boldsymbol{g}_1$, is related to the wavelet filter by (Percival and Walden 2000, p. 75, Geilhufe et al. 2013)

$$g_{1,l} = (-1)^{l+1} h_{1,L-1-l}$$

yielding the vector $\boldsymbol{g}_1 = (g_{1,0}, g_{1,1})' = (1/\sqrt{2}, 1/\sqrt{2})'$. As an aside, Percival and Walden (2000, p. 75) also note that if we are given the scaling filter, then the wavelet filter coefficients can be deduced similarly by

$$h_{1,l} = (-1)^{l} g_{1,L-1-l} \tag{1}$$

This latter method is used in `waveslim::qmf`. If the scaling filter is as given above, then (1) gives wavelet coefficients: $\boldsymbol{h}_1 = (h_{1,0}, h_{1,1})' = (1/\sqrt{2}, -1/\sqrt{2})'$. These coefficients are also used by Geilhufe et al. (2013) and are the reverse of those given above. The order does matter in the sense of how the matrices set up below. Note the indexes on $h_{j,l}$ in the following...

```
R> #Haar wavelet coefficients from scaling coefficeints...
R> (g = c(1/sqrt(2), 1/sqrt(2)))


[1] 0.70710678 0.70710678


R> L = 2
R> l = 0:1
R> (-1)^l


[1]  1 -1


R> L-1-l


[1] 1 0
```

```
R> (-1)^l*rev(g)


[1]  0.70710678 -0.70710678


R> (h = (-1)^l*g[L-1-l])


[1]  0.70710678 -0.70710678


R> waveslim::wave.filter('haar')


$length
[1] 2

$hpf
[1]  0.70710678 -0.70710678

$lpf
[1] 0.70710678 0.70710678
```

In the list returned from `waveslim::wave.filter`, $\boldsymbol{h}_1$ is the `hpf` *(high pass filter)* slot, and similarly, $\boldsymbol{g}_1$ is the `lpf` *(low pass filter)* slot.

The two-dimensional filters obtained from the wavelet and scaling vectors are given by the outer (tensor)[1] product of the vectors as in Lark and Webster (2004); *viz.*,

$$\boldsymbol{hh} = \boldsymbol{h}_1\boldsymbol{h}_1' = \frac{1}{2}\cdot\begin{bmatrix}-\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix}\begin{bmatrix}-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix} = \begin{bmatrix}\frac{1}{4} & -\frac{1}{4}\\ -\frac{1}{4} & \frac{1}{4}\end{bmatrix} \quad \text{wavelet-wavelet: diagonal} \tag{2}$$

$$\boldsymbol{hg} = \boldsymbol{h}_1\boldsymbol{g}_1' = \frac{1}{2}\cdot\begin{bmatrix}-\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix} = \begin{bmatrix}-\frac{1}{4} & -\frac{1}{4}\\ \frac{1}{4} & \frac{1}{4}\end{bmatrix} \quad \text{wavelet-scaling: vertical} \tag{3}$$

$$\boldsymbol{gh} = \boldsymbol{g}_1\boldsymbol{h}_1' = \frac{1}{2}\cdot\begin{bmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix}\begin{bmatrix}-\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix} = \begin{bmatrix}-\frac{1}{4} & \frac{1}{4}\\ -\frac{1}{4} & \frac{1}{4}\end{bmatrix} \quad \text{scaling-wavelet: horizontal} \tag{4}$$

$$\boldsymbol{gg} = \boldsymbol{g}_1\boldsymbol{g}_1' = \frac{1}{2}\cdot\begin{bmatrix}\frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}}\end{bmatrix}\begin{bmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\end{bmatrix} = \begin{bmatrix}\frac{1}{4} & \frac{1}{4}\\ \frac{1}{4} & \frac{1}{4}\end{bmatrix} \quad \text{scaling-scaling: smooth} \tag{5}$$

[1] See, e.g., http://exampleproblems.com/wiki/index.php/Tensor_product

Table 1 presents a cross reference chart that shows how the above filters correspond to the results in `modwt.2d` (first column) and the matrices from Geilhufe et al. (2013). See § 4.2 for more on the `list` structure returned from `modwt.2d`.

Table 1: Key to the `list` structure returned in, e.g., `modwt.2d` and related in **R** package `waveslim` with reference to Geilhufe et al. (2013) for the matrix and filter notation. Note that the matrices returned as the `modwt.2d` "List Elements" below are for levels $j = 1, \ldots, J$ as detailed in § 4.2.

List Element (2^{nd}, 1^{st})	Description	Filters		Matrix
		Type[a] (2^{nd}, 1^{st})	Convolution (2^{nd}·1^{st})	
LHj	horizontal	S–W	$g \cdot h$	$\widetilde{\mathbf{V}}$
HLj	vertical	W–S	$h \cdot g$	$\widetilde{\mathbf{U}}$
HHj	diagonal	W–W	$h \cdot h$	$\widetilde{\widetilde{\mathbf{W}}}$
LLJ	smooth	S–S	$g \cdot g$	$\widetilde{\mathbf{Z}}$

[a] W is the wavelet (H: high-pass) filter (h).
S is the scaling (L: low-pass) filter (g).

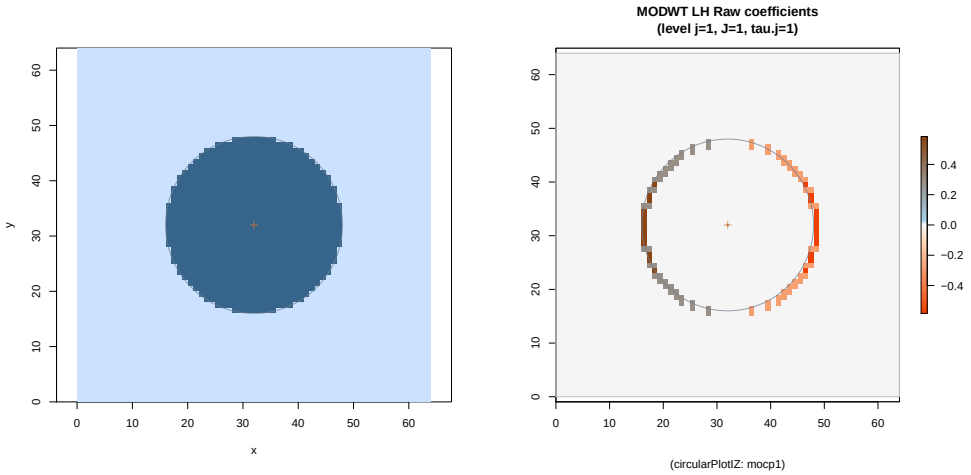
Note that it appears in Geilhufe et al. (2013) that the sense of the filter names (i.e., W-S) are the opposite of what is adopted elsewhere; e.g., Lark and Webster (2004).

An example is presented in Figure 1 for an LH1 decomposition. Note that as the matrix in (4) is evaluated across the scene, it produces a positive value at the western inclusion zone boundary where the surface increases from zero to the estimate height, and a negative value on the eastern edge of the boundary where the reverse situation is encountered (negative of the estimate to zero).

2.2 Wavelet variances

There are several sources on wavelet variance and the sample variance. Most of these are one-dimensional (time series) applications and theory and include Percival and Walden (2000, Chap. 8) and Percival and Mondal (2012). Two-dimensional (image) theory and applications include Mondal and Percival (2012), Geilhufe et al. (2013) and Lark and Webster (2004), the latter of which also includes some information on wavelet covariances when comparing two images (or, in our case, two sampling methods).

There are two main types of variances from wavelet analysis that are discussed in the literature above. The first is called the *wavelet variance* and only involves the wavelet components of the



(a) `sampSurf` (b) MODWT

Figure 1: Sampling surface for single-tree circular plot sampling (see § 3) and corresponding raw wavelet coefficients (right) for the LH1 decomposition from `modwt.2d`.

decomposition (no scaling components are involved—see the *scaling variance* below for more information). The second is called the *sample variance*, which does indeed involve both components, and will be shown below to be equivalent to the `sampSurf` surface variance. The wavelet variance is calculated as part of the sample variance as shown below and is of interest at all scales for its contribution to the decomposition of the `sampSurf` total surface variance.

Finally, the concept of the *energy* in a time series (7) (Percival and Walden (2000, p. 42), Percival and Mondal (2012)) or image (12) (Mondal and Percival, 2012) was evidently coined by physicists. It is equivalent to what is more commonly known as the sums of squares. However, note that the term *average energy* (AE) is used herein to apply to $\mathrm{E}[X^2]$ in (6), because it is what is calculated in the routine `covMODWT`, and seems to be more appropriate for our purposes as a component of the sample variance. Thus average energy is equivalent to the mean sums of squares, or simply mean square. Please be aware that energy is adopted here and that the two terms may be used somewhat interchangeably in what follows.

2.2.1 1-dimensional development

The variance decomposition in one dimension assumes a vector time series $\mathbf{X}$ with N observations X_t ranging from $t = 0, 1, \ldots, N-1$. The sample variance derives from the familiar relation[2]

$$\mathrm{Var}(x) = \mathrm{E}[X^2] - \mathrm{E}[X]^2 \tag{6}$$

[2]See, e.g., `https://en.wikipedia.org/wiki/Algebraic_formula_for_the_variance`.

Now, let the 'energy' in the time series $\mathbf{X}$ be given by[3]

$$\|\mathbf{X}\|^2 = \mathbf{X}'\mathbf{X} = \sum_{t=0}^{N-1} X_t^2 \tag{7}$$

or, substituting the jth-level wavelet $\widetilde{\mathbf{W}}_j$ and scaling $\widetilde{\mathbf{V}}_j$ (smooth) coefficient vectors gives

$$\|\mathbf{X}\|^2 = \|\widetilde{\mathbf{W}}_1\|^2 + \|\widetilde{\mathbf{V}}_1\|^2 \tag{8}$$

for the first-level ($j = 1$) decomposition. the sample variance is given by...

$$\widehat{\sigma}_X^2 = \frac{1}{N}\sum_{t=0}^{N-1}(X_t - \bar{X}_t)^2 \tag{9}$$

$$= \underbrace{\frac{1}{N}\|\mathbf{X}\|^2}_{\mathrm{E}[X^2]} - \underbrace{\bar{X}_t^2}_{\mathrm{E}[X]^2} \tag{10}$$

and substituting (8) gives

$$= \underbrace{\frac{1}{N}\|\widetilde{\mathbf{W}}_1\|^2}_{\widehat{\sigma}_{W_1}^2} + \underbrace{\frac{1}{N}\|\widetilde{\mathbf{V}}_1\|^2 - \bar{X}_t^2}_{\widehat{\sigma}_{V_1}^2} \tag{11}$$

For a J_0th level decomposition, the above extends directly as

$$\|\mathbf{X}\|^2 = \sum_{j=1}^{J_0}\|\widetilde{\mathbf{W}}_j\|^2 + \|\widetilde{\mathbf{V}}_{J_0}\|^2$$

which implies that

$$\widehat{\sigma}_X^2 = \sum_{j=1}^{J_0}\widehat{\sigma}_{W_j}^2 + \widehat{\sigma}_{V_{J_0}}^2$$

where, analogous to the above,

$$\widehat{\sigma}_{W_j}^2 = \frac{1}{N}\|\widetilde{\mathbf{W}}_j\|^2$$

$$\widehat{\sigma}_{V_{J_0}}^2 = \frac{1}{N}\|\widetilde{\mathbf{V}}_{J_0}\|^2 - \bar{X}_t^2$$

[3]Where the squared matrix norm is equivalent to the dot, or inner product; *viz.*, $\|\mathbf{X}\|^2 \equiv \langle \mathbf{X}, \mathbf{X}\rangle$ (see, e.g., https://en.wikipedia.org/wiki/Inner_product_space#Norms_on_inner_product_spaces).

where $\widehat{\sigma}^2_{W_j}$ is the *empirical wavelet variance* and $\widehat{\sigma}^2_{V_{J_0}}$ is the *empirical scaling variance*. More details can be found in the above references, where most of the above comes from Percival and Mondal (2012).

2.2.2 2-dimensional development

The two-dimensional variance decomposition is a straightforward extension of the one-dimensional case. Let $\mathbf{X}_{u,v}$ index an image $\mathbf{X}$ (formally a realization of a random field) with finite extents $u = 0, 1, \ldots, M-1$ rows and $v = 0, 1, \ldots, N-1$ columns (Mondal and Percival, 2012, § V). Mondal and Percival (2012) and Geilhufe et al. (2013) both get more complicated than we will here, allowing for differing levels j and j' (more precisely, scales τ_j and $\tau_{j'}$) for rows and columns, respectively. Here we simply are interested in the diagonal decomposition where $j \equiv j'$, i.e., the level of the decomposition is always the same between rows and columns, vastly simplifying both notation and implementation. Given this, also define $\widetilde{\mathbf{W}}_{j,u,v}$ as the (u,v)th cell of the jth level wavelet-wavelet decomposition matrix $\widetilde{\mathbf{W}}_{j,j}$ (Table 1). Matrices $\widetilde{\mathbf{V}}_{j,u,v}$ & $\widetilde{\mathbf{V}}_{j,j}$, $\widetilde{\mathbf{U}}_{j,u,v}$ & $\widetilde{\mathbf{U}}_{j,j}$, and $\widetilde{\mathbf{Z}}_{j,uv}$ & $\widetilde{\mathbf{Z}}_{j,j}$, all have similar definitions.

The decomposition again starts with (6), such that the total energy in the image is given by the sums of squares of all elements (Mondal and Percival, 2012, § V)

$$\|\mathbf{X}\|^2 = \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} \mathbf{X}^2_{u,v} \tag{12}$$

and again (assuming for the moment that $J_0 = 1$), substituting in the image decomposition matrices for $j = 1$ gives

$$= \|\widetilde{\mathbf{W}}_1\|^2 + \|\widetilde{\mathbf{U}}_1\|^2 + \|\widetilde{\mathbf{V}}_1\|^2 + \|\widetilde{\mathbf{Z}}_1\|^2 \tag{13}$$

as the total energy in the scene. Note from the above that the wavelet transformation is energy preserving (Mondal and Percival, 2012). It follows that in two dimensions, the sample variance for level $j = 1$ is given by

$$\widehat{\sigma}^2_X = \frac{1}{NM} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} (\mathbf{X}_{u,v} - \bar{X})^2 \tag{14}$$

$$= \underbrace{\frac{1}{NM}\|\mathbf{X}\|^2}_{\mathrm{E}[X^2]} - \underbrace{\bar{X}^2}_{\mathrm{E}[X]^2} \tag{15}$$

and substituting (13) gives

$$= \underbrace{\frac{1}{NM}\left(\|\widetilde{\mathbf{W}}_1\|^2 + \|\widetilde{\mathbf{U}}_1\|^2 + \|\widetilde{\mathbf{V}}_1\|^2\right)}_{\widehat{\sigma}^2_{WUV_1}} + \underbrace{\frac{1}{NM}\|\widetilde{\mathbf{Z}}_1\|^2 - \bar{X}^2}_{\widehat{\sigma}^2_{Z_1}} \tag{16}$$

This generalizes in a straightforward way to any level J_0 as

$$\|\mathbf{X}\|^2 = \sum_{j=1}^{J_0}\left(\|\widetilde{\mathbf{W}}_j\|^2 + \|\widetilde{\mathbf{U}}_j\|^2 + \|\widetilde{\mathbf{V}}_j\|^2\right) + \|\widetilde{\mathbf{Z}}_{J_0}\|^2 \tag{17}$$

giving (Mondal and Percival, 2012, first equation, p. 543)

$$\widehat{\sigma}^2_X = \frac{1}{NM}\left(\sum_{j=1}^{J_0}\left(\|\widetilde{\mathbf{W}}_j\|^2 + \|\widetilde{\mathbf{U}}_j\|^2 + \|\widetilde{\mathbf{V}}_j\|^2\right) + \|\widetilde{\mathbf{Z}}_{J_0}\|^2\right) - \bar{X}^2 \tag{18}$$

which again is in the form $\mathrm{E}[X^2] - \mathrm{E}[X]^2$, and implies that

$$\widehat{\sigma}^2_X = \sum_{j=1}^{J_0}\widehat{\sigma}^2_{WUV_j} + \widehat{\sigma}^2_{Z_{J_0}}$$

where, analogous to the above,

$$\widehat{\sigma}^2_{WUV_j} = \frac{1}{NM}\left(\|\widetilde{\mathbf{W}}_j\|^2 + \|\widetilde{\mathbf{U}}_j\|^2 + \|\widetilde{\mathbf{V}}_j\|^2\right) \tag{19}$$

$$\widehat{\sigma}^2_{WUV} = \sum_{j=1}^{J_0}\widehat{\sigma}^2_{WUV_j} \tag{20}$$

$$\widehat{\sigma}^2_{Z_{J_0}} = \frac{1}{NM}\|\widetilde{\mathbf{Z}}_{J_0}\|^2 - \bar{X}^2 \tag{21}$$

such that $\widehat{\sigma}^2_{WUV_j}$ is the jth level *empirical wavelet variance* with total $\widehat{\sigma}^2_{WUV}$, and $\widehat{\sigma}^2_{Z_{J_0}}$ is the *empirical scaling variance*. More details can be found in the above references, where most of the above comes from Mondal and Percival (2012). Note that in their first equation for the diagonal sample variance decomposition, (Mondal and Percival, 2012, p. 543), have the outer parenthesis in the wrong place, which is corrected above in (18). This is also true of their equation (12) on the previous page for the general j, j' decomposition.

Finally, let the *average energy* in the image corresponding to $\mathrm{E}[X^2]$ be given by

$$\widehat{\mathcal{E}} = \frac{1}{NM}\left(\sum_{j=1}^{J_0}\left(\|\widetilde{\mathbf{W}}_j\|^2 + \|\widetilde{\mathbf{U}}_j\|^2 + \|\widetilde{\mathbf{V}}_j\|^2\right) + \|\widetilde{\mathbf{Z}}_{J_0}\|^2\right) \tag{22}$$

which is simply (17) divided by the number of samples (cells) in the image. Thus, we can also write

$$\widehat{\sigma}_X^2 = \widehat{\mathcal{E}} - \bar{X}^2 \tag{23}$$

2.3 Wavelet covariances

The covariance development exactly parallels that for the variance in the last section. Here, however, in two dimensions we have images $\mathbf{X}$ and $\mathbf{Y}$ which have the same raster extents. These two images will have been generated from two different sampling methods applied to the *same* population of trees or downed logs. Our relation (6) generalizes to

$$\text{Cov}(x, y) = \text{E}[X \cdot Y] - \text{E}[X]\text{E}[Y] \tag{24}$$

with total covariance energy—sums of cross products of all elements—given (loosely) as

$$\|\mathbf{XY}\| = \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} \mathbf{X}_{u,v} \mathbf{Y}_{u,v} \tag{25}$$

Basically everything else follows in similar manner from § 2.2.2. There will be two input sets of *raw* wavelet decomposition matrices (one for each method), and the squared terms in § 2.2.2 become the multiplication of the respective components for $\mathbf{X}$ and $\mathbf{Y}$ over all (u, v) as in (25). The result is one output set of wavelet covariance components exactly as described for the variance. In fact, the routine (`covMODWT`) that computes the variances for a one-scene analysis and covariances for a two-scene analysis uses the same covariance code and simply duplicates the single scene in $\mathbf{X}$ to create a replicate $\mathbf{Y}$ for variance calculation.

3 Example Surfaces Used

In the following we generate several surfaces that will be used as illustrations in the remainder of this document. All sampling simulations are volume surfaces. However, any attribute available in `sampSurf` could be used if desired.

As note in § 2.1, the `waveslim` package `modwt.2d` routine for MODWT analysis is used for the decomposition in the `ssMODWT` method used below. The latter also performs a MODWT+MRA (multiresolution analysis) decomposition using `waveslim::mra.2d`. These will be discussed more in § 4.2.

3.1 Simple 1-, 2-, or 3-tree surface

Simple is good for understanding what is going on. In this spirit, we have some simple surfaces for illustration. These all use circular plot sampling; the single-tree surface was used in Figure 1...

```
R> sscp1 = makeSS('ss.1tree', 'circularPlotIZ', plotRadius = 16)


Number of trees in collection = 1
Heaping tree: 1,


R> sscp2 = makeSS('ss.2tree', 'circularPlotIZ', plotRadius = 16, runQuiet = TRUE)
R> sscp3 = makeSS('ss.3tree', 'circularPlotIZ', plotRadius = 16, runQuiet = TRUE)
R> J.cp = 3
R> mocp1 = ssMODWT(sscp1, J = J.cp)


Inclusion zone class: circularPlotIZ
Surface variance = 0.21720573
Wavelet variance = 0.2171527 (MODWT)
Surf/Wave var ratio = 1.0002442
Check: surface var matrix = 0.2171527 = E[X^2] - E[X]^2
Surface mean = 0.23171762
Wavelet mean = 0.23171762 (MODWT)
Wavelet mean = 0.23171762 (MRA)


R> mocp2 = ssMODWT(sscp2, J = J.cp)


Inclusion zone class: circularPlotIZ
Surface variance = 2.9649516
Wavelet variance = 2.9642277 (MODWT)
Surf/Wave var ratio = 1.0002442
Check: surface var matrix = 2.9642277 = E[X^2] - E[X]^2
Surface mean = 1.0431076
Wavelet mean = 1.0431076 (MODWT)
Wavelet mean = 1.0431076 (MRA)


R> mocp3 = ssMODWT(sscp3, J = J.cp)


Inclusion zone class: circularPlotIZ
```

```
Surface variance = 20.917246
Wavelet variance = 20.912139 (MODWT)
Surf/Wave var ratio = 1.0002442
Check: surface var matrix = 20.912139 = E[X^2] - E[X]^2
Surface mean = 2.7554208
Wavelet mean = 2.7554208 (MODWT)
Wavelet mean = 2.7554208 (MRA)
```

In each of the above wavelet decompositions we have used $J = 3$ for the maximum decomposition level.[4]

3.2 Bartlett Experimental Forest plots

The following example uses the 1989 measurement for plot 34 on the 'Density Study' at the Bartlett Experiment Forest, Bartlett, NH, USA (BEF). The Density Study is one of many silvicultural treatments on the BEF. 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 structure. More details may be found in Leak and Solomon (1975) or Leak and Gove (2008); final stand structures are described in Gove et al. (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. ...

```
R> #circular plot sampling
R> ss.bef = ssBEF(befp34, plotRadius = 16, bufferWidth = 20, runQuiet = TRUE)


Input BEF plot = befp34
Number of trees = 77
Units = English
Inclusion zone: circularPlotIZ
Plot width = 130
Buffer width = 20
Plot extent w/ buffer = -20 to 150


R> J.bef = 5                                    #J_0 for BEF
R> mo.bef = ssMODWT(ss.bef$ss, J = J.bef)


Inclusion zone class: circularPlotIZ
```

[4] The `makeSS` function simply facilitates the creation of the sampling surfaces and is available from the author on request.

```
Surface variance = 1260260
Wavelet variance = 1260216.4 (MODWT)
Surf/Wave var ratio = 1.0000346
Check: surface var matrix = 1260216.4 = E[X^2] - E[X]^2
Surface mean = 977.65134
Wavelet mean = 977.65134 (MODWT)
Wavelet mean = 977.65134 (MRA)
```

The routine **ssBEF**[5] simply makes a **sampSurf** object from these data with the appropriate plot size and buffer; it is available on request from the author. Note that all trees are used here, though one can subsample the plot if desired with the **N** argument in **ssBEF**. The maximum MODWT decomposition level for plot 34 is $J = 5$.

The measurements on the Density study were all in 'English' (Imperial) units.[6] One can easily generate a comparable data frame in metric (SI) units using **ssBEF**, which allows for a coarser resolution (Δxy of one meter versus one foot), simplifying the subsequent analysis with fewer grid cells. However, it looks blocky in display so we will stay with English units here. The 16ft (a 50th-acre plot) plot radius is approximately equivalent to 5m.

3.3 Critical height sampling surface

This example is a sampling surface using critical height sample with a metric 3 BAF prism[7]...

```
R> jtr = Tract(c(x = 64, y = 64), cellSize=1) #square tract, dyadic grid ~0.5ha
R> jbtr = bufferedTract(10, jtr)
R> ag3m = angleGauge(3) #metric
R> sschs = sampSurf(20, jbtr, iZone = 'criticalHeightIZ', angleGauge = ag3m,
+                   topDiam = c(0,0), startSeed = 123, referenceHeight = 'dbh')


Number of trees in collection = 20
Heaping tree: 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
```

The wavelet analysis on the critical height surface is...

[5] Available from the author on request.

[6] More appropriately in the United States, U.S. Customary Units. See https://en.wikipedia.org/wiki/United_States_customary_units for the distinction.

[7] With reference height at DBH so it can be directly compared with other methods like horizontal point sampling (HPS).

```
R> J.chs = 5
R> modwtchs.p = ssMODWT(sschs, reflect=FALSE, J=J.chs)


Inclusion zone class: criticalHeightIZ
Surface variance = 52.440687
Wavelet variance = 52.427884 (MODWT)
Surf/Wave var ratio = 1.0002442
Check: surface var matrix = 52.427884 = E[X^2] - E[X]^2
Surface mean = 4.6024029
Wavelet mean = 4.6024029 (MODWT)
Wavelet mean = 4.6024029 (MRA)
```

3.4 Summary

In each of the above results from the MODWT analysis (which includes a MODWT+MRA § 4.5), note that both the wavelet means are the same as that for the actual sampling surface. The variance is also very close for the MODWT analysis and will be discussed in more detail in § 4.2.[8] This MODWT decomposition uses periodic boundary correction (see § 6 for details).

4 The ssWavelets Class Structure

The main objects used in package **ssWavelets** arise from formal S4 class definitions. The class hierarchy relations are shown in the sections that follow.

4.1 The "ssWavelet" class

The wavelet decomposition classes begin with the virtual "base" class "ssWavelet". Its structure is quite simple and forms a simple basis for other functional classes.

```
R> showClass('ssWavelet')


Virtual Class "ssWavelet" [package "ssWavelets"]

Slots:
```

[8] Of course, variance decomposition does not work on the MRA.

```
Name:  description      wfName          ss
Class:   character   character    sampSurf

Known Subclasses:
Class "ssMODWT", directly
Class "ssCovMODWT", by class "ssMODWT", distance 2
```

The slots in the "ssWavelet" class are...

description: A character description pertaining to the object.

wfName: The wavelet filter name, which must be one of `waveslim::wave.filter`.

```
R> modwtchs.p@wfName

[1] "haar"
```

ss: The sampling surface object on which the decomposition is performed.

4.2 The "ssMODWT" class

The "ssMODWT" class discussed in detail in this section, is a subclass of "ssWavelet"...

```
R> showClass('ssMODWT')


Class "ssMODWT" [package "ssWavelets"]

Slots:

Name:     ss.modwt  vars.modwt      ss.mra      levels description      wfName
Class:        list        list        list        list   character   character

Name:           ss
Class:    sampSurf

Extends: "ssWavelet"

Known Subclasses: "ssCovMODWT"
```

In the above we see that objects of "ssMODWT" inherit three slots from the "ssWavelet" virtual class as described in the previous section. Other slots added in this class are...

ss.modwt: The MODWT analysis performed by `waveslim::modwt.2d` via the `ssMODWT` object constructor method. Note that the results are spatially rectified back to the original scene using `shift.2d` by default (`shift = TRUE`) in the call to `ssMODWT`. This is discussed more in § 4.5.

```
R> names(modwtchs.p@ss.modwt)

 [1] "LH1" "HL1" "HH1" "LH2" "HL2" "HH2" "LH3" "HL3" "HH3" "LH4" "HL4" "HH4"
[13] "LH5" "HL5" "HH5" "LL5"
```

The names of its contents are shown in the above for a level $J = 5$ decomposition. Each of these decomposition matrices are copied exactly from the `list` output of `modwt.2d` and saved in the object.[9] Those for level $j = 1$ are shown in the following...

1. `LH1`: the wavelet "detail" horizontal decomposition for level one.
2. `HL1`: the wavelet "detail" vertical decomposition for level one.
3. `HH1`: the wavelet "detail" diagonal decomposition for level one.
4. `LLJ`: the final "smooth" or average for the last level (coarsest), taking the `mean` of this matrix gives the `LL.mean` discussed in § 4.3.2.

though in general, for $j = 1, \ldots, J$, they follow Table 1. Note that summing the above matrices at each level yields the isotropic decomposition for the level.

vars.modwt: This slot contains a rather extensive list of variances. These are discussed in detail below in § 4.3.

ss.mra: The results from doing a MODWT+MRA analysis using `waveslim::mra.2d` exactly parallel those for the `ss.modwt` slot.

```
R> names(modwtchs.p@ss.mra)

 [1] "LH1" "HL1" "HH1" "LH2" "HL2" "HH2" "LH3" "HL3" "HH3" "LH4" "HL4" "HH4"
[13] "LH5" "HL5" "HH5" "LL5"
```

levels: This is a list with levels and other parameters that are discussed in more detail § 4.4.

[9] The same can be said for other methods in `waveslim` such as `mra.2d`.

4.3 ssMODWT variances

This section details the different variances stored in the "ssMODWT" object. The object has several different wavelet variances that are calculated and stored automatically.

```
R> str(modwtchs.p@vars.modwt, 1)


List of 4
 $ isCovariance: logi FALSE
 $ summary     :List of 4
 $ total       :List of 3
 $ image       :List of 6
```

The following three sections provide details on the makeup of these slots and how they relate to the variances given in § 2.2.2. The `isCovariance` slot is `TRUE` if the variances below were obtained from a covariance analysis between two different sampling methods (see § 7.2) and thus we are dealing with an object of class "ssCovMODWT" in § 4.6.[10] Note in the sections below that when `isCovariance = TRUE`, "variance" can be replaced with "covariance".

4.3.1 summary: Marginal summary variances

The `summary` component of this list contains the 'marginal' wavelet variances. These correspond to the raw wavelet coefficient vectors described in § 4.2. The isotropic variance is also included in addition to the directional wavelet variances...

```
R> str(modwtchs.p@vars.modwt$summary)


List of 4
 $ LH.var : Named num [1:5] 0.876 2.22 4.318 5.265 5.449
  ..- attr(*, "names")= chr [1:5] "LH1" "LH2" "LH3" "LH4" ...
 $ HL.var : Named num [1:5] 0.988 2.684 5.648 6.367 4.02
  ..- attr(*, "names")= chr [1:5] "HL1" "HL2" "HL3" "HL4" ...
 $ HH.var : Named num [1:5] 0.0993 0.3241 1.2986 3.2764 2.6559
  ..- attr(*, "names")= chr [1:5] "HH1" "HH2" "HH3" "HH4" ...
 $ iso.var: Named num [1:5] 1.96 5.23 11.26 14.91 40.25
  ..- attr(*, "names")= chr [1:5] "iso1" "iso2" "iso3" "iso4" ...
```

[10] The `isCovariance` element is a hold-over from the pre-S4 version, which I elected not to change for simplicity. Under S4, a simple class check could replace this.

These are vectors, one component for each level in the decomposition $j = 1, \ldots, J_0$.

Directional variances: The directional *wavelet* variances are `LH.var`, `HL.var` and `HH.var` and do not include any contributions from the smooth at level J_0. These can be computed by taking the mean squares (average energies) of each raw wavelet coefficient matrix at each level. Alternatively, since the matrices of mean squares are found in the `image` slot of the object 4.3.3, one can simply sum the corresponding image matrices to get the components of each vector as demonstrated below.

LH.var: This is the horizontal (scaling-wavelet) variance component at each level; i.e., it is the jth level average energy of the raw coefficients...

$$\begin{aligned}\widehat{\sigma}^2_{V_j} &= \frac{1}{NM}\sum_{u=0}^{N-1}\sum_{v=0}^{M-1}\widetilde{\mathbf{V}}^2_{j,u,v} \qquad j = 1, \ldots, J_0 \\ &= \frac{1}{NM}\|\widetilde{\mathbf{V}}_j\|^2\end{aligned}$$

As noted above, this can also be computed from the average energy in the corresponding `image` matrix, as demonstrated in the following...

```
R> modwtchs.p@vars.modwt$summary$LH.var        #horizontal


       LH1        LH2        LH3        LH4        LH5
0.87579834 2.21966697 4.31772209 5.26465564 5.44890494


R> sapply(modwtchs.p@vars.modwt$image$LH.var, sum)


       LH1        LH2        LH3        LH4        LH5
0.87579834 2.21966697 4.31772209 5.26465564 5.44890494
```

HL.var: The vertical (wavelet-scaling) variance component at each level is given by

$$\begin{aligned}\widehat{\sigma}^2_{U_j} &= \frac{1}{NM}\sum_{u=0}^{N-1}\sum_{v=0}^{M-1}\widetilde{\mathbf{U}}^2_{j,u,v} \qquad j = 1, \ldots, J_0 \\ &= \frac{1}{NM}\|\widetilde{\mathbf{U}}_j\|^2\end{aligned}$$

As demonstrated in the following...

```
R> modwtchs.p@vars.modwt$summary$HL.var       #vertical


       HL1        HL2        HL3        HL4        HL5
0.98770304 2.68413193 5.64757758 6.36724800 4.02045680


R> sapply(modwtchs.p@vars.modwt$image$HL.var, sum)


       HL1        HL2        HL3        HL4        HL5
0.98770304 2.68413193 5.64757758 6.36724800 4.02045680
```

HH.var: The diagonal (wavelet-wavelet) variance component at each level, which is given by

$$\widehat{\sigma}^2_{W_j} = \frac{1}{NM}\sum_{u=0}^{N-1}\sum_{v=0}^{M-1}\widetilde{\mathbf{W}}^2_{j,u,v} \qquad j = 1,\ldots,J_0$$
$$= \frac{1}{NM}\|\widetilde{\mathbf{W}}_j\|^2$$

As demonstrated in the following...

```
R> modwtchs.p@vars.modwt$summary$HH.var       #diagonal


        HH1         HH2         HH3         HH4         HH5
0.099296932 0.324114937 1.298573279 3.276368629 2.655936137


R> sapply(modwtchs.p@vars.modwt$image$HH.var, sum)


        HH1         HH2         HH3         HH4         HH5
0.099296932 0.324114937 1.298573279 3.276368629 2.655936137
```

Isotropic Variances: The isotropic wavelet variance (`iso.var`) for each vector component corresponds to $\widehat{\sigma}^2_{WUV_j}$, for each level $j = 1,\ldots,J_0-1$. Note, however, that the J_0th level adds in the

smooth contribution to the wavelet variance; *viz.*,

$$\begin{aligned}\widehat{\sigma}^2_{\text{iso}_{J_0}} &= \widehat{\sigma}^2_{V_{J_0}} + \widehat{\sigma}^2_{U_{J_0}} + \widehat{\sigma}^2_{W_{J_0}} + \frac{1}{NM}\|\widetilde{\mathbf{Z}}_{J_0}\|^2 \\ &= \widehat{\sigma}^2_{WUV_{J_0}} + \frac{1}{NM}\|\widetilde{\mathbf{Z}}_{J_0}\|^2 \\ &= \widehat{\sigma}^2_{WUV_{J_0}} + \widehat{\sigma}^2_{Z_{J_0}} + \bar{X}^2\end{aligned}$$

```
R> (i.v = modwtchs.p@vars.modwt$summary$iso.var)      #isotropic


      iso1       iso2       iso3       iso4       iso5
 1.9627983  5.2279138 11.2638729 14.9082723 40.2471393


R> (i.v2 = with(modwtchs.p@vars.modwt$summary, LH.var + HL.var + HH.var))


       LH1        LH2        LH3        LH4        LH5
 1.9627983  5.2279138 11.2638729 14.9082723 12.1252979


R> (LL.var = i.v[J.chs] - i.v2[J.chs])                #S-S variance


     iso5
28.121841


R> #total isotropic average energy over all j...
R> (ae1 = sum(i.v2 + c(rep(0, J.chs - 1), LL.var)))


[1] 73.609997


R> (ae2 = sum(modwtchs.p@vars.modwt$summary$iso.var))


[1] 73.609997
```

Thus, the sum over all levels is the average energy $\widehat{\mathcal{E}}$ (i.e., corresponding to $\mathrm{E}[X^2]$) in (22) as shown above. Subtracting $\bar{X}^2$ from this yields the total sample variance as shown in the following section.

4.3.2 `total`: Totals

The `total` component of this list contains the mean and wavelet variance for the smooth component, and overall sample variance.

```
R> str(modwtchs.p@vars.modwt$total)


List of 3
 $ LL.mean  : num 4.6
 $ LL.var   : num 28.1
 $ modwt.var: num 52.4
```

The overall wavelet sample variance, `modwt.var`, is $\widehat{\sigma}^2_X$ in relation (18). The energy identity (12) is the squared norm (since $\mathbf{X}$ is a matrix with N rows and M columns) giving the sums of squares of the sampling surface values (e.g., Percival and Walden, 2000, p. 42). Thus, to get the overall wavelet sample variance (equal to the `sampSurf` variance) one must subtract off the squared surface mean (`LL.mean`) from the sum of the isotropic variances over all levels. This is shown in the result below, which can be compared with the corresponding slots in the object...

```
R> with(modwtchs.p@vars.modwt, sum(summary$iso.var) - total$LL.mean^2)


[1] 52.427884


R> (modwt.var = modwtchs.p@vars.modwt$total$modwt.var)


[1] 52.427884


R> (ssVar = modwtchs.p@ss@surfStats$var)


[1] 52.440687
```

Note the slight discrepancy between the MODWT and `sampSurf` variances in the above. Recall that the wavelet variances are defined based on a divisor of MN, the dimensions of the entire scene. However, the `sampSurf` variance is the traditional sample variance that is based on $MN - 1$. We can illustrate this difference as follow...

```
R> NM = ncell(modwtchs.p@ss@tract)                    #total image size
R> modwt.var*NM/(NM - 1)                              #equals ssVar


[1] 52.440687
```

The wavelet variance for the smooth term, `LL.var`, is simply $\frac{1}{NM}\|\widetilde{\mathbf{Z}}_{J_0}\|^2$ as demonstrated in § 4.3.1. Finally, $\widehat{\sigma}^2_{Z_{J_0}}$ can be easily calcluated as...

```
R> with(modwtchs.p@vars.modwt$total, LL.var - LL.mean^2)


[1] 6.9397292
```

4.3.3 `image`: Image variances

The "image variances" are the average energy on a *per cell* (u, v) basis, with each set of images indexed over $j = 1, \ldots, J_0$ as applicable; i.e., only J_0 for `LL.var`. Therefore, each of the different components of this list is either another list or a matrix...

```
R> str(modwtchs.p@vars.modwt$image, 1)


List of 6
 $ LH.var     :List of 5
 $ HL.var     :List of 5
 $ HH.var     :List of 5
 $ LL.var     : num [1:64, 1:64] 0.000764 0.00074 0.000703 0.000672 0.000646 ...
 $ iso.var    :List of 5
 $ isoSurf.var: num [1:64, 1:64] 0.00115 0.00119 0.00122 0.00123 0.00123 ...
```

The slots in the lists above each contain image matrices as will be shown and described more fully in the following sections.

In what follows, we define the square of a matrix to be the same as squaring each individual element of the matrix as in R. The elements in each matrix are the individual values of the average energy at (u, v). These might be loosely denoted as $\mathrm{E}[D^2_{j,u,v}]$, where D is one of the wavelet decomposition matrices at level j.

LH.var: This set of matrices corresponds to the average energy in the horizontal decomposition; that is, the $\frac{1}{NM}\widetilde{\mathbf{V}}_j^2, j = 1, \ldots, J_0$ matrices, each with elements (u, v).

```
R> str(modwtchs.p@vars.modwt$image$LH.var)


List of 5
 $ LH1: num [1:64, 1:64] 0 0 0 0 0 0 0 0 0 0 ...
 $ LH2: num [1:64, 1:64] 0 0 0 0 0 0 0 0 0 0 ...
 $ LH3: num [1:64, 1:64] 0 0 0 0 0 0 0 0 0 0 ...
 $ LH4: num [1:64, 1:64] 2.13e-06 4.73e-06 8.64e-06 1.39e-05 1.90e-05 ...
 $ LH5: num [1:64, 1:64] 1.24e-04 9.92e-05 8.59e-05 7.55e-05 6.67e-05 ...
```

HL.var: The $\frac{1}{NM}\widetilde{\mathbf{U}}_j^2, j = 1, \ldots, J_0$ vertical average energy matrices, each with elements (u, v).

```
R> str(modwtchs.p@vars.modwt$image$HL.var)


List of 5
 $ HL1: num [1:64, 1:64] 0 0 0 0 0 0 0 0 0 0 ...
 $ HL2: num [1:64, 1:64] 0 0 0 0 0 0 0 0 0 0 ...
 $ HL3: num [1:64, 1:64] 0 0 0 0 0 0 0 0 0 0 ...
 $ HL4: num [1:64, 1:64] 2.13e-06 4.73e-06 8.64e-06 1.39e-05 2.18e-05 ...
 $ HL5: num [1:64, 1:64] 0.000258 0.000336 0.000399 0.000432 0.000445 ...
```

HH.var: The $\frac{1}{NM}\widetilde{\mathbf{W}}_j^2, j = 1, \ldots, J_0$ diagonal average energy matrices, each with elements (u, v).

```
R> str(modwtchs.p@vars.modwt$image$HH.var)


List of 5
 $ HH1: num [1:64, 1:64] 0 0 0 0 0 0 0 0 0 0 ...
 $ HH2: num [1:64, 1:64] 0 0 0 0 0 0 0 0 0 0 ...
 $ HH3: num [1:64, 1:64] 0 0 0 0 0 0 0 0 0 0 ...
 $ HH4: num [1:64, 1:64] 2.13e-06 4.73e-06 8.64e-06 1.39e-05 1.90e-05 ...
 $ HH5: num [1:64, 1:64] 2.22e-07 1.21e-06 7.50e-06 1.25e-05 1.48e-05 ...
```

LL.var: The $\frac{1}{NM}\widetilde{\mathbf{Z}}^2_{J_0}$ smooth average energy matrix with elements (u, v).

```
R> str(modwtchs.p@vars.modwt$image$LL.var)


 num [1:64, 1:64] 0.000764 0.00074 0.000703 0.000672 0.000646 ...
```

iso.var: Like the marginal vector of the same name, this matrix is the sum over all of the directional component matrices and includes the smooth matrix at the final level...

$$\widetilde{\mathcal{I}}_j = \begin{cases} \frac{1}{NM}\left(\widetilde{\mathbf{V}}^2_j + \widetilde{\mathbf{U}}^2_j + \widetilde{\mathbf{W}}^2_j\right) & \text{for } j = 1, \ldots, J_0 - 1, \quad J_0 > 1 \\ \frac{1}{NM}\left(\widetilde{\mathbf{V}}^2_{J_0} + \widetilde{\mathbf{U}}^2_{J_0} + \widetilde{\mathbf{W}}^2_{J_0} + \widetilde{\mathbf{Z}}^2_{J_0}\right) & \text{for } j = J_0 \end{cases} \tag{26}$$

```
R> str(modwtchs.p@vars.modwt$image$iso.var)


List of 5
 $ iso1: num [1:64, 1:64] 0 0 0 0 0 0 0 0 0 0 ...
 $ iso2: num [1:64, 1:64] 0 0 0 0 0 0 0 0 0 0 ...
 $ iso3: num [1:64, 1:64] 0 0 0 0 0 0 0 0 0 0 ...
 $ iso4: num [1:64, 1:64] 6.39e-06 1.42e-05 2.59e-05 4.17e-05 5.99e-05 ...
 $ iso5: num [1:64, 1:64] 0.00115 0.00118 0.0012 0.00119 0.00117 ...
```

isoSurf.var: This matrix is the sum over all of the individual J_0 level matrices in the previous `iso.var` slot; i.e., $\widetilde{\mathcal{I}}_1 + \ldots + \widetilde{\mathcal{I}}_{J_0}$. Below we see that the sum over all (u, v) cells in `isoSurf.var` is just another sample estimate of $\mathrm{E}[X^2]$, the total average energy, to be compared with that given by the sampling surface itself...

```
R> str(modwtchs.p@vars.modwt$image$isoSurf.var)


 num [1:64, 1:64] 0.00115 0.00119 0.00122 0.00123 0.00123 ...


R> NM = ncell(modwtchs.p@ss@tract)                    #total image size
R> AE = sum(matrix(modwtchs.p@ss@tract)^2)/(NM)       #sampSurf total AE
R> SV = sum(modwtchs.p@vars.modwt$image$isoSurf.var) #MODWT total AE
R> AE  - SV


[1] 1.8474111e-13
```

It is important to note that even though the total average energies in the original sampling surface and the MODWT wavelet decomposition sum to exactly the same value, their image representations are different on a per cell basis. That is, while the MODWT decomposition preserves total energy, it does not preserve the individual (u, v) energies on a per cell basis. Hence, the MODWT average energy surface will be different from the `sampSurf` surface, as would be expected since the former is composed of the sum of the scale-based decompositions, while the latter is essentially the average-square of the original sampling surface values as shown above.

An example contrasting the `sampSurf` and cumulative (`isoSurf.var`) MODWT variance surfaces at level J_0 (the only level we can compare them at) is presented in Figure 2. The figure illustrates quite plainly the difference between the two surfaces. Level $j = 1$ inclusion zone boundaries can be seen exaggerated like moon craters in several places. The MODWT surface is spread out, with a smaller maximum than the `sampSurf` version, due to the different combined scale smoothing effects in the image. The `ssEnergy` function computes the average energy from a `sampSurf` scene in the corresponding slot of the "ssMODWT" object.[11]

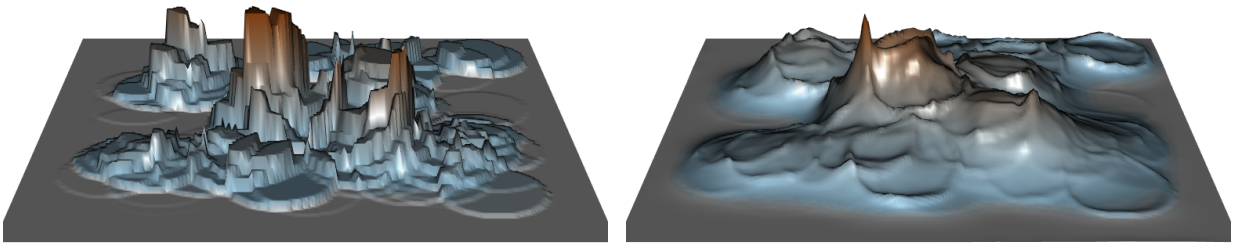
```
R> ae = ssEnergy(mo.bef, showPlot = FALSE)     #sampSurf AE
R> (mo.bef.range = range(matrix(ae)))


[1]    0.0000 1014.6854


R> mo.pal = palMODWT(100, range = mo.bef.range)  #a nice palette
R> plot3D(ae, col = mo.pal)
```

```
R> (mo.bef.range = range(mo.bef@vars.modwt$image$isoSurf.var))


[1]   0.00000 571.91254


R> M = nrow(mo.bef@ss@tract)
R> N = ncol(mo.bef@ss@tract)
R> plot3D(raster(mo.bef@vars.modwt$image$isoSurf.var, 0,M, 0,N), col = mo.pal)
```

4.4 Decomposition levels

The levels used in the decomposition are given in the `levels` slot and correspond to the definitions in § 1.1...

[11] Please see `?ssEnergy` for more information on this function.



(a) sampSurf (b) MODWT

Figure 2: An `sampSurf` (left) and MODWT (right) isotropic average energy surface variance decomposition for the BEF example in § 3.2.

```
R> modwtchs.p@levels


$J
[1] 5

$N
[1] 64

$M
[1] 64

$L
[1] 2

$j
[1] 1 2 3 4 5

$tau.j
[1]  1  2  4  8 16

$Lj
```

```
[1]  2  4  8 16 32

$Nj
[1] 63 61 57 49 33

$Mj
[1] 63 61 57 49 33
```

Note in this example that we have $N = M = 64$ which, if we were doing a DWT diadic decomposition would make $N = M = 2^J = 2^6$. But here, we have specified $J = 5$, which is really $J_0 < J$ in our above notation. In essence, J has no meaning in MODWT, though as noted above, we are treating $J \equiv J_0$ in the R code and elsewhere for MODWT.

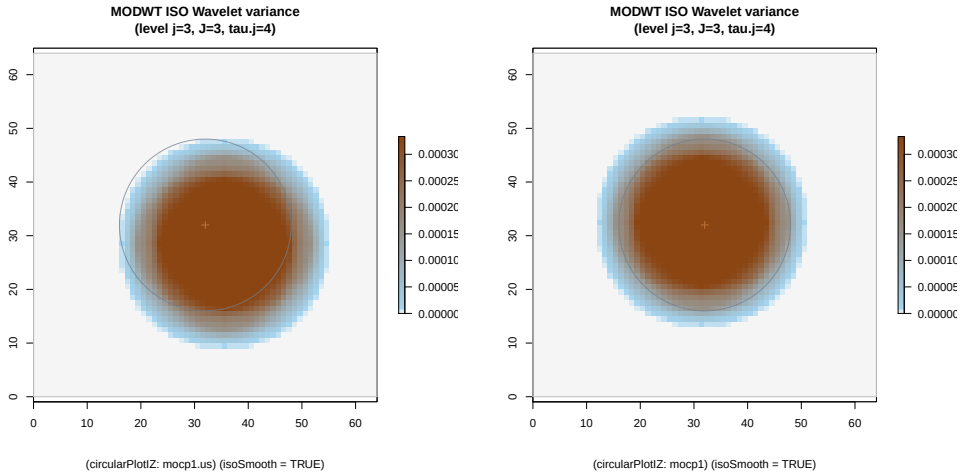
4.5 MODWT versus MODWT+MRA

As noted above (§ 3), variances can only be calculated for the MODWT decomposition, not the accompanying MRA. However, the MRA provides a multi-resolution analysis that is spatially rectified to the original image, which is necessary for conducting any further analysis where juxtaposition and alignment actually matter (just about everything). The normal result returned from a MODWT analysis gets shifted progressively further away from the correct alignment, the higher the decomposition level; i.e., the misalignment is cumulative.

```
R> mocp1.us = ssMODWT(sscp1, J = J.cp, shift = FALSE, runQuiet = TRUE)
R> plotMODWT2D(mocp1.us, waveType = 'ISO', type = 'var', level = J.cp,
+              decompType = 'modwt')
R> plotMODWT2D(mocp1, waveType = 'ISO', type = 'var', level = J.cp,
+              decompType = 'modwt')
```

The shift for a MODWT analysis is illustrated in Figure 3, where the level $J = 3$ decomposition for a MODWT is shown both unshifted and shifted. Note that the wavelet coefficients do not align correctly with the inclusion zone (they appear shifted to the right and down) in the unshifted MODWT image, but they align perfectly after shifting. A MODWT+MRA decompostion would appear very similar to the shifted version; however, the scales would be slightly different between the two images, reflecting the difference in the decompositions.[12]

[12] Simply change `decompType='mra'` to plot the MODWT+MRA analysis.



(a) Unshifted (b) Shifted

Figure 3: An unshifted (left) MODWT isotropic variance decomposition at level $J = 3$ on the single-tree circular plot sampling surface. Note that the decomposition is offset in the positive x and negative y directions from the actual inclusion zone (solid line) compared to the rectified (shifted) version (right).

4.6 The "ssCovMODWT" Class

A simple extension to the "ssMODWT" class is used to create a subclass that will allow wavelet covariance estimation between two "sampSurf" objects (see § 7.2 for an example). The "ssCovMODWT" class is defined as...

```
R> showClass('ssCovMODWT')


Class "ssCovMODWT" [package "ssWavelets"]

Slots:

Name:         ss.b  ss.modwt.b    ss.mra.b    covStats    ss.modwt  vars.modwt
Class:    sampSurf        list        list        list        list        list

Name:       ss.mra      levels description      wfName          ss
Class:        list        list   character   character    sampSurf

Extends:
Class "ssMODWT", directly
Class "ssWavelet", by class "ssMODWT", distance 2
```

In addition to the slots in "ssMODWT", this subclass adds the following slots...

ss.b: The second "sampSurf" object.

ss.modwt.b: The MODWT analysis corresponding to `ss.b`.

ss.mra.b: The MODWT+MRA analysis corresponding to `ss.b`.

covStats: A simple summary of the sampling surface and wavelet decomposition covariances and correlations; note that the respective entries should match...

- *ssCov:* The covariance for the two sampling surfaces.
- *ssCor:* The correlation for the two sampling surfaces.
- *modwtCov:* The overall covariance for the MODWT analysis.
- *modwtCor:* The overall correlation for the MODWT analysis.

One very important point to keep in mind with regard to the "ssCovMODWT" object is that the `vars.modwt` slot now holds a *covariance* analysis, not a *variance* analysis as in the simpler "ssMODWT" objects. The individual wavelet variance analyses are *not* directly available in the "ssCovMODWT" class object; therefore, it is necessary to save the "ssMODWT" objects that were used to create the covariance object if individual variance results are required in further analysis.

5 Plotting

The `raster` package is used heavily in `sampSurf`. It is also extremely useful when displaying the results from a `modwt.2d` wavelet analysis. It is a simple matter to cast the resultant matrices into a "raster" class object and take advantage of the plotting and summary options built into `raster`, both two- and three-dimensional. Examples of figures that can be generated for different decomposition levels are illustrated in the following. These all take advantage of the `raster` facilities for display.

5.1 `plotMODWT2D`: individual components within level

The most basic plotting routine is `plotMODWT2D`, it generates a display of the jth level decompostion (MODWT or MODWT+MRA). However, it will only display one component at a time; i.e., LHj, HLj, HHj, LLj, or ISOj, where the latter is the isotropic display. An illustration of this is displayed in Figure 3.

This routine also supports a `Lattice` plot of an "ssMODWT" object. An example is given in Figure 4 from the following...

```
R> rl = plotMODWT2D(mocp2, waveType = 'ISO', type = 'var', level = 0,
+              decompType = 'modwt', addLattice = TRUE, showPlot = FALSE)
R> plot(rl$plt)
```

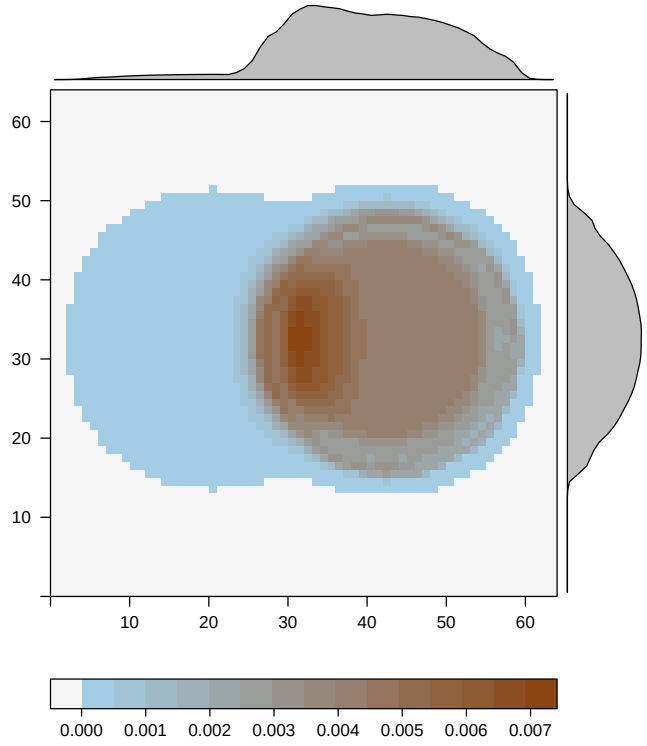


Figure 4: A lattice version of an isotropic MODWT decomposition over all levels (`isoSurf.var`: § 4.3.3.

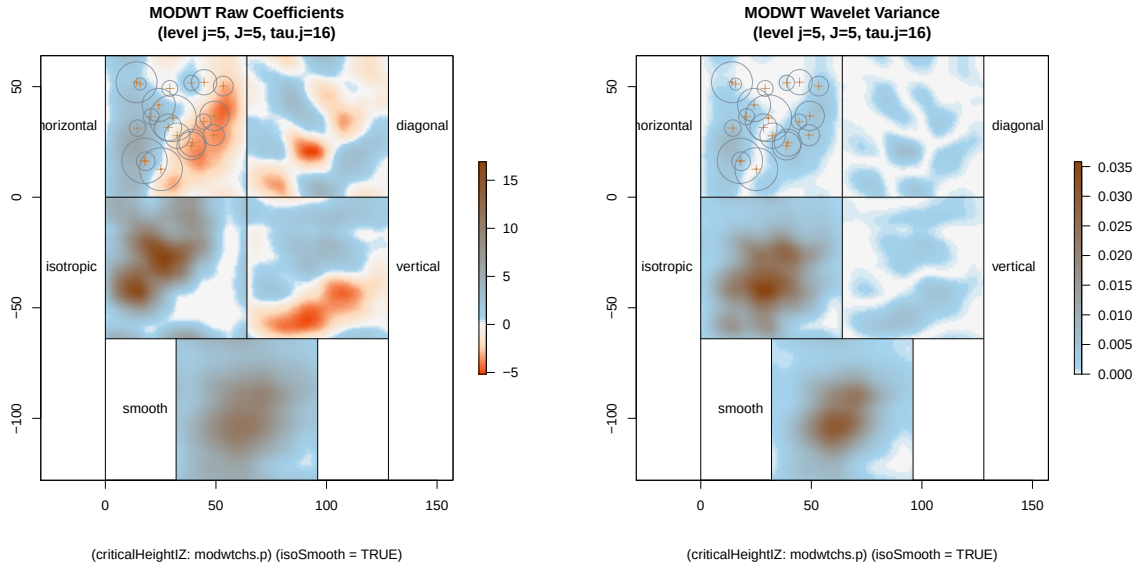
Note that the return object is a list with the Lattice plot as one of the components that can simply be plotted. This option is not available for `plotLevel2D` in § 5.2.

5.2 `plotLevel2D`: all components within level

A perhaps more interesting display is produced by `plotLevel2D`, which calls `plotMODWT2D` to get the individual component plots, and then combines them into one set for the jth level. The one drawback to this plot can be that the scale of the isotropic component can be much greater than that of the other components, and thus sometimes make it difficult to see details on the anisotropic figures. This occasionally can be remedied by using a different color scheme, but more often the

scale differences are too large. The inclusion zones, if desired, are displayed only the upper left (horizontal) component. An illustration of this method is displayed in Figure 5.

```
R> plotLevel2D(modwtchs.p, type = 'var', level = J.chs, decompType = 'modwt')


Top left: horizontal
Top right: diagonal
Bottom right: vertical
Botton left: isotropic
Bottom middle: final smooth


R> plotLevel2D(modwtchs.p, type = 'raw', level = J.chs, decompType = 'modwt',
+              runQuiet = TRUE)
```



(a) Raw (b) Variance

Figure 5: A MODWT raw (left) and variance (right) decomposition at level $j = 5$ on the CHS sampling surface (LH=upper left, HH=upper right, HL=lower right, ISO=lower left, LL=bottom).

Notice the difference in scale between the two sets of images in Figure 5. Even though the energy is the square of the raw image coefficients (which can be large), recall that the average energy, and thus the variance, is divided by $N \times M$, and therefore produces small values for the variance when displayed on a per-pixel basis.

6 Boundaries

As in forest sampling, there are potential pitfalls with edges of images because as the wavelet is dilated to larger scales part of a wavelet can extend beyond the boundary when the location nears the edge of the scene. Addison (2002, §2.15) has the most thorough discussion on this subject of the books I have consulted so far, the result of which he calls a *boundary effect*, as we do in sampling. He lists several potential correction mechanisms with regard to time series or signals, but most of these also readily extend to images; e.g.,...

1. *Zero padding:* Pad around the image with zeros (background values in `sampSurf`). This is automatic with "bufferedTract" boundary correction in `sampSurf`.
2. *Value padding:* Pad with some positive value.
3. *Decay padding:* Assume a decaying signal (surface) past the edges.
4. *Periodization:* Also known as "periodic boundary correction" (PBC: `https://en.wikipedia.org/wiki/Periodic_boundary_conditions`). This is toroidal correction (Ripley, 1981, p. 152 & 166). It is a translation of the image such that the end wraps back to the beginning. This is an *immutable* "option" in the R `waveslim` package without changing the code. In fact, one can not choose no correction, a look at the R & C code shows that this is hard-wired in the C `dwt` and `modwt` routines (both found in `dwt.c`). It is also hardwired into the `mra` and `mra.2d` code, which calls these functions.
5. *Reflection:* The image is reflected in all directions. That is, it is folded about each edge to make a mirror image on each side (Lark and Webster, 2004, Figure 4).

   1D: Reflection *is* an option in `waveslim`, but *only* for 1D DWT, MODWT and MRA analysis. If it is chosen the series is padded *only* on the end with the reflected values. In fact, the R code is the following (see any of `dwt`, `modwt` or `mra`), with `x` as the time series passed...

   ```
   switch(boundary,
          ``reflection''  = x  <- c(x, rev(x)),
          ``periodic''  = invisible(),
          stop(``Invalid boundary rule in dwt''))
   ```

   If one specifies `''periodic''`, then nothing is done, because this is the unchangeable default. If you specify `''reflection''`, then the series is reflected *on the upper end only*, and periodic correction is applied to this reflected series. Note that the 1D MRA routine, `mra`, also allows reflection (and, as above, *always* applies periodization since it simply calls the above routines first for the general decomposition)—the code is the same as above.

   2D: However, the 2D routines (`dwt.2d`, `modwt.2d` or `mra.2d`), have no facility for reflection, they always simply apply periodization. The code looks like the following...

```
switch(boundary,
       ``periodic''  = invisible(),
       stop(``Invalid boundary rule in mra''))
```

It is difficult to know whether this is by design or oversight? It is probably by design as the series would require reflecting for both rows and columns, and would, I think, also need the diagonals (see the example from `ssReflect` below).

6. *Windowing:* It is not entirely clear at present how this is used for boundary correction.
7. *Polynomial fitting:* Use a fitted polynomial to extend in all directions.
8. *Signal following:* If we have a larger signal (area) surrounding the area of interest, then we can simply use the extended area. We call this *buffering* in the case of, e.g., growth plots where an external buffer is maintained surrounding the plot.
9. *Wraparound:* Rather than wrapping the signal/image, this wraps the wavelet and seems to be favored by Addison (2002, p. 59).

The use of padding and periodization was discouraged by Lark and Webster (2004). In discussing the latter they note...

> *Again this is clearly inappropriate. Discontinuities may appear at the boundary, and the benefit of localization in wavelet analysis is lost if we generate coefficients near one boundary using values from the opposite boundary. Standard implementations of the pyramid algorithm use this solution.*

These authors advocated reflection as the most appropriate. It is true that in some sense localization is lost, however, as we note below, a more insidious problem can arise with reflection. Also, as noted above, the `waveslim` package uses periodization exclusively. Even if one were to reflect the image into a larger area, the `modwt.2d` routine will still used periodization on this image; more on this below. These authors note, with respect to reflection...

> *This creates discontinuities in the derivatives of the data, but it is unlikely to lead to misleading results as long as the scale parameters are restricted to values less than or equal to half the shortest dimension of the sampling grid.*

Ma and Zhang (2015) also eschew padding and periodization as "not appropriate for our study." They chose reflection stating that...

> *This approach may effectively reduce the boundary effect if we control the values of scale parameters less than or equal to half of the shortest dimension of original data.*

However, these authors used the `waveslim` package `modwt.2d` routine in their analysis with no mention of modification. Thus, it is possible that they were unaware that it automatically applied periodization to the enlarge reflected image.

In a paper explicitly devoted to two-dimensional wavelet variance decomposition, Geilhufe et al. (2013) make no mention of boundary correction methods applied in their analysis.

Reflection seems to be the correction method of choice. However, there is one salient point with regard to tree populations that may negate this as a preferred method. In a spatial inhibition process, as would be used to generate most tree populations—and indeed would be found in the woods—generating trees near the boundary has preference because the boundary has no other individuals to inhibit placing a tree there, unless there is one close by within the tract. Thus, there is the potential for there to be more trees right near the edge. If the image is folded/reflected, this will result in the same tree being replicated very close to itself (perhaps like a forked tree in some extreme cases). This may or may not be acceptable. This does not appear to be an issue in the Ma and Zhang and Lark and Webster studies as their phenomenon under study were not so constrained. An example of a reflection applied to a sampling surface image is shown in Figure 6.

```
R> refchs = ssReflect(sschs)
R> plot(refchs, col = palMODWT(100, range = cellStats(refchs, range)))
R> plot(perimeter(sschs), add = TRUE)
```

The potential inhibition-related problem would not be of much concern in the case where we generated a sampling surface using a buffered tract to contain all of the inclusion zones, since then the trees themselves would lie within the internal buffer region. However, if we used mirage correction (or walkthrough when it gets implemented), trees can lie right next to the boundary of the tract. Also, coarse woody debris can cross the boundary and be clipped, and when reflected create a mirror image of the log in question. This is not an inhibition problem, but would create a type of discontinuity in the surface at the boundary due to the mirroring of the log.

The arguments put forth in the above studies that periodization is "clearly inappropriate" seem reasonable for those studies, but as noted, may not be as applicable in the case of sampling surfaces, especially using buffered tracts. Periodization is the toriodal correction as noted above and this has been advocated by many authors like Ripley for boundary correction in spatial analysis. And it does not suffer from the potential inihbition problem discussed above, unless a tree on one edge is aligned with one on the opposite edge when translating, and this seems to have less of a potential for happening than under reflection. The fact that it is in `modwt.2d` and also used in other branches makes this appealing, but as noted, it will be applied regardless if this routine is used for the analysis. An idealized example of a periodization as applied to the same sampling surface image presented in Figure 6 is shown in Figure 7.

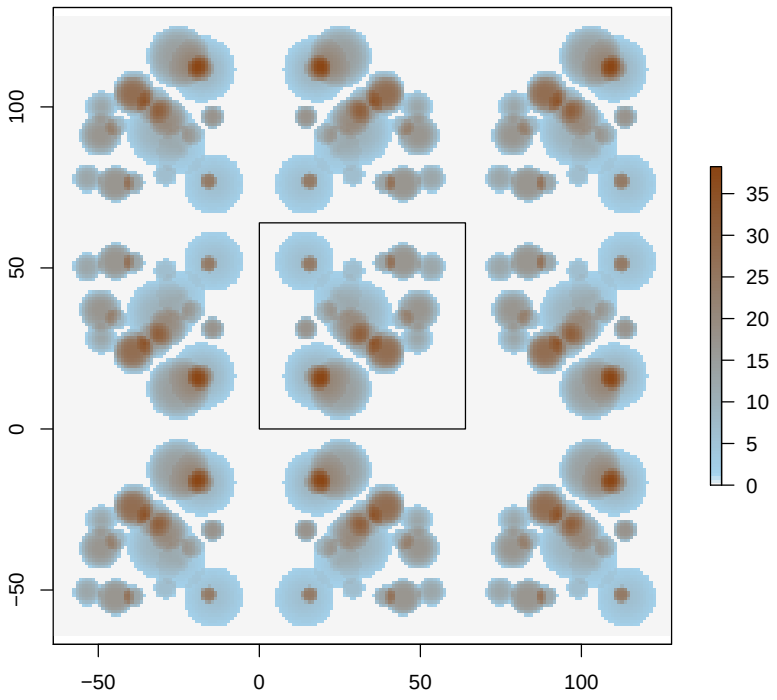


Figure 6: Reflection of the critical height sampling surface (middle) about all sides.

```
R> torchs = ssToroid(sschs)
R> plot(torchs, col = palMODWT(100, range = cellStats(torchs, range)))
R> plot(perimeter(sschs), add = TRUE)
```

The example above is conceptually correct, but not algorithmically what is used in `modwt.2d` via the pyramid algorithm. There, rows and columns are filtered one-by-one as if they were individual time series, using the appropriate **C** routine for univariate time series filtering under MODWT in `waveslim`. This is done in the correct order to generate the resulting matrices in Table 1. This same routine is where periodization is automatically applied.

6.1 Reflection Plus Periodization

Gençay et al. (2002, p. 144) suggest that using both reflection and periodization (R+P) is a so-

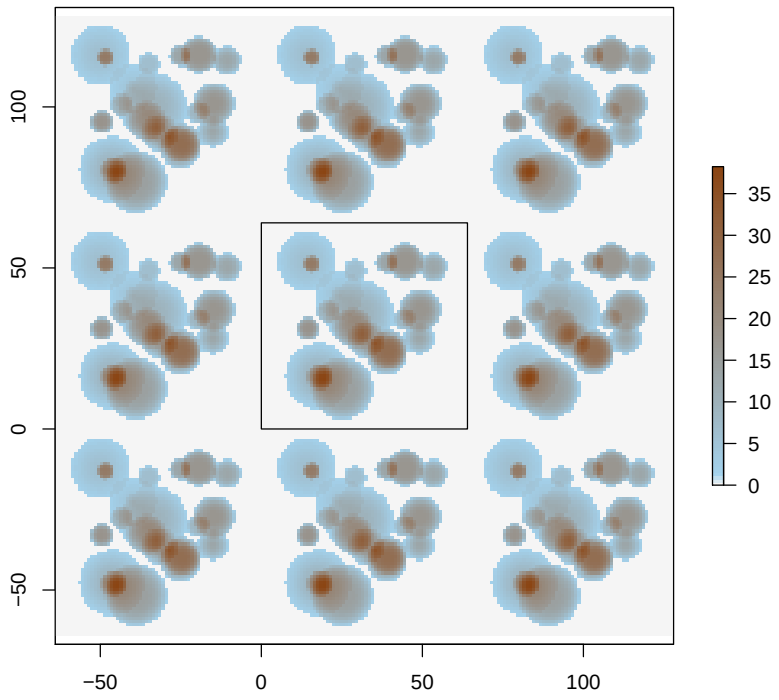


Figure 7: Periodization of the critical height sampling surface (middle) about all sides.

lution that has been used in Fourier analysis of signals (note this book is about time series, not images). The `ssMODWT` constructor allows this option by passing `reflect = TRUE` in the call. The MODWT decomposition is then applied to the entire image as in Figure 6 using periodization at the boundaries of the extended image. Then the results of the analysis are clipped back to the original tract extents at each level. However, this seems to cause wavelet statistics not to equate correctly with the sampling surface statistics at higher levels. The following example compares the previous MODWT decomposition, which used only periodization for boundary correction, to one that uses periodization on a reflected sampling surface "image."

```
R> modwtchs.rp = ssMODWT(sschs, reflect = TRUE, J = J.chs)


Reflection + Periodic
Inclusion zone class: criticalHeightIZ
Surface variance = 52.440687
Wavelet variance = 52.540737 (MODWT)
```

```
Surf/Wave var ratio = 0.99809577
Check: surface var matrix = 52.541155 = E[X^2] - E[X]^2
Surface mean = 4.6024029
Wavelet mean = 4.6024483 (MODWT)
Wavelet mean = 4.6024029 (MRA)


R> sum(sapply(modwtchs.p@ss.modwt, sum)[1:(3*J.chs)]) #all but LLJ--should be zero


[1] -4.1530256e-14


R> sum(sapply(modwtchs.p@ss.mra, sum)[1:(3*J.chs)]) #all but LLJ--should be zero


[1] 1.94e-06


R> sum(sapply(modwtchs.rp@ss.modwt, sum)[1:(3*J.chs)]) #all but LLJ--should be zero


[1] 178.14795


R> sum(sapply(modwtchs.rp@ss.mra, sum)[1:(3*J.chs)]) #all but LLJ--should be zero


[1] -4.96e-06
```

Note in the above that the sums for the R+P correction are positive, not zero as they should be. Further analysis (available from the author) shows that the problem comes at the higher levels of decomposition, where the the sums are non-zero.

Elsewhere it has been demonstrated that it is the act of shifting and subsequent clipping out of the internal image corresponding to the original tract that creates the bias in the MODWT decomposition.[13] The main culprit appears to be the act of shifting using `waveslim::shift.2d`, which is applied to rectify the full image *prior* to clipping. This follows because the MODWT+MRA analysis, which is also applied to the full augmented image *before* clipping, produces a zero bias; however, it preserves additivity over sub-images as shown in the summary results above. Unfortunately, shifting is necessary for re-alignment in MODWT, which is required for variance decomposition, so we can not take advantage of the MODWT+MRA result here. Any analysis based on the raw wavelets,

[13] Extended analysis available from the author on request.

however, would work fine with regard to the MODWT+MRA decomposition and that would be the correct one to use in this case.

In summary, even though R+P is included in the `ssMODWT` generator function, it should be used with caution when the results might be applied to variance estimates. However, it should be fine for MODWT+MRA based on the raw wavelet decomposition.

7 Comparing Methods

In this section, we look at a few different ways that wavelets might give a little new insight into comparing sampling methods.

7.1 Sample variance

The derivation (sort of) of the sample variance with regard to wavelet decomposition was outlined in § 2.2. These same methods can be used to look at the wavelet variances as well and are mentioned where appropriate. Most of this discussion will center around summaries given through the plotting routines.

7.1.1 Image variance plots

The image plotting routines were demonstrated in § 5. Figures 3 and 5 show wavelet variance results for single component plots at level j, and the set of directional and isotropic component plots at the jth level, respectively. The image variances are interesting, but often, due to the nature of the phenomenon of interest (areal sampling surfaces), may not be as informative for comparing methods as the marginal displays demonstrated in the following section.

7.1.2 Marginal variance plots

The 'marginal' sample and wavelet variance plots are often more useful. They are marginal in the sense that they display the sum of the variances over all sample points (cells) in individual images at levels $j = 1, \ldots, J_0$. Unlike the image plots, these can be displayed only for the isotropic variance components at the present time[14]. The following `sampSurf` and associated MODWT results are used to compare circular plot sampling and horizontal point sampling on the simple three-tree population.

[14] Though this could easily be changed if desired.

```
R> #circularPlotIZ (default) plot radius is in meters...
R> sscp.8 = makeSS('ss.3tree', plotRadius = 8, runQuiet = TRUE)
R> sscp.10 = makeSS('ss.3tree', plotRadius = 10, runQuiet = TRUE)
R> sscp.12 = makeSS('ss.3tree', plotRadius = 12, runQuiet = TRUE)
R> hfs.J = 4
R> mocp.8 = ssMODWT(sscp.8, J = hfs.J)


Inclusion zone class: circularPlotIZ
Surface variance = 74.459497
Wavelet variance = 74.441318 (MODWT)
Surf/Wave var ratio = 1.0002442
Check: surface var matrix = 74.441318 = E[X^2] - E[X]^2
Surface mean = 2.761457
Wavelet mean = 2.761457 (MODWT)
Wavelet mean = 2.761457 (MRA)


R> mocp.10 = ssMODWT(sscp.10, J = hfs.J, runQuiet = TRUE)
R> mocp.12 = ssMODWT(sscp.12, J = hfs.J, runQuiet = TRUE)
```

```
R> #angleGauge object default is metric baf...
R> sshps.3 = makeSS('ss.3tree', 'horizontalPointIZ', angleGauge=angleGauge(3),
+                   runQuiet = TRUE)
R> sshps.4 = makeSS('ss.3tree', 'horizontalPointIZ', angleGauge=angleGauge(4),
+                   runQuiet = TRUE)
R> sshps.5 = makeSS('ss.3tree', 'horizontalPointIZ', angleGauge=angleGauge(5),
+                   runQuiet = TRUE)
R> mohps.3 = ssMODWT(sshps.3, J = hfs.J)


Inclusion zone class: horizontalPointIZ
Surface variance = 45.642074
Wavelet variance = 45.630931 (MODWT)
Surf/Wave var ratio = 1.0002442
Check: surface var matrix = 45.630931 = E[X^2] - E[X]^2
Surface mean = 2.7776477
Wavelet mean = 2.7776477 (MODWT)
Wavelet mean = 2.7776477 (MRA)


R> mohps.4 = ssMODWT(sshps.4, J = hfs.J, runQuiet = TRUE)
R> mohps.5 = ssMODWT(sshps.5, J = hfs.J, runQuiet = TRUE)
```

The `hfsMODWT` method combines any number of objects generated by `ssMODWT` into a data frame in either *(i)* wide, or *(ii)* long format. For example, the results for horizontal point sampling from the above objects in long format are...

```
R> args(hfsMODWT)


function (..., ids = NA, long = TRUE, runQuiet = TRUE)
NULL


R> cp.hfs = hfsMODWT(mocp.8, mocp.10, mocp.12, ids=c('cp8','cp10','cp12'),
+                    long = TRUE)
R> hps.hfs = hfsMODWT(mohps.3, mohps.4, mohps.5, ids = c('baf3','baf4','baf5'),
+                     long = TRUE)
R> head(hps.hfs)


    id      mean  scaleVar   waveVar waveSampVar     avg.E    izArea iso.j
1 baf5 2.7236325 31.044980 36.953536   67.998515 75.416689 151.84364  iso1
2 baf4 2.7796883 29.313877 28.085211   57.399087 65.125754 189.80456  iso1
3 baf3 2.7776477 25.337202 20.293729   45.630931 53.346258 253.07274  iso1
4 baf5 2.7236325 31.044980 36.953536   67.998515 75.416689 151.84364  iso2
5 baf4 2.7796883 29.313877 28.085211   57.399087 65.125754 189.80456  iso2
6 baf3 2.7776477 25.337202 20.293729   45.630931 53.346258 253.07274  iso2
    iso.var tau.j j
1 4.9260724     1 1
2 3.3381179     1 1
3 2.2378952     1 1
4 6.1527443     2 2
5 4.3362901     2 2
6 2.9489208     2 2
```

Long format: Long format is set up for conditioning plots using `lattice` or similar. In the example here we see several columns of information that are contained in the data frame as follows...

`id`: A simple sampling method identifier that will be used in the legend to identify the methods. This defaults to the object names passed.

`mean`: The sampling surface mean.

`scaleVar`: The empirical scaling variance, which is given by $\widehat{\sigma}^2_{Z_{J_0}}$ (21).

`waveVar`: The total empirical wavelet variance over all levels as given by $\widehat{\sigma}^2_{WUV}$ (20).

`waveSampVar`: The wavelet sample variance, $\widehat{\sigma}^2_X$, which is equal to the sampling surface variance given by `sampSurf`, as demonstrated in § 4.3.2.

`avg.E`: This is the average energy, $\widehat{\mathcal{E}}$ (22), in the marginal decomposition for each sampling method passed.

`izArea`: The *average* inclusion zone area for variable radius methods. Of course, this is simply the plot size for fixed-area methods.

`iso.j`: This is the isotropic identifier at level j that corresponds to the column names in wide format giving the isotropic variances below.

`iso.var`: The isotropic variances as described in § 4.3.1. These are indexed by `id` and `j` above, and corresponding to the `iso.j` columns in wide format.

`tau.j`: The τ_j as defined in § 1.1; i.e., distance at level j.

`j`: The decomposition level j as defined in § 1.1

The sample variance plot is generated using `varPlot` from the data frames created in the calls to `hfsMODWT`. The variance actually plotted is the `iso.var` described above...

```
R> args(varPlot)


function (..., 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)
NULL


R> varplt = varPlot(cp.hfs, hps.hfs, sampMeth = c('cp','hps'), groupSM = TRUE,
+                     scales = 'free', theme = 'ggplot')


units = metric
isoSmooth = TRUE
```

The number of sampling methods that can be compared is essentially unlimited and can simply be added to the `'...'` argument list; one such data set is all that is required. The `sampMeth` argument provides the identifiers for the conditioning panels corresponding to the data frames. Any argument following the `'...'` must be specifically named or the routine will interpret them as members of the `'...'` argument list and throw an error. The plot generated above is shown in Figure 8.

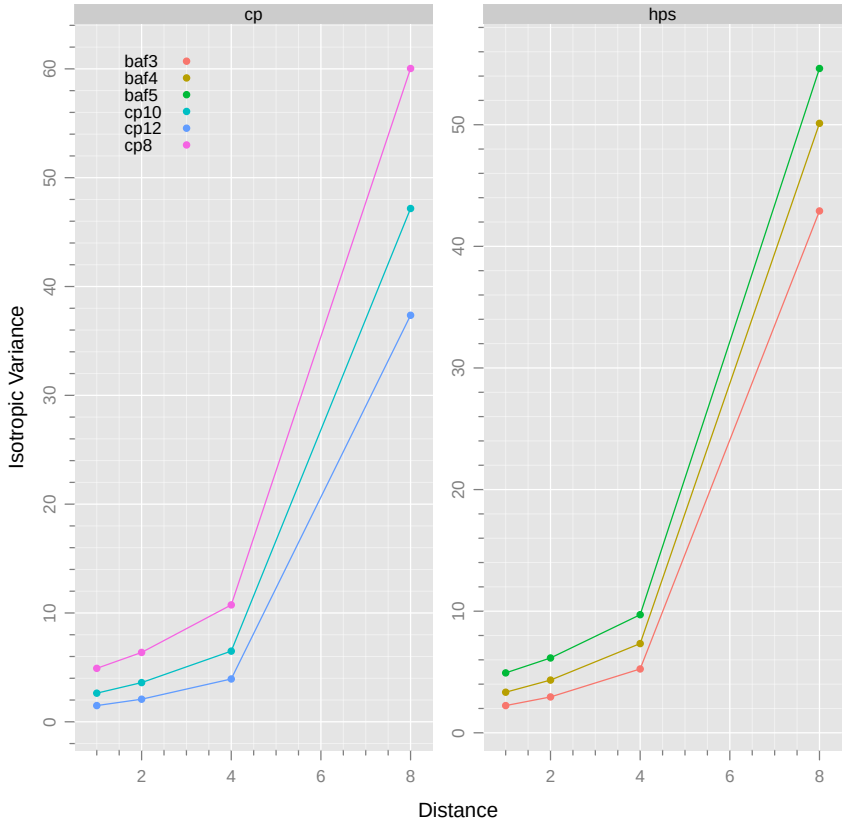


Figure 8: Marginal isotropic sample variance plot comparing circular plot and horizontal point sampling for volume on a 3-tree population.

Wide format: While long format is most useful for conditioning plots, `hfsMODWT` also can produce a data frame in wide format. For example, the horizontal point sampling results in wide format are...

```
R> cp.hfsw = hfsMODWT(mocp.8, mocp.10, mocp.12, ids=c('cp8','cp10','cp12'),
+                     long = FALSE)
R> hps.hfsw = hfsMODWT(mohps.3, mohps.4, mohps.5, ids = c('baf3','baf4','baf5'),
+                     long = FALSE)
R> head(hps.hfsw)


          id      iso1      iso2      iso3      iso4      mean  scaleVar
mohps.5 baf5 4.9260724 6.1527443 9.7108995 54.626973 2.7236325 31.044980
mohps.4 baf4 3.3381179 4.3362901 7.3363250 50.115021 2.7796883 29.313877
mohps.3 baf3 2.2378952 2.9489208 5.2490808 42.910361 2.7776477 25.337202
```

```
          waveVar waveSampVar     avg.E    izArea
mohps.5 36.953536   67.998515 75.416689 151.84364
mohps.4 28.085211   57.399087 65.125754 189.80456
mohps.3 20.293729   45.630931 53.346258 253.07274
```

It is straightforward to match the results in wide format to those in long format described previously. For example, the `ISO`j columns in wide format correspond to the `iso.var` column at the jth level. They have simply been 'stacked' and indexed to create the long format version. An example where we can bind together wide format results is shown in § 8.

Other uses for `hfsMODWT` output are described in § 8 and **??**.

7.2 Sample covariance

In § 2.3 we briefly described the calculation of the wavelet covariance between two sampling methods represented on two images, $\mathbf{X}$ and $\mathbf{Y}$. The covariances themselves are calculated by passing the two `sampSurf` images to `covMODWT` as described there. However, the function `ssCovMODWT` wraps this into a simple interface that takes as arguments the two respective MODWT objects as returned from `ssMODWT`. It returns an object of class "ssCovMODWT" that is a subclass of "ssMODWT", and is detailed in § 4.6. Extensions include a second `sampSurf` object (the $\mathbf{Y}$ image) that is returned in the `ss.b` slot. Furthermore, the `vars.modwt` slot now contains the covariance results in exactly the same format as presented in § 4.3, where the `isCovariance` slot is now set to `TRUE`.

```
R> args(ssCovMODWT)


function (ssMODWT.a, ssMODWT.b, ...)
NULL
```

The function `exCovMODWT`[15] applies `ssCovMODWT` on our one-, two-, or three-tree sample populations from § 3.1. The examples that follow use circular plot sampling with different plot sizes for a volume sampling surface.

7.2.1 One-tree Example

The examples that follow use a 12 m and 16 m plot radius.

[15] Available from the author on request.

```
R> exCovMODWT('ss.1tree', level = 1, waveType = 'LH', runQuiet = TRUE)
```

```
Wavelet type = LH
Squared differences...
  Level 1 computed covariances = 0
```

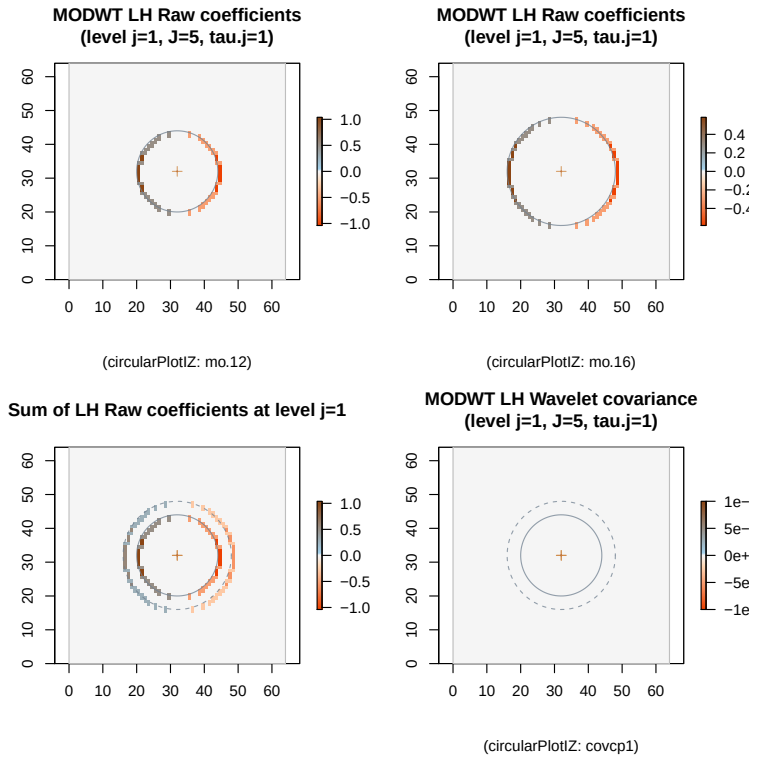


Figure 9: Horizontal (LH1) MODWT 12 m raw (top left), 16 m raw (top right), and raster sum of the two surfaces (bottom left) with the covariance surface decomposition at level $j = 1$ (bottom right) for the single-tree circular plot sampling surface.

First the above output from `exCovMODWT` demonstrates that the squared difference between a covariance (marginal sum over the image) calculated on the raster sum of the two methods (Figure 9 lower left) is equivalent to the covariance image (marginal sum) calculated in `ssCovMODWT` and displayed in the lower right. Note that since the raw wavelet coefficients do not overlap at any lattice point, the covariance is zero for both images.

By way of explanation, when the LH1 filter in (4) is applied to the 12 m surface we see the usual outline as in the top left figure, and similarly for the top right for the 16 m. It is important to note that all other grid cells are zero in the background. The raw coefficient (raster addition) composite of the two is in the bottom left to illustrate what is available for the product of the two images for

covariance calculation. Since they have no non-zero grid cells in the respective decompositions in common, the result is zero covariance when multiplied (as mentioned, zero surface values on one image multiplied by non-zero along the edges of the other). Note that the first three raw images are displayed simply to illustrate what is going on here in pictures, which makes it much simpler to understand than the final (zero-surface) covariance product alone.

Level $j = 2$ is very similar to that of the above, but with two double-ring approximately four-pixel wide ($\tau_2 = 2$ m) results, still non-overlapping (zero covariance).[16] Each level is quite interesting to look at and decipher what is going on with regard to the covariance surface at those distances. It is not until level $j = 3$ that we actually get some overlap because now $\tau_3 = 4$ m, which just "happens" to be the distance between the two plot radii, generating covariance in the two surfaces. Figure 10 presents the results of the following run for an isotropic pair of surfaces with the same sampling methods as in Figure 9, but at $j = 3$.

```
R> exCovMODWT('ss.1tree', level = 3, waveType = 'ISO', runQuiet = TRUE)
```

```
Wavelet type = ISO
Squared differences...
  Level 3 computed covariances = 7.5231638e-37
***>Careful: the product of 2 ISOs is not the same as the sum of the products
             of the individual anisotropic components if cross-product != 0<***
```

The same idea applies here as for the horizontal decomposition above. The background cells are all zero other than those shown by the isotropic raw wavelets for the two images, and the composite raster sum. The final covariance can be envisioned by imagining the product of the two surfaces in the top row as before.[17] It is interesting that the covariance surface is all positive in this case, simply because of the perfect separation at $j = 3$ and alignment of the two inclusion zones. In other not so contrived examples, this will not be the case, the covariance will be both positive and negative through the surface.

Two total average covariance images are shown in Figure 11. The leftmost image is the average covariance energy as calculated directly from the two individual `sampSurf` images using `ssEnergy`. The right image is the average covariance energy for the combined scenes; that is, the cumulative isotropic covariance over all levels, plus the level J_0 smooth as defined in 4.3.3, but for covariances, in the `image$isoSurf.var` slot in the object.

```
R> exCovMODWT('ss.1tree', level = 0, waveType = 'ISO', runQuiet = TRUE)
```

[16] Because of the circular edge nature of the inclusion zone, decomposition pixel width will vary for each image row.

[17] Again, the sum in the lower left just shows the overall coverage potential of the two images and can't be used to obtain the covariance.

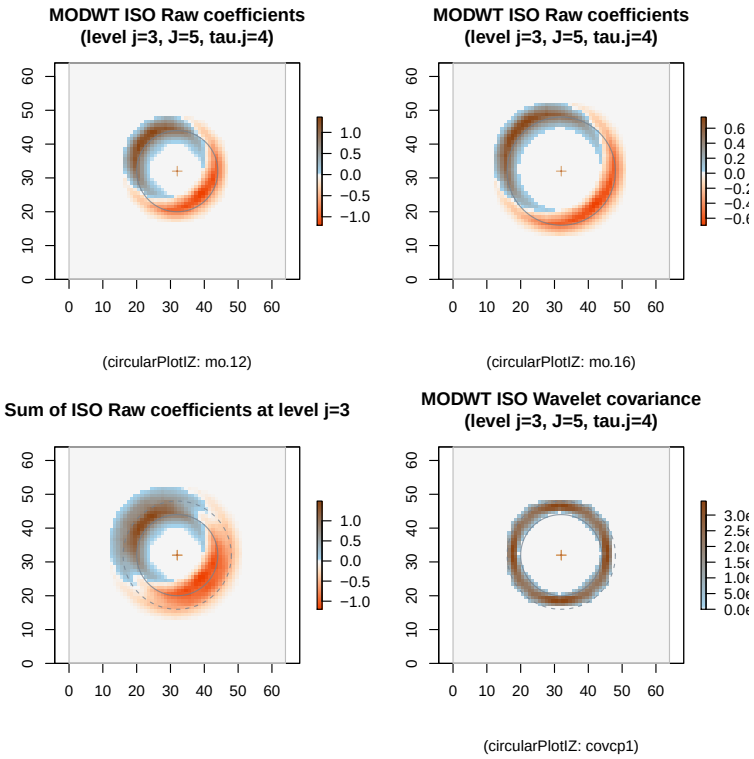


Figure 10: Isotropic (ISO3) MODWT 12 m raw (top left), 16 m raw (top right), and raster sum of the two surfaces (bottom left) with the covariance surface decomposition at level $j = 3$ (bottom right) for the single-tree circular plot sampling surface.

```
Wavelet type = ISO
Squared differences...
  Total average energies = 8.9055001e-31
Total "average energy" = 0.26565714
```

The `sampSurf` covariance energy image is, of course, what we would expect as the definitive image without wavelet decomposition. It is a positive-valued cylinder contained within the 12 m radius circular plot, with all other values outside the plot zero. Again, this derives from the fact that even though the surface cylinder associated with the 16 m plot has values throughout its inclusion zone, when those values outside the 12 m plot are multiplied by the 12 m surface, they become zero. Therefore, the covariance inside the 12 m plot is the product of the two sample surfaces within that circumference.

On the other hand, the wavelet covariance energy image accumulates all of the individual isotropic images at each scale plus the smooth. As such, it can be seen to spread the variance out over an area that not only contains the 16 m plot, but also many background values as well. Recall

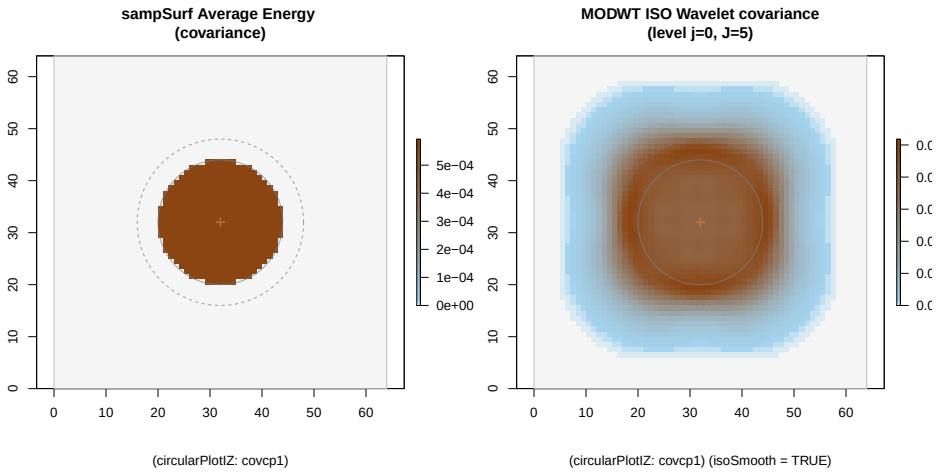


Figure 11: The average sampling surface covariance energy from the two sampling methods (left) and the isotropic surface MODWT total average covariance energy as given by the `image$isoSurf.var` image (right) for the single-tree circular plot sampling surfaces.

that $J_0 = 5$ implies that $\tau_5 = 16$ m in this example, so it has spread out in all directions by that amount due to the level $j = 5$ component of the decomposition. Thus, just as in the case of the total average energy for the variance, it is assigning covariance to cells where there is no original surface. The interpretation can conveniently be thought of as a covariance between sample points at these distances, since a sample point with zero estimate (surface height) is just as acceptable as one with an non-zero estimate.

Finally, in the results of the above run from `exCovMODWT` we see that the squared differences between the sum of the energies for each of the two images is zero. In other words, the sample covariance energy is preserved in the spatial decomposition surface since it exactly equals the population value. Thus, the spatial covariance decomposition of the average energy is different from the population image, but the sum of these, which is our estimate for $\mathrm{E}[X \cdot Y]$, is the same for both surfaces.

7.3 Resolution

All of the examples have been based on a resolution of $\Delta xy = 1$ m (ft) (§ 1.1). In theory, any resolution should work for wavelet decomposition just as it does in `sampSurf`. However, a simple example with a resolution of $\Delta xy = 0.5$ m demonstrates that there is a problem in `waveslim::shift.2d`.

```
R> sscp3.s = makeSS('ss.3tree', 'circularPlotIZ', plotRadius = 16, cellSize = 0.5)


Number of trees in collection = 3
```

```
Heaping tree: 1,2,3,


R> ssMODWT(sscp3.s, J = J.cp, shift = FALSE)


Inclusion zone class: circularPlotIZ
Surface variance = 20.949876
Wavelet variance = 20.948597 (MODWT)
Surf/Wave var ratio = 1.000061
Check: surface var matrix = 20.948597 = E[X^2] - E[X]^2
Surface mean = 2.7655902
Wavelet mean = 2.7655902 (MODWT)
Wavelet mean = 2.7655902 (MRA)


R> ssMODWT(sscp3.s, J = J.cp, shift = TRUE)


Warning in .local(ss, ...):
Resolutions other than 1 do not work with waveslim::shift.2d!


Inclusion zone class: circularPlotIZ
Surface variance = 20.949876
Wavelet variance = 10.346457 (MODWT)
Surf/Wave var ratio = 2.0248358
Check: surface var matrix = 4.7650647 = E[X^2] - E[X]^2
Surface mean = 2.7655902
Wavelet mean = 1.4377403 (MODWT)
Wavelet mean = 2.7655902 (MRA)
```

The first decomposition where no shifting (§ 4.5) is done is fine, the results are comparable both in variance and in means for `sampSurf`, MODWT, and MODWT+MRA. Unfortunately, this is not the case for the second example where shifting has been applied. We see that the `sampSurf` and MODWT+MRA means agree, but the MODWT mean is off by a factor less than two; given that the resolution is half the original in both directions, this odd factor suggests no simple clear correction. However, the fact that the MODWT+MRA results agree with those of `sampSurf`, and that all agree in the first example without shifting, pinpoints the problem. In addition, the variances are off by a factor greater than two (even with sample/population correction for the divisor, § 4.3.2) so that the two errors are not commensurate. This points quite clearly to a problem with `waveslim::shift.2d`.

Due to the above problem, I have added a check that throws a warning (see above) in the `ssMODWT`

constructor method if shifting is requested for `sampSurf` objects with a resolution different from $\Delta xy = 1$ m (ft). This is kind of a shame really because it restricts the full use of `sampSurf`, but in practice, it should not present much of a constraint.

8 Smith's "Law" Decomposed

The following is a simple example of how we could compare different sampling methods at different scales (distances) in the sense of 'plot size' using the Smith plot (Smith, 1938). In this example, three different plot sizes for circular plot sampling plus three different BAF's for horizontal point sampling are compared in the plot.

```
R> hfsplt = hfsPlot(cp.hfs, hps.hfs, sampMeth = c('cps','hps'), scales = 'free',
+                   conditionOn = 'tau.j', theme = 'ggplot')


units = metric
```

Note that the isotropic variance (`iso.var`) is used in the display in Figure 12. Therefore, the smooth component is included at level J_0. This inflates the y-scale—the variance—on this one panel and also changes the shape of the curves in the tail on this panel since the `LL.var` is different, not only for each sampling method, but for each plot size or BAF.

We can look at the overall `sampSurf` variance in a similar fashion as something to compare these against using the wide format results from `hfsMODWT`...

```
R> #reuse the wide format data frames from the variance plot section...
R> hps.hfsw$sm = 'hps'
R> cp.hfsw$sm = 'cps'
R> hfs = rbind(cp.hfsw, hps.hfsw)
R> xyplot(waveSampVar ~ izArea, hfs, groups=sm, type = 'b',
+        ylab = 'Total surface variance',
+        xlab = 'Average Inclusion Zone Area',
+        auto.key = list(x = .8, y = .9, corner = c(0, 0)),
+        par.settings = ggplot2like(),
+        lattice.options = ggplot2like.opts())
```

Figure 13 displays the results, showing that, at least for this very small population, the same trend in variance agreement is preserved with the full sample variance.

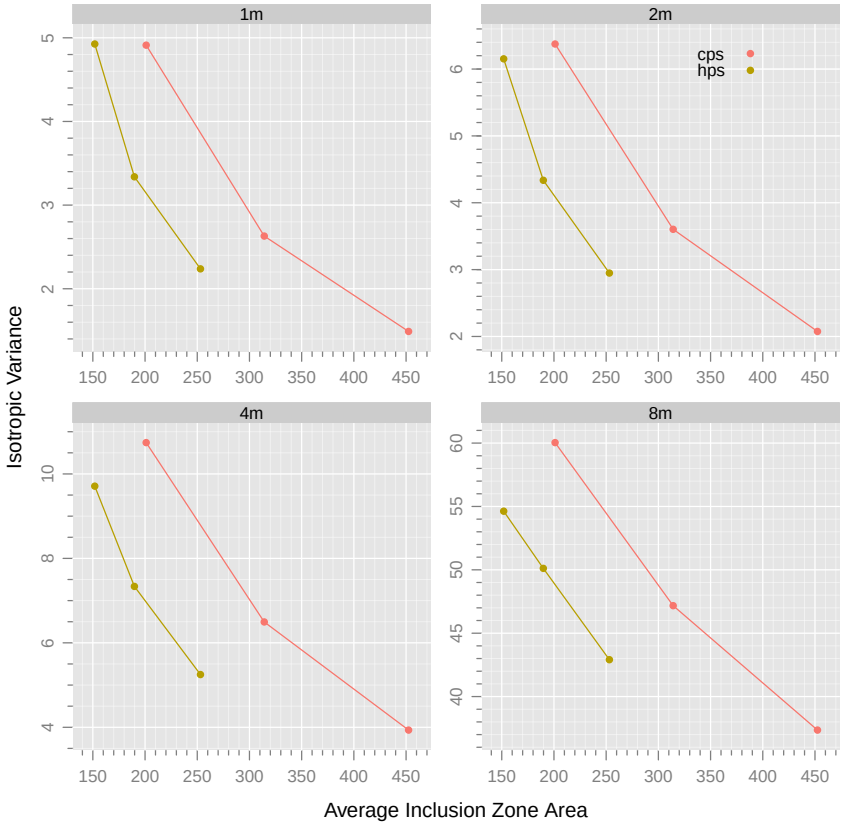


Figure 12: H. F. Smith plot comparing circular plot and horizontal point sampling by distance, τ_j, for volume on a 3-tree population.

8.1 Bartlett Example

The following is a slightly more involved example. It uses circular plot sampling (CPS), critical height sampling and horizontal point sampling on the BEF data (§ 3.2), each with different sampling efforts. For circular plot sampling, plot sizes of $^1/_{10}$th, $^1/_{20}$th, and $^1/_{50}$th acre are used. With both HPS and CHS, BAFs of 10, 20 and 30 are used. For reference, an example sampling surface for the BEF using circular $^1/_{50}$th acre (approximately 16ft radius) is shown in Figure 2.

The H. F. Smith plot comparison based on the full sampling surfaces for these methods is presented in Figure 14.[18]

```
R> bef.hfs = hfsBEF(J = 6, showPlot = FALSE, runQuiet = TRUE)
R> plot(bef.hfs$plt.pop)
```

[18]The `hfsBEF` function is available from the author on request—it simply facilitates generating this example.

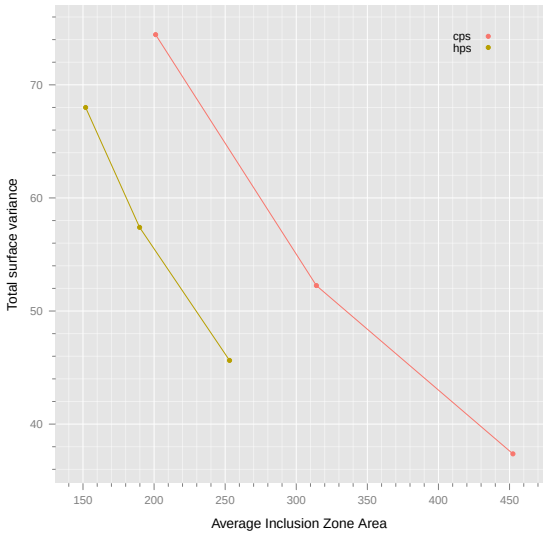


Figure 13: H. F. Smith plot comparing circular plot and horizontal point sampling using the total `sampSurf` variance for volume on a 3-tree population.

Note that the results conform to our expectations, that overall, HPS is more efficient that CHS, which is more efficient that CPS.

These methods can also be applied to the MODWT decomposition of the respective sampling surfaces to see how the methods compare at differing spatial scales. The results for the BEF are presented in Figure 15.

```
R> plot(bef.hfs$modwt$plt)
```

Note that on first glance, something perhaps anomolous is happening the smaller scales. Here, the variance is smaller for CHS for scales up to (and perhaps including at smaller BAFs) $\tau' = 4$ft. However, if we consider the shape of a CHS surface verses a HPS surface, coupled with what we now know about MODWT decompositions at smaller scales, this begins to make some sense. Consider a sampling surface for one tree under both these methods. The sampling surface for CHS will be approximated by a parabolic shape, while that for HPS is a cylinder. When the wavelet filter hits the wall of the HPS cylinder, it registers a large difference, but then advancing across the cylinder top, the difference is zero until it again hits the other edge, which records a negative difference. The sum total in terms of energy results in large variation going from zero to cylinder height and back again. Conversely, the parabolic surface of CHS takes a small jump at the edge of the inclusion zone (smaller than that of the HPS jump) then gradually increases and decreases over the sampling surface height. Thus, the surface variance at small scales is less than that for HPS given the same tree and sampling characteristics. The CPS interpretation is similar to that of HPS since it is also a cylinder.

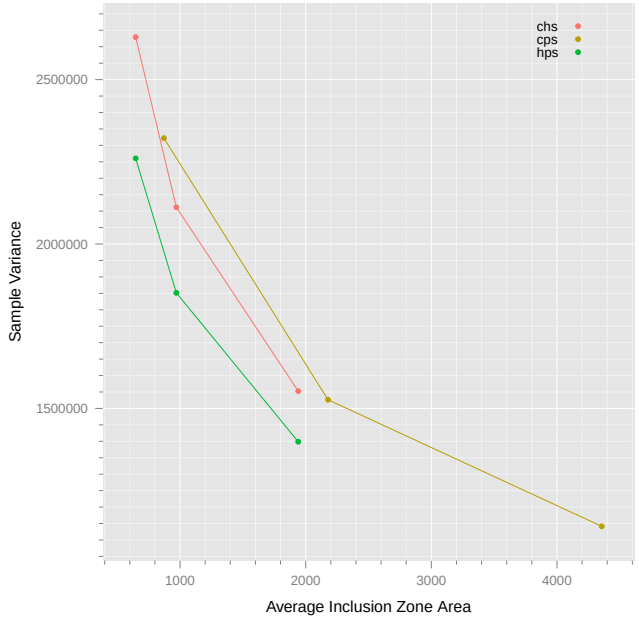


Figure 14: H. F. Smith plot comparing circular plot, critical height and horizontal point sampling at different sampling efforts on the BEF population (see text for details).

The above is a reasonable explanation for a single tree. However, the BEF is a plot with over 70 trees. It is not at all clear why the above explanation should carry over to a more complex surface. It must be that the larger sharp jumps of HPS are still preserved within this sampling surface, as are the smoother, smaller jumps of CHS. Also, CPS keeps its ranking as the least efficient for volume in this case, though midlevel scales with smaller plot sizes are similar to CHS. This is an interesting and perhaps unintuitive result for a larger population. Whether it really means much in terms of sampling is uncertain until further simulations can be made.

One potentially useful extension of the above results to the field would be that small shifts (i.e., a few feet) in point placement are less likely to cause any significant change in the volume estimate under CHS. On the other hand, while not all such small shifts will cause a change under HPS or CPS, there is the capacity for a large change in volume when a tree is either added or missed based on a small change in the sample point position. This is also fairly intuitive when one considers it, but without an analysis such as the wavelet decomposition, the reason for considering it, and the tools for quantitative interpretation rather than intuition are lacking.

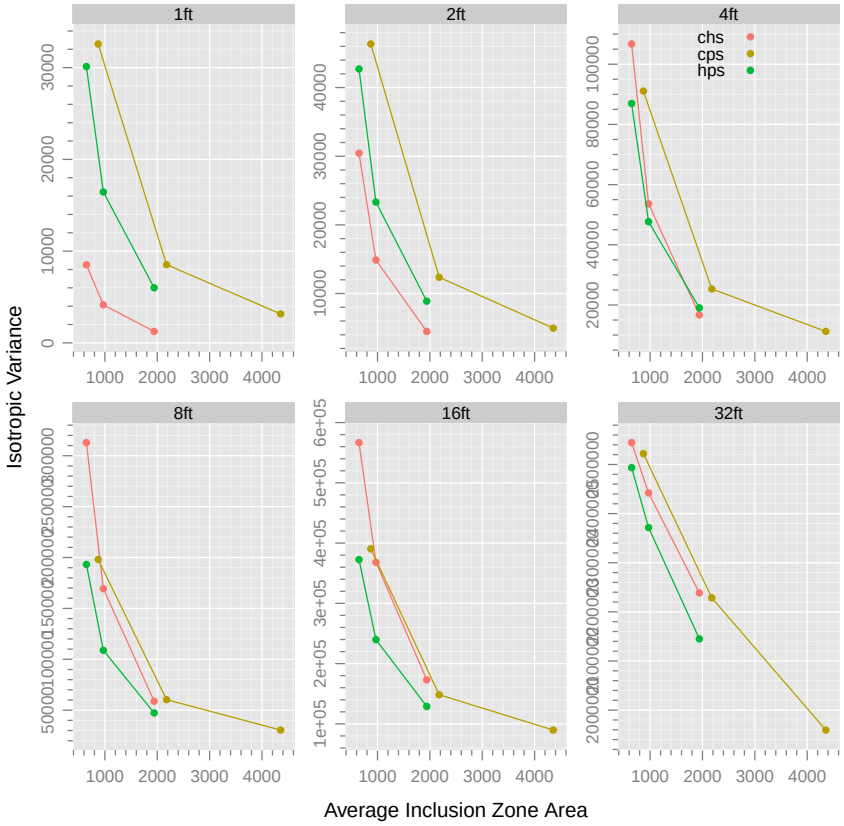


Figure 15: H. F. Smith plot comparing circular plot, critical height and horizontal point sampling at different sampling efforts on the MODWT decomposition of the BEF population (see text for details).

References

P. S. Addison. *The illustrated wavelet transform handbook: Introductory theory and applications in science, engineering, medicine and finance*. Taylor & Francis, 2002. 33, 34

M. Geilhufe, D. B. Percival, and H. L. Stern. Two-dimensional wavelet variance estimation with application to sea ice sar images. *Computers and Geosciences*, 54:351–360, 2013. 2, 3, 4, 6, 9, 35

R. Gençay, F. Selçuk, and B. Whitcher. *An introduction to wavelets and other filtering methods in finance and economics*. Academic Press, 2002. 2, 36

J. H. Gove. sampSurf: Sampling surface simulation. 2012. URL `https://r-forge.r-project.org/projects/sampsurf/`. 1

J. H. Gove, M. J. Ducey, W. B. Leak, and L. Zhang. Rotated sigmoid structures in managed

uneven-aged northern hardwood stands: a look at the Burr type III distribution. *Forestry*, 81 (2):161–176, 2008. 13

R. J. Hijmans. raster: Geographic data analysis and modeling. 2016. URL `http://cran.r-project.org/package=raster`. 3

R. M. Lark and R. Webster. Analysing soil variation in two dimensions with the discrete wavelet transform. *European Journal of Soil Science*, 55:777–797, 2004. 5, 6, 33, 34, 35

W. B. Leak and J. H. Gove. Growth of northern hardwoods in New England—a 25-year update. *Northern Journal of Applied Forestry*, 25(2):103–105, 2008. 13

W. B. Leak and D. S. Solomon. Influence of residual stand density on regeneration of northern hardwoods. Research Paper NE-310, USDA Forest Service, Northeastern Research Station, Upper Darby, PA, 1975. 13

Z. Ma and L. Zhang. Modeling bird species richness at multiple spatial scales using two-dimensional wavelet analysis. *Forest Science*, 61(1):1–16, 2015. 34, 35

D. Mondal and D. B. Percival. Wavelet variance analysis for random fields on a regular lattice. *IEEE Transactions on Image Processing*, 21(2):537–549, 2012. 6, 7, 9, 10

D. B. Percival and D. Mondal. *A wavelet variance primer*, volume 30 of *Handbook of Statistics*, chapter 22. Elsevier, 1st edition, 2012. 6, 7, 9

D. B. Percival and A. T. Walden. *Wavelet methods for time series analysis*. Cambridge University Press, 2000. 2, 3, 4, 6, 7, 22

B. D. Ripley. *Spatial Statistics*. John Wiley and Sons, 1981. 33, 35

H. F. Smith. An empirical law describing heterogeneity in the yields of agricultural crops. *Journal of Agricultural Science*, 28:1–23, 1938. 50

J. S. Walker. *A primer on wavelets and their scientific applications*. CRC Press, second edition, 2008. 3

B. Whitcher. waveslim: Basic wavelet routines for one-, two- and three-dimensional signal processing. `http://lib.stat.cmu.edu/R/CRAN/web/packages/waveslim/index.html`. 2