The "InclusionZoneGrid" Class*

J. H. Gove

USDA Forest Service, Northern Research Station, 271 Mast Road, Durham, NH 03824 USA

(603) 868-7667; e-mail: jgove@fs.fed.us

Tuesday 10th December, 2013 3:13pm

Contents

1 Introduction

When we think about building a sampling surface piece by piece from the inclusion zones of individual "Stem" objects, we assign the appropriate attribute value to each grid cell within the inclusion zone for an object, with zero values elsewhere, and then algebraically add this grid layer to the overall "Tract" grid. Thus, we "heap up" the inclusion zone density of the grid cells within the tract, which in the end is a discrete estimate of the sampling surface.

*R `sampSurf` package vignette series paper: http://sampsurf.r-forge.r-project.org/.

This class allows us to do the geometry associated with the individual "InclusionZone" objects for each "Stem" object in the population. Recall that an "InclusionZone" object has both an "ArealSampling" and "Stem" subclass as slots in its definition. For example, an object of subclass "standUpIZ" would have both a "circularPlot" class object and a "downLog" object making up the overall bounding box. There are methods for the "InclusionZoneGrid" class objects that work with each type of "InclusionZone" object. In general, the methods are all of the name `izGrid`, whose signature objects are used to dispatch the appropriate method.[1] The general class hierarchy is presented in Figure 1.

In the following, we show some general constructs of the class with respect to individual subclasses of "InclusionZone" objects they are based on. Graphics play a big role in getting the idea, and each class is a little different, so several illustrations are presented. Whether a surface within an inclusion zone is constant or variable height depends on the sampling method. For example, what we term "canonical" perpendicular distance sampling (PDS) has constant height surfaces, while "omnibus" PDS has surfaces of varying height. In addition, the somewhat odd, but interesting "full chainsaw" method where the entire sausage-based inclusion zone is filled with variable-height individual chainsaw estimates is also in this latter category as are the other Monte Carlo-based methods.

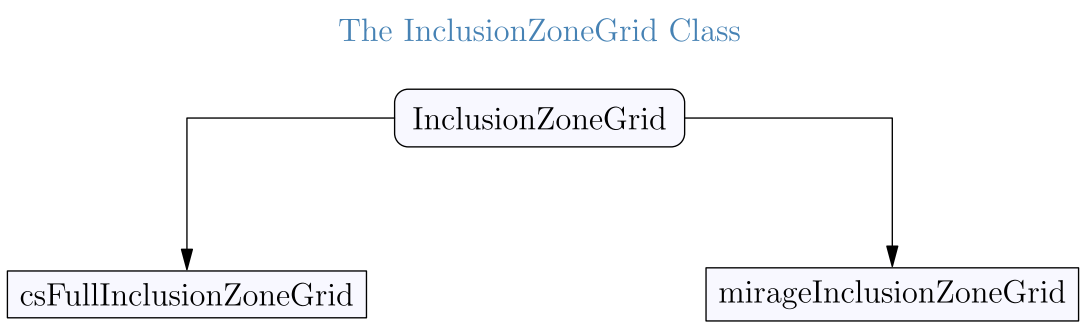

Figure 1: An overview of the "InclusionGridZone" class.

2 The "InclusionZoneGrid" Class

The base class is defined with the slots...

```
R> showClass('InclusionZoneGrid')
```

[1] An exception is when the mirage method is used as we will see later in § 24.

```
Class "InclusionZoneGrid" [package "sampSurf"]

Slots:

Name:    description            iz          grid         data          bbox
Class:     character InclusionZone   RasterLayer   data.frame        matrix

Extends: "izgNULL"

Known Subclasses: "csFullInclusionZoneGrid", "mirageInclusionZoneGrid"
```

2.1 Class slots

- *description*: Some descriptive text about this class.
- *iz*: An object of one of the "InclusionZone" subclasses.
- *grid*: A "RasterLayer" object.
- *data*: A data frame holding the values for each of the per unit area estimates available in the "InclusionZone" object in the columns, with rows for grid cells.
- *bbox*: The overall bounding box for the object, which includes the inclusion zone and the "Stem" subclass object plus the grid. Sometimes the inclusion zone itself includes the stem, but other times it does not.

I made a decision when designing this class to go with the slots above. A possible problem with this design is the very real possiblity of people misunderstanding the class structure. This is because the grid object has values of just zero and `NA`. In other words, it does not store the per unit area estimates within the cells of the grid. These are stored in the `data` slot of the object and can be swapped into the `grid` slot object with a simple command...

```
R> x@grid = setValues(x@grid, x@data[,estimate])
```

where `x` is the "InclusionZoneGrid" object. Unfortunately, again this may cause problems with misunderstanding, down the line as it is going to have to be done whenever one wants an underlying `grid` with real per unit area values. An alternative to this would have been (and still could be) to use a "RasterStack" or "RasterBrick" class for the `grid` slot. This would obviate the need for the `data` slot. But it also has drawbacks, because these objects seem to be designed more for map layers that are algebraically related. I did not want people unwittingly summing layers within this object, for example, and thereby adding things like number of stems and cubic volume. I may reconsider this, however, as the latter approach does have its benefits (even though is takes more storage). Another potential drawback of the latter is that there seems to be no way to name the "layers" within a brick or stack, they simply get assigned numbers, so we'd again have to keep track of this with program code.

When plotting the object, the above substitution gets made automatically, one simply has to specify the desired attribute to plot in the `estimate` argument to the `plot` method. For example, to plot the coverage area surface, specify `estimate = 'coverageArea'` in the plot command. The default is to plot the surface for volume.

Many of the examples that follow show summary reports for the "InclusionZoneGrid" objects. These reports contain a data frame with summary statistics for each attribute estimate. In addition, they now (summer 2013) contain a new column labeled `depth`. The depth column will always be a column of 1's within the inclusion zone, 0's background. It is used to record the sample depth (number of overlap zones) at each grid cell when the surface is constructed in `lapSurf`. More details are given in Gove (2013c).[2] Additionally, one may specify `estimate='depth'` in a call to `sampSurf` to plot the depth of overlap surface.

3 Example: The "standUpIZ" Class

Refer to *The "InclusionZone" Class* vignette for more information on this class.

Here we demonstrate the construction of an "InclusionZoneGrid" object from an object of class "standUpIZ".

```
R> tra = Tract(c(x=100, y=100), cellSize = 0.5, units = 'metric',
+              description = 'a 1-hectare tract')
R> btr = bufferedTract(10, tra)
R> btr
```

```
------------------------------------------------------------
a 1-hectare tract
------------------------------------------------------------
Measurement units = metric
Area in square meters = 10000 (1 hectares)

class       : bufferedTract
dimensions  : 200, 200, 40000  (nrow, ncol, ncell)
resolution  : 0.5, 0.5  (x, y)
extent      : 0, 100, 0, 100  (xmin, xmax, ymin, ymax)
```

[2]The "lapSurf" class is a work in progress, and may not be included in the CRAN version yet (summer 2013).

```
coord. ref. : NA
data source : in memory
names       : surf
values      : 0, 0  (min, max)

Buffer width =  10
```

```
R> dlogs = downLogs(1, container=btr@bufferRect, buttDiam=c(30,40),
+                   logLen=c(6,10), topDiams=c(0,0.5), solidTypes=c(2,4),
+                   vol2wgt=20.1, wgt2carbon=0.5)
R> sup = standUpIZ(dlogs@logs$log.1, 3)
R> izgSU = izGrid(sup, btr)
```

Here we have created a 200 by 200 cell raster grid in the form of a "Tract" object, having resolution of 0.5 meters, yielding spatial extents of 100×100 meters, (i.e., a 1 hectare tract), with origin at $(0, 0)$ meters. Then we create a buffered tract object with a 10-meter buffer, and drew a single random "downLog" object within the tract, which we used to create an object of class "standUpIZ" with a 3 meter radius for the circular plot. Finally, using the "standUpIZ" object and the underlying buffered tract grid, we create the "InclusionZoneGrid" object that will be aligned to the tract grid.

Now, in the following example we plot this object...

```
R> plot(izgSU)
```

There are two ways to generate the object which depend on how one wants the underlying grid developed. The argument `wholeIZ` determines how this is done, and defaults to `TRUE`. We can see in Figure 2 that the underlying grid covers the entire inclusion zone plus the down log object. If we wanted to just cover the inclusion zone only, with a minimal bounding grid we would do the following (Figure 3)...

```
R> izmbgSU = izGrid(sup, btr, wholeIZ=FALSE)
R> plot(izmbgSU, gridCenters=TRUE)
```

4 On Design Motivation

Having now seen that we can make the grid object cover more than just the inclusion zone if applicable (i.e., the stem lies partially outside it), we can delve a little more into the motivation for this class. First, it is entirely possible to approach the sampling surface construction in a more brute-force manner in one of at least two ways...

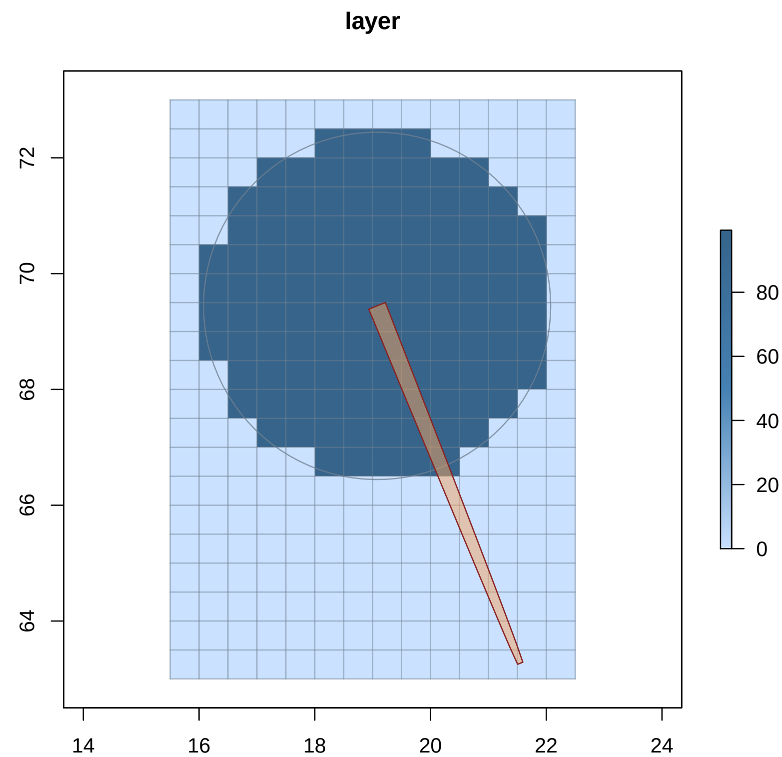

Figure 2: An "InclusionZoneGrid" object based on a "standUpIZ" object.

1. Brute-force method 1...
   - Make an exact duplicate of the tract and assign zeros to all its values.
   - Then overlay the inclusion zone onto this to get a mask.
   - Assign the per acre values to the cells within the masked inclusion zone and zero outside.
   - Algebraically add this layer to the base tract and repeat for all stems.
2. Brute-force method 2...
   - Use the `rasterize` function within "raster" with `overlap='sum'` on the "InclusionZone" objects to sum them into the base tract grid.
   - Repeat for all stems.

The problem with each of these approaches, even though they work, is the time it takes to implement them. In each case they are working on the entire tract and take significantly more time to

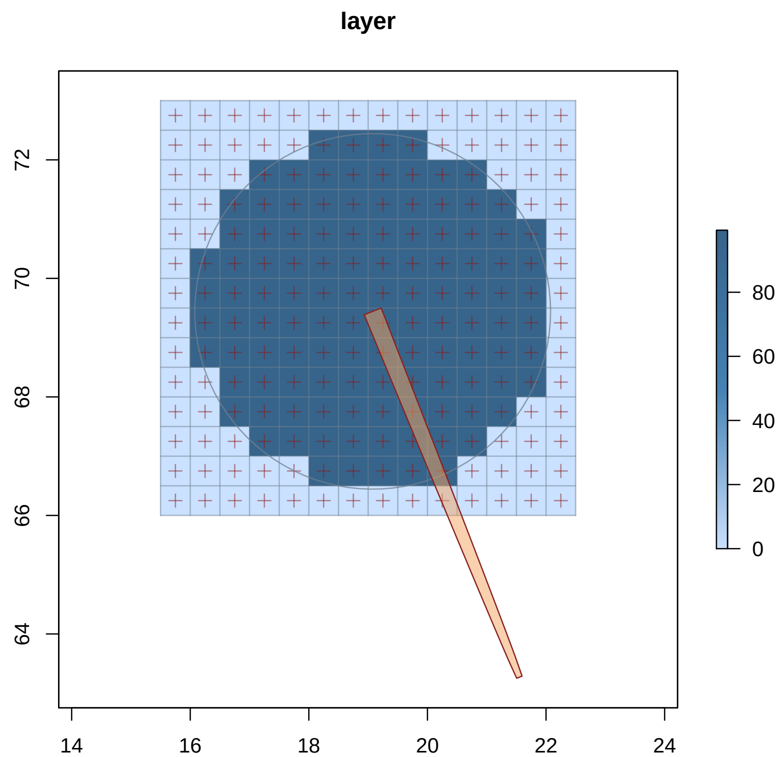

Figure 3: An "InclusionZoneGrid" object based on a "standUpIZ" object where only the actual inclusion zone is covered by the bounding grid, and grid cell centers are plotted as an option.

do the accumulation than overlaying onto the smaller grid within the "InclusionZoneGrid" class. This is compounded if the tract is large, or the resolution is small, which we want usually for better estimates. This, along with the thought that perhaps the following approach is simpler to understand because one can see the individual surface components graphically, is why I elected to go this route.

In slightly more detail, the steps in accumulating the surface under the "InclusionZoneGrid" scheme are...

1. Create an "InclusionZoneGrid" object.
2. Expand its grid to the extent of the tract.
3. Add this expanded grid to the tract surface.
4. Repeat for all stems.

Conceptually it is similar to the other approaches except that the actual overlay is done on a minimal bounding grid for the object, and therefore is much faster. Expanding the grid mask in the "InclusionZoneGrid" object takes little effort, and adding it is the same in all steps.

5 Example: The "sausageIZ" Class

Using the same grid and tree as above for an example with sausage sampling, we have...

```
R> saus = sausageIZ(dlogs@logs$log.1, 3)
R> izgSAUS = izGrid(saus, btr, wholeIZ=FALSE)
R> plot(izgSAUS)
```

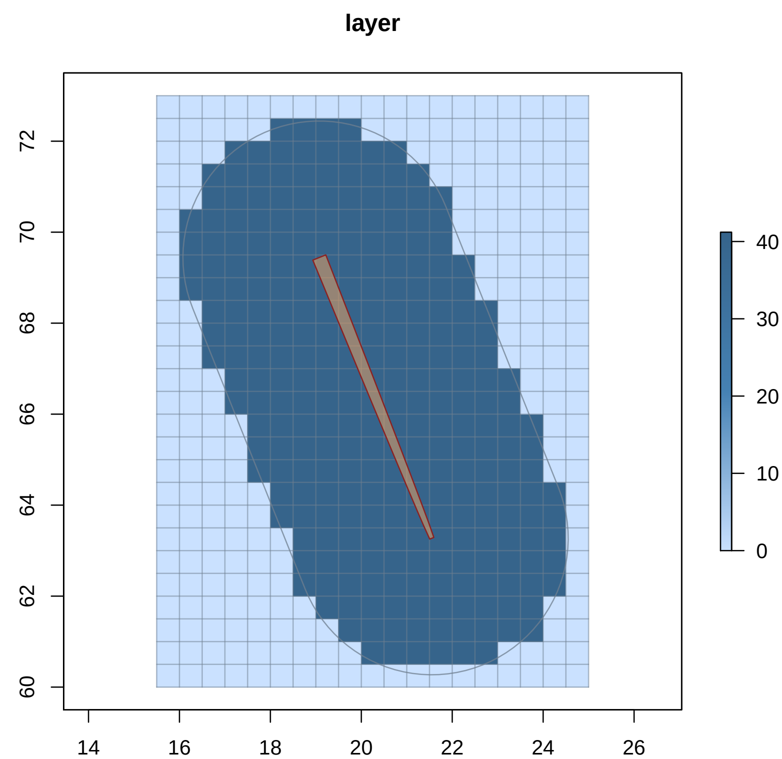

Figure 4: An "InclusionZoneGrid" object based on a "sausageIZ" object showing that the stem and inclusion zone are covered by the bounding grid.

Notice in Figure 4 that whether we specify using the whole inclusion zone or not is immaterial for sausage sampling, because the entire log is always included within the zone. Also note that the minimal bounding grid is always calculated to include the entire zone, even if there is an extra cell padding all around. This is trivial and is necessary for making sure all cells within the zone get assigned the correct value.

6 Example: The "chainSawIZ" Class

Here are a couple similar examples for the "chainSawIZ" class. Gove and Van Deusen (2011) discuss how the inclusion zone is really just a point, the center point of the circular plot that intersects the downed log, rather than the plot itself. Therefore, under this method, we assign the sampling surface value to only that one grid cell that contains the circular plot center point.

```
R> csaw = chainSawIZ(dlogs@logs$log.1, plotRadius = 3,
+                    plotCenter = coordinates(dlogs@logs$log.1@location)[1,] + c(1,-1))
R> izgCSaw = izGrid(csaw, btr, wholeIZ=FALSE)
R> izgCSaw
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
chainSaw grid point inclusion zone grid object
------------------------------------------------------------

InclusionZone class: chainSawIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 1
Cell dimensions: (nrows=1, ncol=1)
Grid cell values**...
  gridValues Freq
1          0    1
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume Density Length surfaceArea coverageArea biomass carbon depth
Min.     60.05   353.7   1767        1123        357.4    1207  603.5     1
1st Qu.  60.05   353.7   1767        1123        357.4    1207  603.5     1
Median   60.05   353.7   1767        1123        357.4    1207  603.5     1
Mean     60.05   353.7   1767        1123        357.4    1207  603.5     1
```

```
3rd Qu.  60.05   353.7   1767         1123         357.4    1207  603.5     1
Max.     60.05   353.7   1767         1123         357.4    1207  603.5     1


  Encapulating bounding box...
        min       max
x 18.314990 24.313480
y 62.357548 69.500813
```

```
R> plot(izgCSaw, gridCenters=TRUE)
```

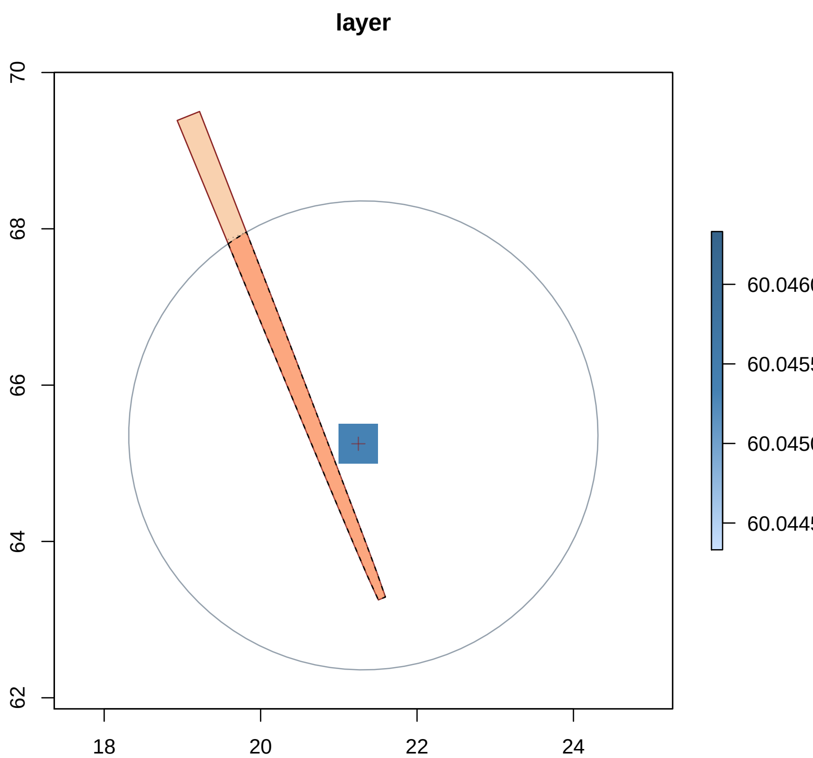

Figure 5: An "InclusionZoneGrid" object based on a "chainSawIZ" object showing that the inclusion zone is only a single point, and therefore is assigned to just one grid cell.

Figure 5 shows the concept for a given circular plot location showing volume in cubic meters per hectare. This clearly shows that only one grid cell gets assigned a value. Note in the above that there is only a single grid cell in the object comprising the "InclusionZoneGrid".

We can also show the minimal bounding grid that includes the whole circular plot plus the log, as we have done in previous examples. Figure 6 presents this graphically.

```
R> izgCSaw = izGrid(csaw, btr, wholeIZ=TRUE)
R> plot(izgCSaw, gridCenters=TRUE, estimate='Density',
+       showPlotCenter=TRUE, izCenterColor = 'white')
```

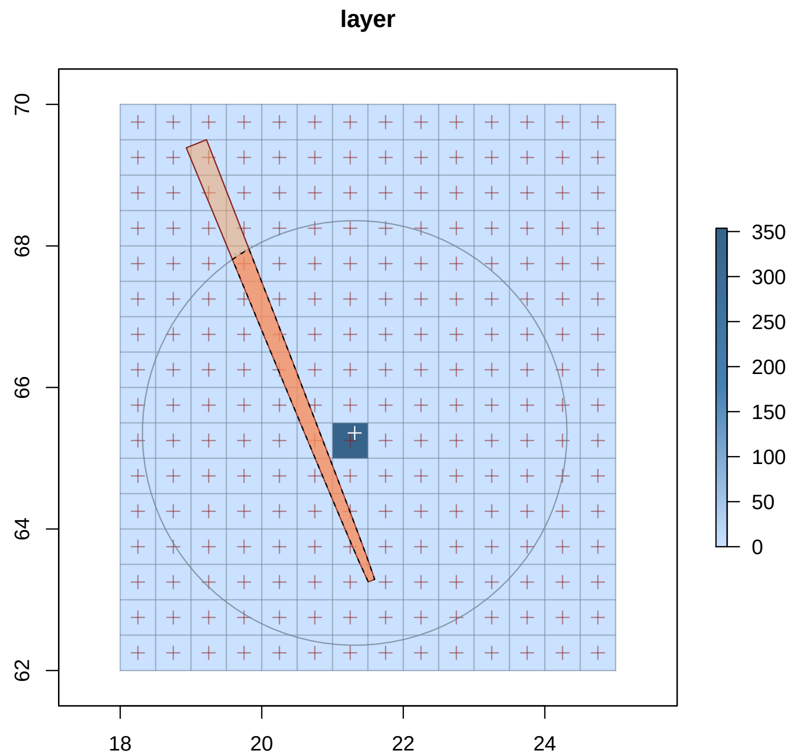

Figure 6: An “InclusionZoneGrid” object based on a “chainSawIZ” object showing the minimal bounding grid for the entire inclusion zone object; this also illustrates that the inclusion zone is only a single point, since only one grid cell is non-zero valued in terms of number of stems per unit area estimate.

As with the other figures, the overall minimal bounding grid in Figure 6 has all grid cells other than the one at the center of the circular plot set to zero. Therefore, doing any subsequent map algebra will have no effect on the sampling surface using this enlarged grid. Showing this minimal bounding grid is more an effort to help illustrate the underlying concepts.

It is especially important to recognize that the center point of the circular plot does not necessarily lie at the center of any given grid cell. Depending upon the cell resolution used, this can make the placement of this respective cell look "off-center" with respect to the circular plot's perimeter itself. It is only true that the center point falls somewhere within the inclusion zone grid cell, which could even be right on the edge. This is shown in Figure 6 where we use a white cross for the circular plot center, and is demonstrated below in terms of actual coordinates...

```
R> cpt = perimeter(csaw, whatSense='point') #circular plot centerpoint
R> cn = cellFromXY(izgCSaw@grid, cpt)       #cell number for plot center point
R> xy = xyFromCell(izgCSaw@grid, cn)        #cell center point
R> rbind(coordinates(cpt), xy)
```

```
            x         y
[1,] 21.31348 65.357171
[2,] 21.25000 65.250000
```

6.1 Snapping to the grid

As another example, suppose we wanted to develop a figure that shows essentially the same depiction as in Figure 6, but also including the overall sausage inclusion zone, and the plot center exactly aligned to a grid cell center. First we would make a "Tract" object that just holds the sausage inclusion zone object, the log, and the chainsaw inclusion zone object (complete with plot radius as in Figure 6). We can use the above steps to advantage to snap the plot centerpoint to the grid as follows...

```
R> xyExtent = c(x=10, y=10)
R> tra2 = Tract(xyExtent, cellSize=0.5)
R> dl = downLog(buttDiam=40, topDiam=15, logLen=6.5, logAngle=pi/4,
+               centerOffset=xyExtent/2 )
R> cn = cellFromXY(tra2, c(x=6.5, y=4.5))
R> (cpt = xyFromCell(tra2, cn)[,,drop=TRUE])  #coerce to vector from matrix
```

```
   x    y
6.75 4.25
```

```
R> izCS = chainSawIZ(dl, plotRadius = 2, plotCenter = cpt) #cell-based chainsaw iz
R> izgCS = izGrid(izCS, tra2)                      #and chainsaw iz grid object
R> hiz = heapIZ(izgCS, tra2)                       #heap it into the tract object
R> izSaus = sausageIZ(dl, plotRadius=2)            #overall sausage inclusion zone
```

We use the `heapIZ` method in the above to "heap" the "InclusionZoneGrid" object for the single grid cell corresponding to the chainsaw method at that point, onto the tract. The rest of the idea behind this should be fairly standard and is covered in more detail in the vignettes for the respective objects.

To plot this object, we need to basically build it up from scratch, the result is shown in Figure 7...

```
R> plot(hiz, axes=TRUE, gridLines=TRUE)
R> plot(dl, add=TRUE)
R> plot(izSaus, add=TRUE, izColor=NA, lty='dashed', izBorder='gray40')
R> plot(izCS, add=TRUE, izColor=NA, showPlotCenter=TRUE,
+       izCenterColor='white', ltyBolt='solid')
```

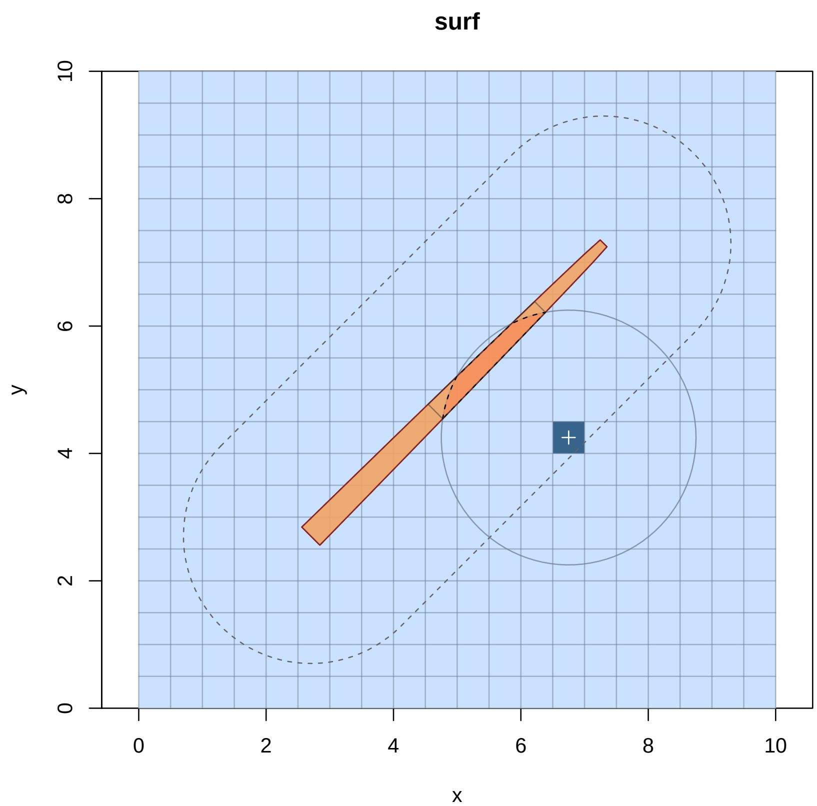

Figure 7: A built-up plot showing not only the "InclusionZoneGrid" object for one cell of the chainsaw method, but also the sausage object inclusion zone as would be determined by protocol 1 of Gove and Van Deusen (2011).

7 The "csFullInclusionZoneGrid" Class

Gove and Van Deusen (2011) describe a certain protocol for the chainsaw method that leads directly to the sausage method. They also show by simulation how the chainsaw method is biased for whole-log attributes, because the full inclusion zone for the log is the sausage zone under this particular protocol. To show this, every grid point within the sausage inclusion zone has to be estimated individually with its midpoint acting as the center of the circular plot, and then applying the chainsaw method to that plot. Again, this is repeated for every grid cell within the sausage inclusion zone.

This necessitates a new class, actually a subclass of "InclusionZoneGrid" with one extra slot, and some more validity checking, as well as a new constructor for the objects. The extra slot is just used to store the result of *each* "chainSawIZ" object applied to each of the internal inclusion zone grid cells.

The class is defined with the slots...

```
R> showClass('csFullInclusionZoneGrid')
```

```
Class "csFullInclusionZoneGrid" [package "sampSurf"]

Slots:

Name:            chiz   description            iz          grid          data
Class:           list     character InclusionZone   RasterLayer    data.frame

Name:            bbox
Class:         matrix

Extends:
Class "InclusionZoneGrid", directly
Class "izgNULL", by class "InclusionZoneGrid", distance 2
```

7.1 Class slots

- *chiz*: This is a list object containing `NA`s for cells outside the inclusion zone, but containing the full set of "InclusionZoneGrid" objects corresponding to each grid cell within the inclusion zone. As mentioned above, the grid cell center is used as the center point of the circular plot that defines the chainsaw intersection with the log.

The nice thing about the subclass extension for this new object is that only one slot was added; therefore, all of the functions that work on "InclusionZoneGrid" objects will also work on objects of this new class "csFullInclusionZoneGrid". These include the `print`, `show`, `summary`, and `plot` routines.

In addition, because each of the componets of the list in `chiz` is of class "InclusionZoneGrid" (or `NA`), we can apply any of the methods for that class on the individual slots. For example, we can plot them as in Figure 5, and look at how the chainsaw method works as we step from one cell to the next, showing the intersections of the circular plots with the log.

7.2 Object Construction and Plotting

Now, object construction takes quite a while because it has to compute the chainsaw intersections, etc., for each internal sausage grid cell. So here we use an existing object to demonstrate the idea. In the following, we show how to make an object of class "csFullInclusionZoneGrid" using its constructor, but do not evaluate it (we use the existing object instead); the steps leading to its creation are also shown...

```
R> btLog = sampleLogs(1, buttDiam=c(30,40), sampleRect=buffTr@bufferRect,
+                     logLen=c(4,6), topDiams=c(0, 0.5) )
R> btLog = downLogs(btLog)
R> btLog = btLog@logs$log.1
R> btLog.fcs = fullChainSawIZ(btLog, 3)
R> btLog.izgFCS = izGrid(btLog.fcs, buffTr)
```

where `buffTr` is essentially the same as `btr` used in the previous examples.

```
R> btLog.izgFCS
```

```
Object of class: csFullInclusionZoneGrid
------------------------------------------------------------
Full chainSaw (sausage) inclusion zone grid object
------------------------------------------------------------

InclusionZone class: fullChainSawIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 441
Cell dimensions: (nrows=21, ncol=21)
```

```
Grid cell values**...
  gridValues Freq
1          0  232
2       <NA>  209
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
          volume Density  Length surfaceArea coverageArea biomass  carbon depth
Min.      0.6743   353.7   38.34       13.02        3.783    14.7    7.35     1
1st Qu.  28.1740   353.7  497.32      383.19      121.070   614.2  307.10     1
Median   64.9000   353.7  873.49      841.26      266.890  1414.8  707.41     1
Mean     66.9720   353.7  907.75      847.21      269.220  1460.0  730.00     1
3rd Qu. 103.3000   353.7 1311.70     1264.40      402.460  2252.0 1126.00     1
Max.    138.3300   353.7 1775.50     1751.30      556.470  3015.5 1507.80     1


  Encapulating bounding box...
   min max
x 67.5  78
y 66.5  77
```

One thing to note in particular, is that with the exception of density in the printed summary of the object above, the summary statistics vary here because the grid cells are composed of individual chainsaw estimates; this is not true for methods where the cells internal to the inclusion zone only take a single constant value (e.g., stand-up, sausage, point relascope, etc.).

And plotting the object is as usual...

```
R> plot(btLog.izgFCS, gridCenters=TRUE, showNeedle=TRUE)
```

As mentioned above, because the `chiz` slot contains a list of "InclusionZoneGrid" objects, for each grid cell within the inclusion zone, or `NA` for grid cells outside the zone, we can step through the internal cells one at a time, looking at summaries or plotting them. One way to do this would be the following...

```
R> cdx = ifelse(is.na(btLog.izgFCS@chiz), FALSE, TRUE)
R> table(cdx)
```

```
cdx
FALSE  TRUE
  209   232
```

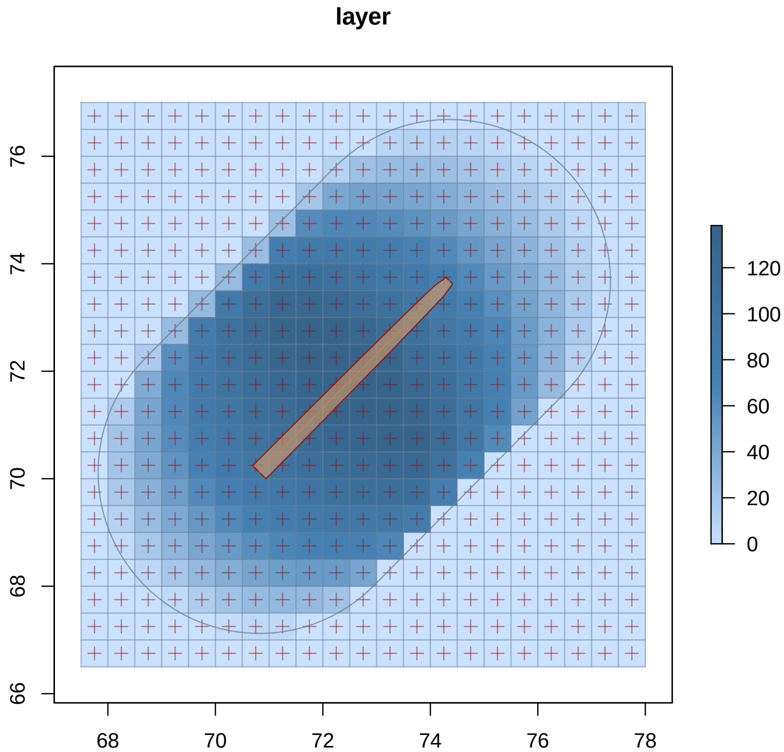

Figure 8: A "csFullInclusionZoneGrid" object based on a "sausageIZ" object.

```
R> csl = btLog.izgFCS@chiz[cdx]
R> length(csl)
```

```
[1] 232
```

```
R> sapply(csl[1:4], class)
```

```
           izgCS.33             izgCS.34             izgCS.35             izgCS.36
"InclusionZoneGrid" "InclusionZoneGrid" "InclusionZoneGrid" "InclusionZoneGrid"
```

Again, we could then plot the slivers we are interested in, stepping through to see how the chainsaw method slices the log up for each individual grid cell.

8 Example: The "pointRelascopeIZ" Class

Here we present an example for the point relascope sampling method (Gove et al. 1999, Gove et al. 2001). . .

```
R> (angle = .StemEnv$rad2Deg(2*atan(1/2)))
```

```
[1] 53.130102
```

```
R> prs.as = pointRelascope(angle, units='metric')
R> prs.iz = pointRelascopeIZ(dlogs@logs$log.1, prs=prs.as)
R> (izgPRS = izGrid(prs.iz, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
pointRelascopeIZ inclusion zone grid object
------------------------------------------------------------

InclusionZone class: pointRelascopeIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 616
Cell dimensions: (nrows=22, ncol=28)
Grid cell values**...
  gridValues Freq
1          0  371
2       <NA>  245
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume Density Length surfaceArea coverageArea biomass carbon depth
Min.     30.18   107.4  714.4       505.9          161   606.6  303.3     1
1st Qu.  30.18   107.4  714.4       505.9          161   606.6  303.3     1
Median   30.18   107.4  714.4       505.9          161   606.6  303.3     1
Mean     30.18   107.4  714.4       505.9          161   606.6  303.3     1
3rd Qu.  30.18   107.4  714.4       505.9          161   606.6  303.3     1
Max.     30.18   107.4  714.4       505.9          161   606.6  303.3     1
```

```
  Encapulating bounding box...
   min  max
x 13.5 27.5
y 61.0 72.0
```

```
R> plot(izgPRS)
```

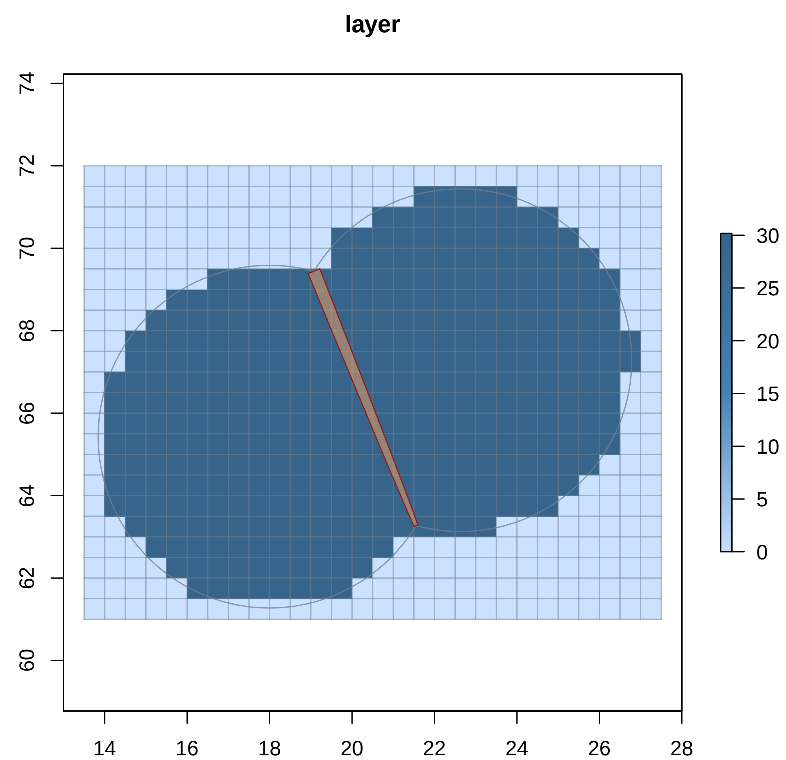

Figure 9: An "InclusionZoneGrid" object based on a "pointRelascopeIZ" object showing that the stem and inclusion zone are covered by the bounding grid.

9 Example: The "perpendicularDistanceIZ" Class

Here we present an example for the perpendicular distance sampling method (Williams and Gove 2003, Williams et al. 2005, Ducey et al. 2008), this example happens to be for volume estimation. Please note that, because we know the log's true volume from simulation, we can in fact estimate all the other attributes for the log. Normally, however, we do not know the true volume from field measurements, so we would only be able to estimate volume under "canonical" PDS, and would use the "omnibus" variant PDS given in the next section to estimate these other quantities. . .

```
R> pdsmet = perpendicularDistance(kpds=50, units='metric')
R> iz.pdsv = perpendicularDistanceIZ(dlogs@logs$log.1, pdsmet)
R> (izgPDS = izGrid(iz.pdsv, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
perpendicularDistanceIZ inclusion zone grid object
------------------------------------------------------------

InclusionZone class: perpendicularDistanceIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 288
Cell dimensions: (nrows=18, ncol=16)
Grid cell values**...
  gridValues Freq
1          0  113
2       <NA>  175
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume Density Length surfaceArea coverageArea biomass carbon depth
Min.       100   355.9   2367        1676        533.5    2010   1005     1
1st Qu.    100   355.9   2367        1676        533.5    2010   1005     1
Median     100   355.9   2367        1676        533.5    2010   1005     1
Mean       100   355.9   2367        1676        533.5    2010   1005     1
3rd Qu.    100   355.9   2367        1676        533.5    2010   1005     1
Max.       100   355.9   2367        1676        533.5    2010   1005     1


  Encapulating bounding box...
   min  max
x 15.0 23.0
y 62.5 71.5
```

```
R> plot(izgPDS)
```

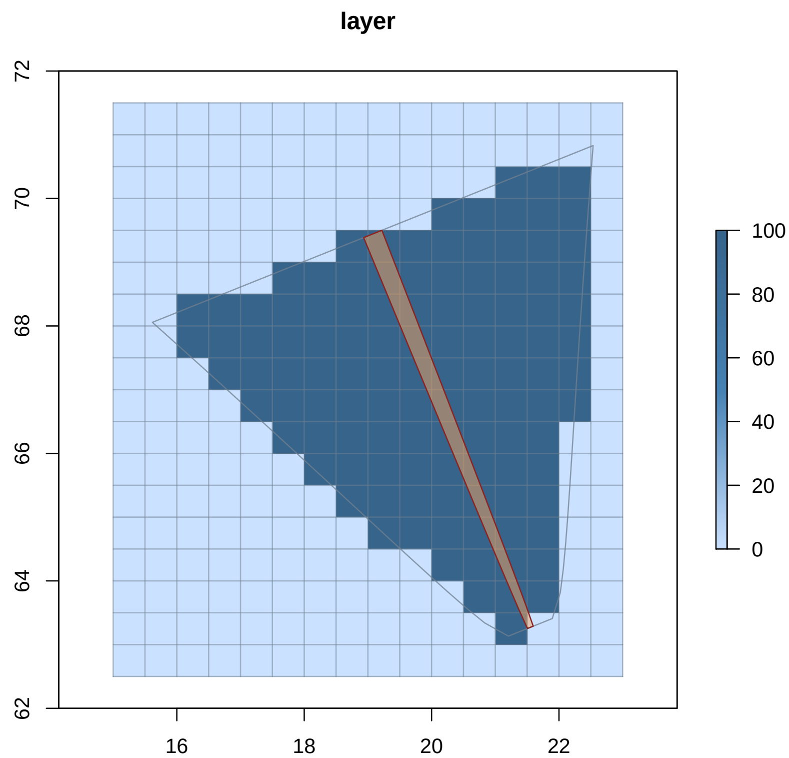

Figure 10: An "InclusionZoneGrid" object based on a "perpendicularDistanceIZ" object showing that the stem and inclusion zone are covered by the bounding grid.

As noted in *The InclusionZone Class* vignette, the inclusion zone for PDS is developed directly from the dataframe in the `taper` slot of the "downLog" object. It can be seen here that if that taper approximation is poor because it uses too few points, it has the potential to exclude grid cells that would otherwise normally be included. This in turn could lead to unexpected "simulation bias" in the sampling surface result (the same thing can happen from using too large a grid cell). Thus, it is fairly important to use a good number of log sections in the taper dataframe to avoid this possibility.

10 Example: The "omnibusPDSIZ" Class

An extension to "canonical" PDS presented in the previous section and given by Ducey et al. (2008), allows one to estimate any attribute on the log and is sometimes referred to as "omnibus"

PDS. Because this method uses stem measurements perpendicular to the sample point to form the estimates, it has varying height surface in each case, with the exception of the variable we use for the PPS selection of the log[3]...

```
R> iz.opds = omnibusPDSIZ(dlogs@logs$log.1, pdsmet)
R> (izgOPDS = izGrid(iz.opds, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
omnibusPDSIZ inclusion zone grid object
------------------------------------------------------------

InclusionZone class: omnibusPDSIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 288
Cell dimensions: (nrows=18, ncol=16)
Grid cell values**...
  gridValues Freq
1          0  113
2       <NA>  175
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume Density Length surfaceArea coverageArea biomass carbon depth
Min.       100   203.2   1351        1303        414.8    2010   1005     1
1st Qu.    100   235.2   1564        1402        446.2    2010   1005     1
Median     100   283.1   1882        1538        489.6    2010   1005     1
Mean       100   351.2   2336        1667        530.6    2010   1005     1
3rd Qu.    100   386.4   2569        1797        572.0    2010   1005     1
Max.       100  1566.1  10415        3618       1151.5    2010   1005     1


  Encapulating bounding box...
   min  max
x 15.0 23.0
y 62.5 71.5
```

```
R> plot(izgOPDS, estimate='coverageArea')
```

[3] Note that if log selection is with PP to volume, then since both biomass and carbon are simply scaled versions of volume, their surfaces will also be constant.

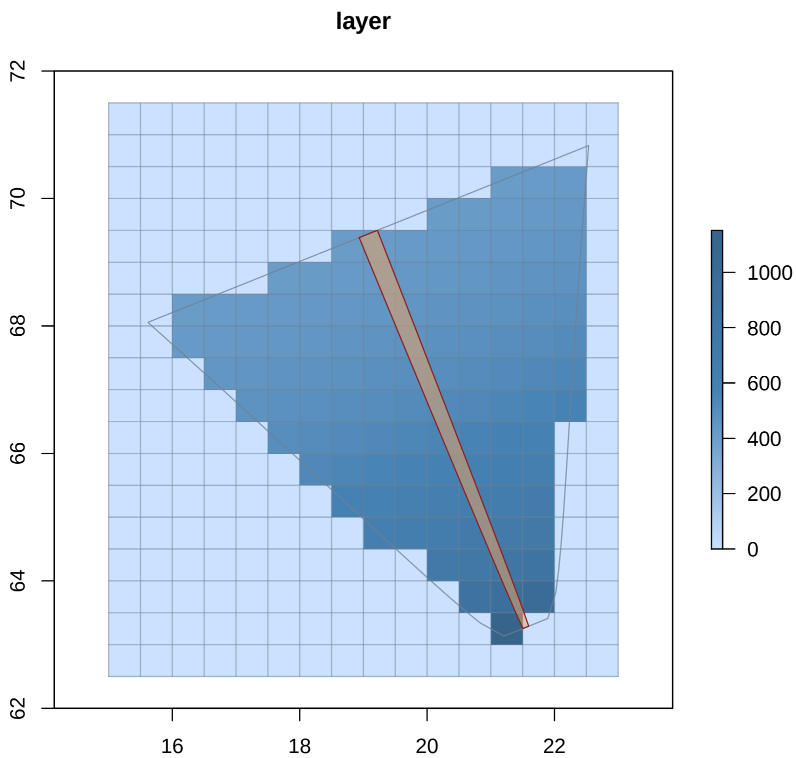

Figure 11: An "InclusionZoneGrid" object based on a "omnibusPDSIZ" object showing the variable height sampling surface for coverage area.

11 Example: The "distanceLimitedIZ" Class

Here we interject a method that is not a PDS variant, but we introduce it now because it is used in the distance limited PDS method presented below. There are two protocols for distance limited sampling: *(i)* a standard/canonical (DLS) with constant surface heights for all attributes, and *(ii)* a crude Monte Carlo based protocol (DLMCS) with varying surface height for most, but not all attributes. The first protocol is discussed here, while DLMCS is discussed in the next section. The protocols are discussed in detail in Gove et al. (2012). The inclusion zone is covered in *The "InclusionZone" Class* vignette.

```
R> dlsMet = distanceLimited(3, units='metric')
R> iz.dls = distanceLimitedIZ(dlogs@logs$log.1, dls=dlsMet)
R> (izgDLS = izGrid(iz.dls, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
distanceLimitedIZ inclusion zone grid object
------------------------------------------------------------

InclusionZone class: distanceLimitedIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 342
Cell dimensions: (nrows=19, ncol=18)
Grid cell values**...
  gridValues Freq
1          0  160
2       <NA>  182
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume Density Length surfaceArea coverageArea biomass carbon depth
Min.     70.41   250.6   1667        1180        375.7    1415  707.6     1
1st Qu.  70.41   250.6   1667        1180        375.7    1415  707.6     1
Median   70.41   250.6   1667        1180        375.7    1415  707.6     1
Mean     70.41   250.6   1667        1180        375.7    1415  707.6     1
3rd Qu.  70.41   250.6   1667        1180        375.7    1415  707.6     1
Max.     70.41   250.6   1667        1180        375.7    1415  707.6     1


  Encapulating bounding box...
   min max
x 16.0  25
y 61.5  71


R> plot(izgDLS, estimate='biomass')
```

The surfaces for log length and log density will be exactly the same under this protocol as the Monte Carlo variant as can be verified with the results in the following section. All other attribute surfaces will differ between the two protocols, as is illustrated for biomass in Figure 12.

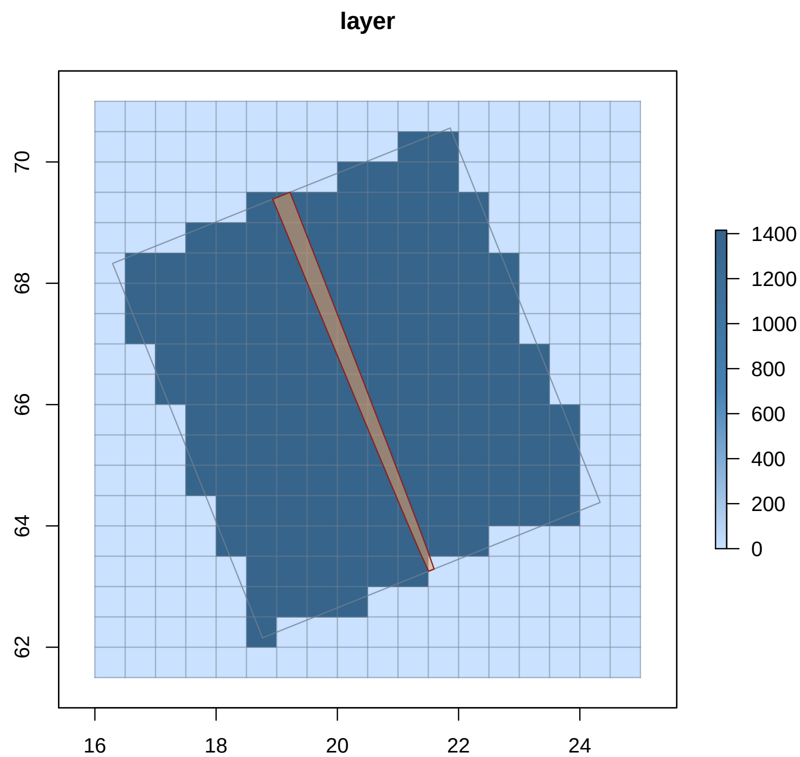

Figure 12: An "InclusionZoneGrid" object based on a "distanceLimitedIZ" object showing the variable height sampling surface for biomass.

12 Example: The "distanceLimitedMCIZ" Class

The Monte Carlo protocol for distance limited sampling is illustrated here. The inclusion zone is covered in *The "InclusionZone" Class* vignette, and the surface will be constant only for log length and density and be exactly the same as for DLS. An example follows...

```
R> dlsMet = distanceLimited(3, units='metric')
R> iz.dlmcs = distanceLimitedMCIZ(dlogs@logs$log.1, dls=dlsMet)
R> (izgDLMCS = izGrid(iz.dlmcs, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
distanceLimitedMCIZ inclusion zone grid object
```

```
------------------------------------------------------------

InclusionZone class: distanceLimitedMCIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 342
Cell dimensions: (nrows=19, ncol=18)
Grid cell values**...
  gridValues Freq
1          0  160
2       <NA>  182
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume Density Length surfaceArea coverageArea biomass carbon depth
Min.     12.46   250.6   1667       510.9        162.6   250.5  125.3     1
1st Qu.  44.17   250.6   1667       961.8        306.1   887.8  443.9     1
Median   70.03   250.6   1667      1211.1        385.5  1407.6  703.8     1
Mean     70.23   250.6   1667      1178.6        375.2  1411.7  705.9     1
3rd Qu.  96.41   250.6   1667      1421.0        452.3  1937.8  968.9     1
Max.    123.30   250.6   1667      1607.0        511.5  2478.3 1239.1     1


  Encapulating bounding box...
   min max
x 16.0  25
y 61.5  71
```

```
R> plot(izgDLMCS, estimate='biomass')
```

Note from the summary output that the surface is indeed variable for all attributes other than density and Length. This is illustrated for biomass in Figure 13.

13 Example: The “distanceLimitedPDSIZ” Class

This class, as explained in *The “InclusionZone” Class* vignette, is a sampling method that restricts the maximum width of the PDS inclusion zone, effectively truncating the search distance for logs. The composite effect comes from the fact that the inclusion zone can have one of three variations: *(i)* the inclusion zone is all PDS, *(ii)* the inclusion zone is all DLS, or *(iii)* it is a combination of

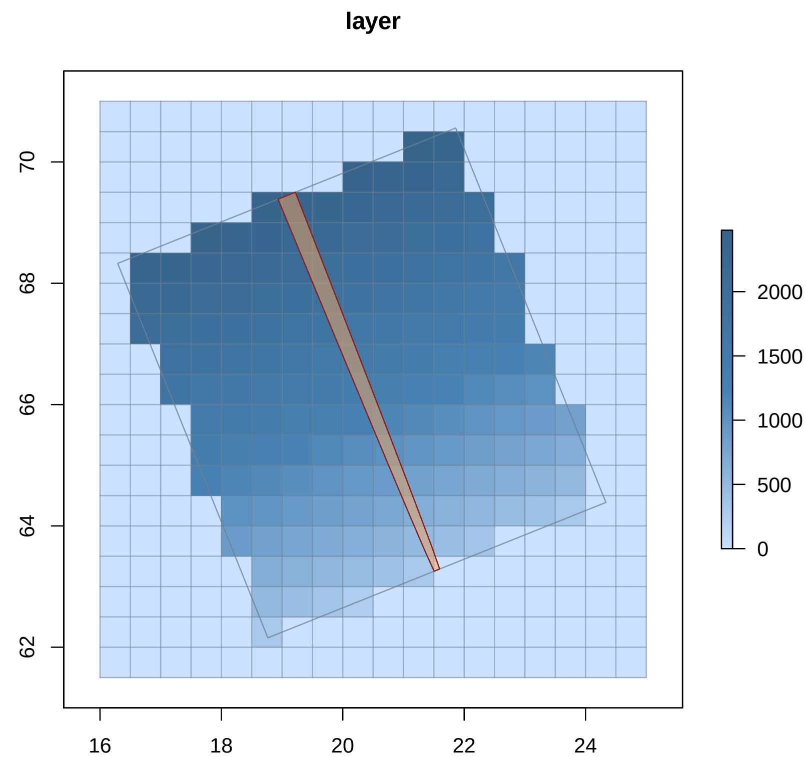

Figure 13: An "InclusionZoneGrid" object based on a "distanceLimitedMCIZ" object showing the variable height sampling surface for biomass.

the two. This flexibility can make things a bit messy, however. We again define several different protocols within this method. First, "canonical" DLPDS uses DLS for the truncated portion of the inclusion zone and canonical PDS for the section that is treated as a normal PDS sample.[4] In the second protocol, we substitute omnibus PDS for any section that is to be sampled with PDS, while DLMCS is used for the distance limited portion, and refer to this as "omnibus" DLPDS. Note that the difference is entirely based on the protocols for both the PDS and distance limited components (if any) for each log. Both components of the inclusion zone can, therefore, be either constant or variable depending on the PPS selection strategy (volume, surface area or coverage area) and the particular attribute we are estimating (refer to the last few sections for more information). Finally, there is a "hybrid" protocol, which uses canonical PDS and DLMCS, and was the original distance limited method introduced by Ducey et al. (2013). Again, this method is useful only for the design attribute in field application and is design-unbiased. The sampling surface will be flat for the PDS

[4]Of course, in field applications, this method is limited to sampling only for the PPS selection variable in the PDS component for the same reasons as described above.

component, but can be either flat or sloping for the DLMCS component. Details on each of the three methods can be found in Gove et al. (2013).

```
R> iz.dlpds = distanceLimitedPDSIZ(dlogs@logs$log.1, pds=pdsmet, dls=dlsMet)
R> (izgDLPDS = izGrid(iz.dlpds, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
a distance limited PDSIZ inclusion zone grid object
------------------------------------------------------------

InclusionZone class: distanceLimitedPDSIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 238
Cell dimensions: (nrows=17, ncol=14)
Grid cell values**...
  gridValues Freq
1          0  109
2       <NA>  129
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume Density Length surfaceArea coverageArea biomass carbon depth
Min.     100.0   276.3   1667        1532        487.5    2010   1005     1
1st Qu.  100.0   276.3   1667        1532        487.5    2010   1005     1
Median   100.0   276.3   2855        1848        588.1    2010   1005     1
Mean     104.0   370.5   2462        1744        554.9    2090   1045     1
3rd Qu.  112.1   561.6   2855        1848        588.1    2253   1126     1
Max.     112.1   561.6   2855        1848        588.1    2253   1126     1


  Encapulating bounding box...
   min max
x 16.0  23
y 62.5  71
```

```
R> plot(izgDLPDS, estimate='biomass', showPDSPart=TRUE)
```

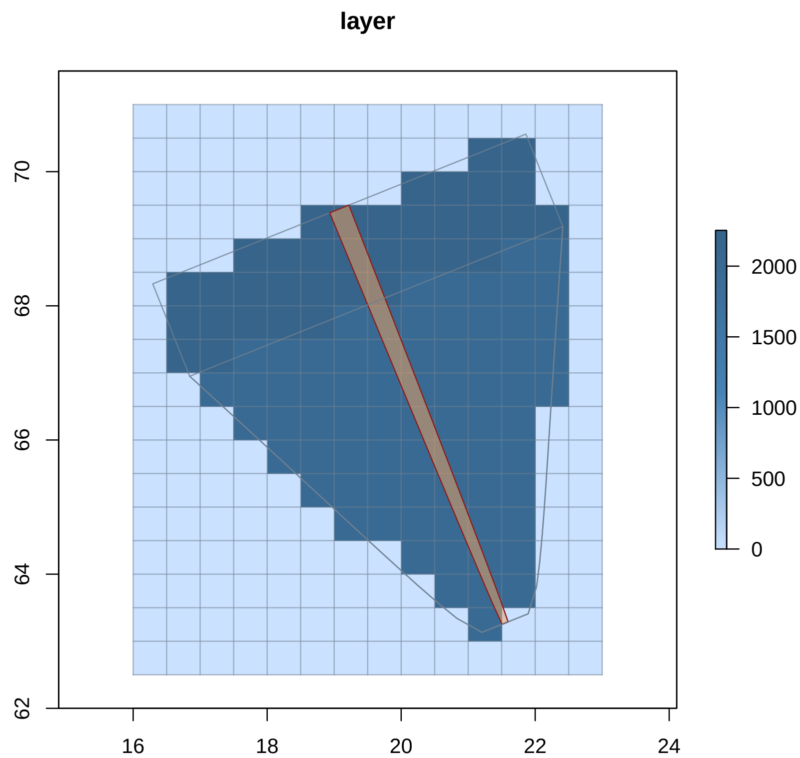

Figure 14: An "InclusionZoneGrid" object based on a "distanceLimitedPDSIZ" object showing the variable "stair step" height sampling surface for biomass.

Figure 14 shows[5] the two inclusion zones in the hybrid region. Since biomass is a scaled version of volume (the PPS selection variable), the PDS component surface is constant as in Figure 10. The DLS component is also constant, just as the surface generated in Figure 12. Now we are able to see why this is a hybrid scheme more clearly. Note from the object summary that the combination of the two types of inclusion zone objects makes all of the attributes variable in case *(iii)*, since both PDS and DLS will always produce constant height surfaces that will, in general, not be the same height, resulting in the entire surface resembling a "step" function.

[5]The log is randomly generated with each run of this document, and so differs with each creation, but the parameters are chosen such that it should show the two zones of case *(iii)*.

14 Example: The "omnibusDLPDSIZ" Class

Omnibus DLPDS is a Monte Carlo-based method. Again, this is similar to canonical DLPDS with the exception that omnibus PDS is employed within the PDS component of the inclusion zone, and DLMCS is used in the distance limited portion, if any. This method will be more appropriate for most field applications since one is able to estimate any attribute shown in Ducey et al. (2008). The resulting surface will be variable as described in the sections for the individual component methods above.

```
R> iz.odlpds = omnibusDLPDSIZ(dlogs@logs$log.1, pds=pdsmet, dls=dlsMet)
R> (izgODLPDS = izGrid(iz.odlpds, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
a distance limited PDSIZ inclusion zone grid object
------------------------------------------------------------

InclusionZone class: omnibusDLPDSIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 238
Cell dimensions: (nrows=17, ncol=14)
Grid cell values**...
  gridValues Freq
1          0  109
2       <NA>  129
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume Density Length surfaceArea coverageArea biomass carbon depth
Min.     100.0   162.8   1667        1451        461.8    2010   1005     1
1st Qu.  100.0   202.6   1667        1515        482.2    2010   1005     1
Median   100.0   295.7   1915        1587        505.2    2010   1005     1
Mean     103.8   367.3   2429        1732        551.2    2088   1044     1
3rd Qu.  105.0   561.6   2570        1797        572.1    2112   1056     1
Max.     123.3  1063.0  10983        3715       1182.5    2478   1239     1


  Encapulating bounding box...
   min max
x 16.0  23
```

```
y 62.5  71
```

```
R> plot(izgODLPDS, estimate='surfaceArea', showPDSPart=TRUE)
```

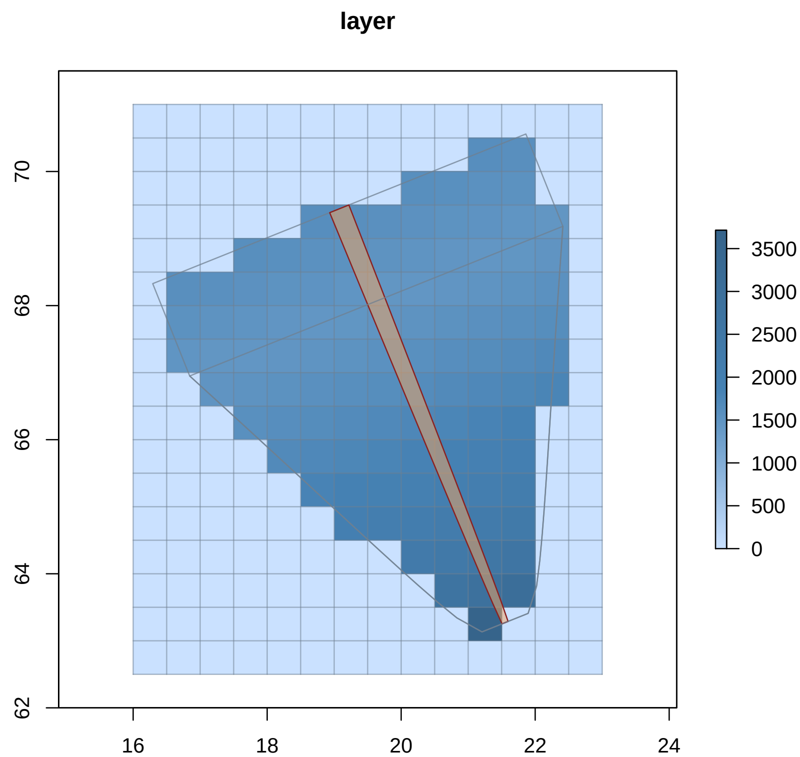

Figure 15: An "InclusionZoneGrid" object based on a "omnibusDLPDSIZ" object showing the variable height sampling surface for surface area.

It may be difficult to see in Figure 15, but the surface is actually slightly convex from butt to tip, because it varies in the reverse sense for each of the two components; i.e., larger near the butt for DLMCS, but larger near the tip for omnibus PDS.

15 Example: The "hybridDLPDSIZ" Class

Hybrid DLPDS is similar to canonical DLPDS with the exception that DLMCS is employed within the DLS component of the inclusion zone, if any. This is the original DLPDS method introduced by Ducey et al. (2013). The resulting surface will be flat or variable depending on the attribute, as described in the sections for the individual component methods above.

```
R> iz.hdlpds = hybridDLPDSIZ(dlogs@logs$log.1, pds=pdsmet, dls=dlsMet)
R> (izgHDLPDS = izGrid(iz.hdlpds, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
a distance limited PDSIZ inclusion zone grid object
------------------------------------------------------------

InclusionZone class: hybridDLPDSIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 238
Cell dimensions: (nrows=17, ncol=14)
Grid cell values**...
  gridValues Freq
1          0  109
2       <NA>  129
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume Density Length surfaceArea coverageArea biomass carbon depth
Min.     100.0   276.3   1667        1451        461.8    2010   1005     1
1st Qu.  100.0   276.3   1667        1577        501.9    2010   1005     1
Median   100.0   276.3   2855        1848        588.1    2010   1005     1
Mean     103.8   370.5   2462        1742        554.6    2088   1044     1
3rd Qu.  105.0   561.6   2855        1848        588.1    2112   1056     1
Max.     123.3   561.6   2855        1848        588.1    2478   1239     1


  Encapulating bounding box...
   min max
x 16.0  23
y 62.5  71
```

```
R> plot(izgHDLPDS, estimate='surfaceArea', showPDSPart=TRUE)
```

Again, it may be difficult to tell from Figure 16, but the surface slopes downward from the butt of the log to the transition point in the DLMCS section of the inclusion zone, and is constant in the

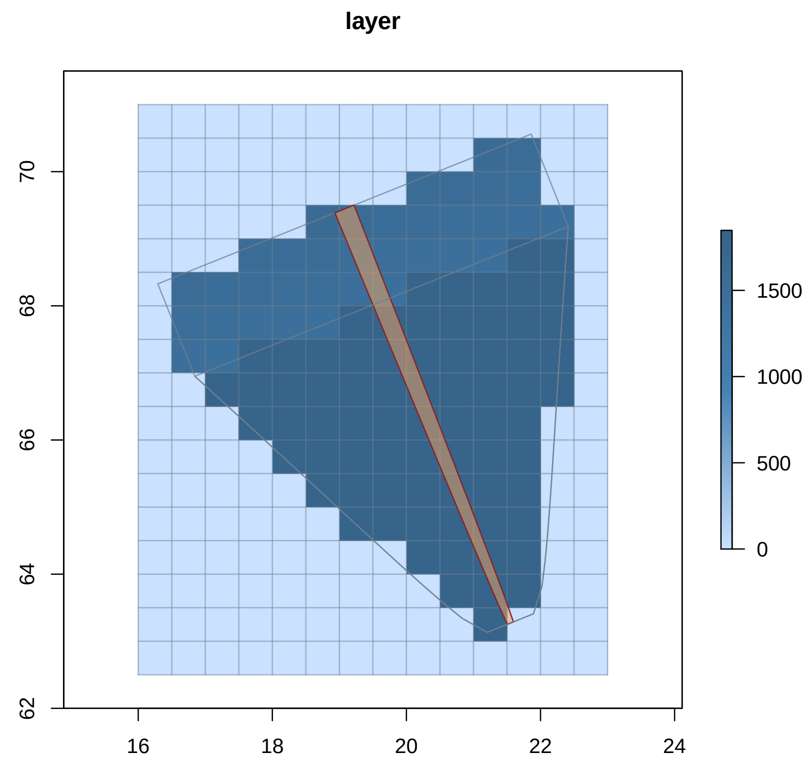

Figure 16: An "InclusionZoneGrid" object based on a "hybridDLPDSIZ" object showing the composite height sampling surface for surface area when sampling with PP volume.

PDS section.

16 Example: The "circularPlotIZ" Class

This example is for standing trees, and the example shows how the classic fixed-area circular plot method works for the "InclusionZoneGrid" class.

```
R> sttr = standingTrees(1, btr, dbh = c(10,20), topDiams=c(0,0),
+                        solidType=c(1.5,3))
R> cpiz = circularPlotIZ(sttr@trees$tree.1, plotRadius=4)
R> (cpIZG = izGrid(cpiz, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
circularPlotIZ inclusion zone grid object
------------------------------------------------------------

InclusionZone class: circularPlotIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 324
Cell dimensions: (nrows=18, ncol=18)
Grid cell values**...
  gridValues Freq
1          0  198
2       <NA>  126
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume Density basalArea surfaceArea depth
Min.     20.06   198.9     4.264       622.5     1
1st Qu.  20.06   198.9     4.264       622.5     1
Median   20.06   198.9     4.264       622.5     1
Mean     20.06   198.9     4.264       622.5     1
3rd Qu.  20.06   198.9     4.264       622.5     1
Max.     20.06   198.9     4.264       622.5     1
--Note: either biomass or carbon (or both) had all NAs because no conversion
        factor was supplied, these columns have been deleted above.

  Encapulating bounding box...
   min  max
x 33.5 42.5
y 59.5 68.5


R> plot(cpIZG, estimate='surfaceArea')
```

17 Example: The "horizontalPointIZ" Class

This section presents an example based on horizontal point sampling for a standing tree. Note that it is very similar to the previous example for sampling with fixed-area circular plots and indeed uses the same methods thanks to class inheritance.

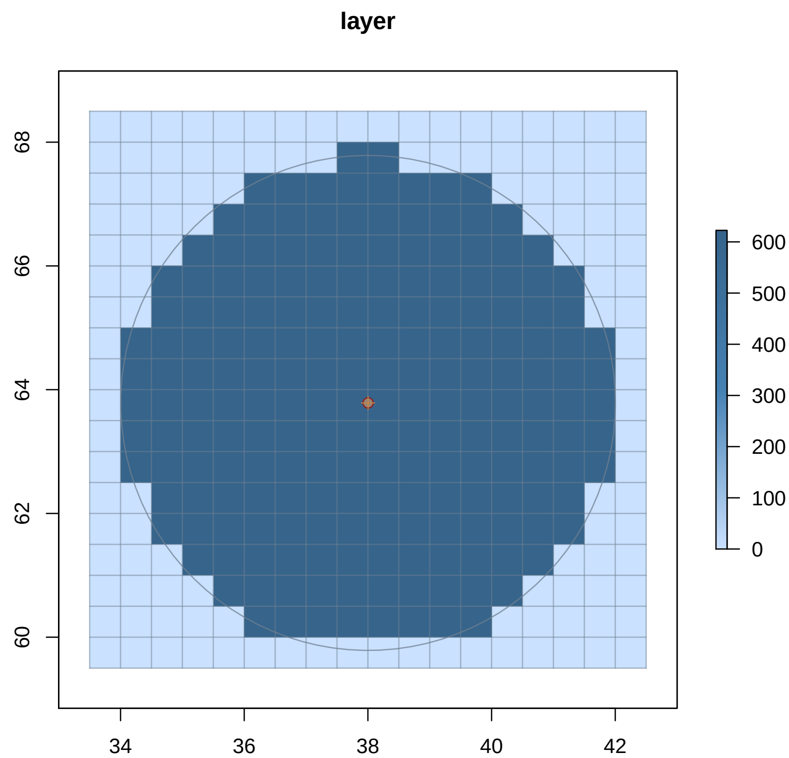

Figure 17: An “InclusionZoneGrid” object based on a “circularPlotIZ” object.

```
R> aGauge = angleGauge(baf=3) #metric
R> hpiz = horizontalPointIZ(sttr@trees$tree.1, angleGauge=aGauge)
R> (hpsIZG = izGrid(hpiz, btr))


Object of class: InclusionZoneGrid
------------------------------------------------------------
circularPlotIZ inclusion zone grid object
------------------------------------------------------------

InclusionZone class: horizontalPointIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 462
Cell dimensions: (nrows=21, ncol=22)
```

```
Grid cell values**...
  gridValues Freq
1          0  282
2       <NA>  180
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume Density basalArea surfaceArea depth
Min.     14.11     140         3         438     1
1st Qu.  14.11     140         3         438     1
Median   14.11     140         3         438     1
Mean     14.11     140         3         438     1
3rd Qu.  14.11     140         3         438     1
Max.     14.11     140         3         438     1
--Note: either biomass or carbon (or both) had all NAs because no conversion
        factor was supplied, these columns have been deleted above.

  Encapulating bounding box...
   min  max
x 32.5 43.5
y 58.5 69.0


R> plot(hpsIZG, estimate='basalArea')
```

18 Example: The "horizontalPointMonteCarloSamplingIZ" Class

One constructor using the class union "horizontalPointMonteCarloSamplingIZ" will handle all three of the Monte Carlo subsampling methods available. These include crude Monte Carlo (CMC) sampling, importance sampling (IS) and control variate sampling (CV). Antithetic variants of each may also be requested. The various subsampling methods are applied at each of the grid cells within the tree's inclusion zone. All of the options for the desired subsampling method *must* have already been set up in the call to the associated "InclusionZone" constructor. The options are stored within this object, and used here. Please see Gove (2012) and Gove (2013a), as well as the on-line help for more details. Please remember that only volume will be estimated with any of the Monte Carlo subsampling methods.

In each of the examples, note that the surface is random in nature, with no discernible shape or pattern. These may be contrasted, for example, to similar methods derived from critical height sampling (§ 20–23), where randomness enters through the position of the sample point (grid cell), forming a replicable shape each time.

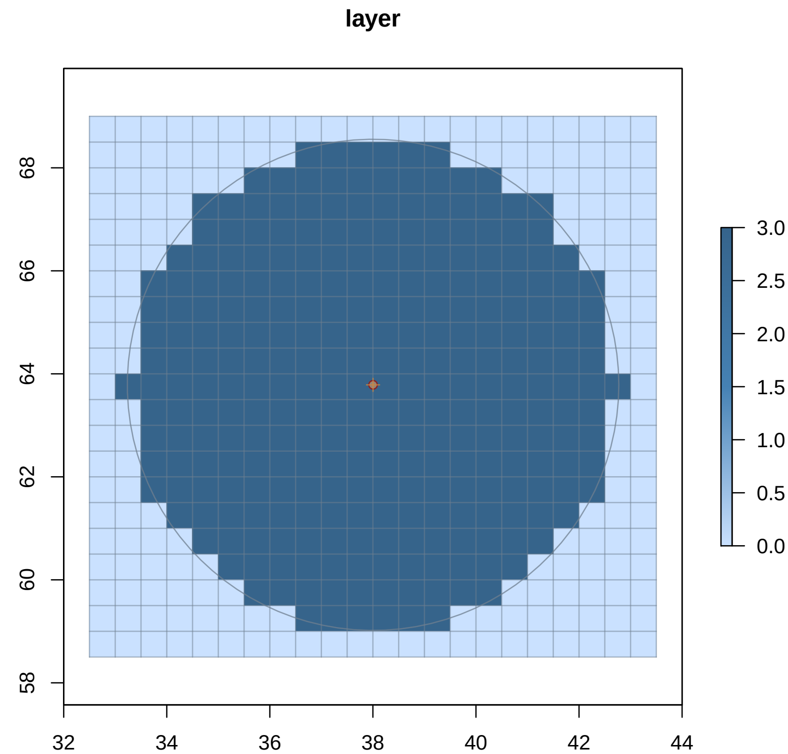

Figure 18: An "InclusionZoneGrid" object based on a "horizontalPointIZ" object.

Additionally, note that the individual "MonteCarloSampling" subclass objects associated with each grid cell are not stored. Only the respective volume estimates themselves are stored as usual in the `data` slot of the object. The information in the `iz` slot is the inclusion zone object passed to it as usual, with the record of options desired. One could recover the individual "MonteCarloSampling" subclass objects by setting the random number seed before a run, and sequentially running though the grid cells, applying the correct Monte Carlo method, as is done in the constructor. Normally, there should be little reason for this extra information.

18.1 Example: The "horizontalPointCMCIZ" Class

This section presents an example based on horizontal point sampling for a standing tree with CMC sampling conducted on the tree at each individual grid point. Note in the following example that antithetic CMC has been applied with a default of one pair of points per grid cell for subsampling, and that this has been determined when constructing the "horizontalPointCMCIZ" class itself...

```
R> hpcmciz = horizontalPointCMCIZ(sttr@trees$tree.1, angleGauge=aGauge,
+                                 antithetic=TRUE)
R> hpcmciz@antithetic
```

```
[1] TRUE
```

```
R> (hpscmcIZG = izGrid(hpcmciz, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
horizontalPoint w/ MC subsampling inclusion zone grid object
------------------------------------------------------------

InclusionZone class: horizontalPointCMCIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 462
Cell dimensions: (nrows=21, ncol=22)
Grid cell values**...
  gridValues Freq
1          0  282
2       <NA>  180
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume depth
Min.     11.85     1
1st Qu.  12.28     1
Median   13.74     1
Mean     14.24     1
3rd Qu.  15.86     1
Max.     18.70     1
--Note: either biomass or carbon (or both) had all NAs because no conversion
        factor was supplied, these columns have been deleted above.
        Other columns with all NAs have also been removed.

  Encapulating bounding box...
   min  max
x 32.5 43.5
y 58.5 69.0
```

```
R> plot(hpscmcIZG)
```

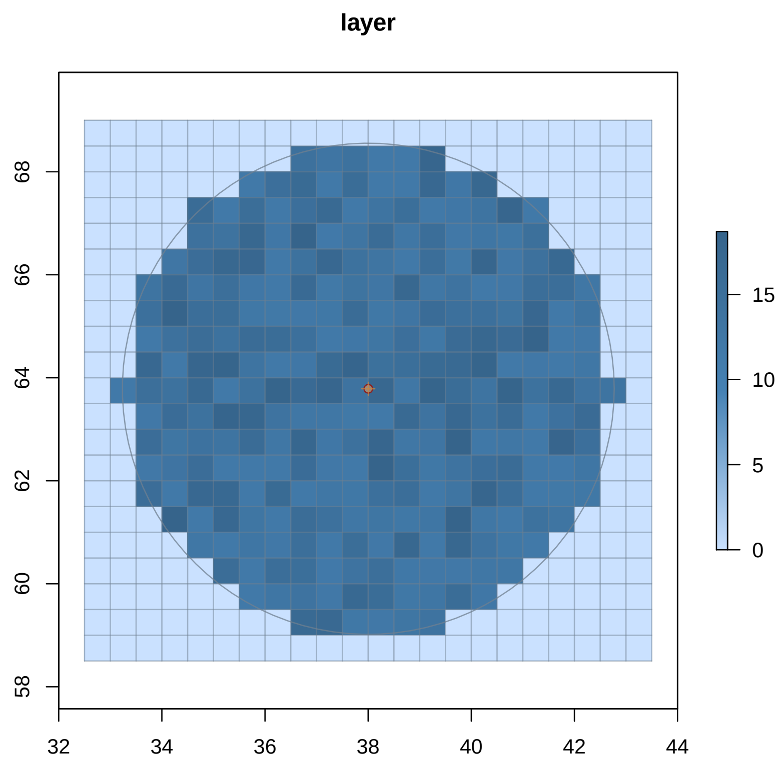

Figure 19: An "InclusionZoneGrid" object based on a "horizontalPointCMCIZ" object.

18.2 Example: The "horizontalPointISIZ" Class

This section presents an example based on horizontal point sampling for a standing tree with importance sampling conducted on the tree at each individual grid point. Note in the following example that when constructing the "horizontalPointISIZ" class, we specify a proxy and desired arguments to that proxy...

```
R> hpisiz = horizontalPointISIZ(sttr@trees$tree.1, angleGauge=aGauge,
+                               proxy='wbProxy', solidTypeProxy = 0.9)
R> c(hpisiz@mcsObj@proxy, hpisiz@mcsObj@userArgs)
```

```
[[1]]
[1] "wbProxy"

$solidTypeProxy
[1] 0.9
```

```
R> (hpsisIZG = izGrid(hpisiz, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
horizontalPoint w/ MC subsampling inclusion zone grid object
------------------------------------------------------------

InclusionZone class: horizontalPointISIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 462
Cell dimensions: (nrows=21, ncol=22)
Grid cell values**...
  gridValues Freq
1          0  282
2       <NA>  180
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
         volume depth
Min.     0.5037     1
1st Qu. 13.8750     1
Median  14.0770     1
Mean    14.0700     1
3rd Qu. 14.3560     1
Max.    14.6370     1
--Note: either biomass or carbon (or both) had all NAs because no conversion
        factor was supplied, these columns have been deleted above.
        Other columns with all NAs have also been removed.

  Encapulating bounding box...
   min  max
x 32.5 43.5
y 58.5 69.0
```

```
R> plot(hpsisIZG)
```

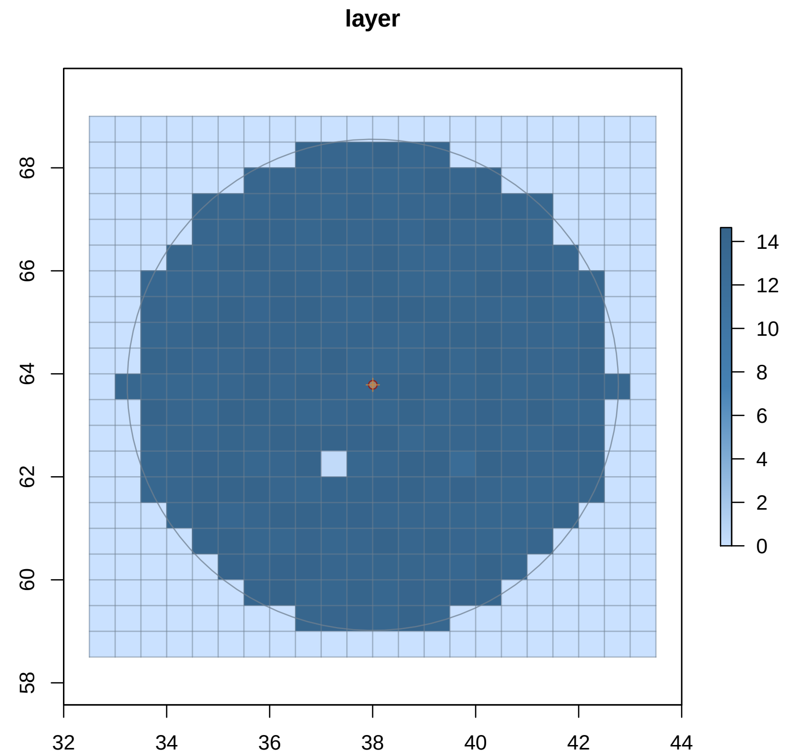

Figure 20: An "InclusionZoneGrid" object based on a "horizontalPointISIZ" object.

18.3 Example: The "horizontalPointCVIZ" Class

This section presents an example based on horizontal point sampling for a standing tree with control variate sampling conducted on the tree at each individual grid point. Again, in the following example we specify a proxy and desired arguments to that proxy when constructing the "horizontalPointISIZ" class, ...

```
R> hpcviz = horizontalPointCVIZ(sttr@trees$tree.1, angleGauge=aGauge,
+                                proxy='wbProxy', solidTypeProxy = 0.9)
R> c(hpcviz@mcsObj@proxy, hpcviz@mcsObj@userArgs)
```

```
[[1]]
[1] "wbProxy"
```

```
$solidTypeProxy
[1] 0.9
```

```
R> (hpscvIZG = izGrid(hpcviz, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
horizontalPoint w/ MC subsampling inclusion zone grid object
------------------------------------------------------------

InclusionZone class: horizontalPointCVIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 462
Cell dimensions: (nrows=21, ncol=22)
Grid cell values**...
  gridValues Freq
1          0  282
2       <NA>  180
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume depth
Min.     13.98     1
1st Qu.  14.27     1
Median   14.58     1
Mean     14.49     1
3rd Qu.  14.74     1
Max.     14.83     1
--Note: either biomass or carbon (or both) had all NAs because no conversion
        factor was supplied, these columns have been deleted above.
        Other columns with all NAs have also been removed.

  Encapulating bounding box...
   min  max
x 32.5 43.5
y 58.5 69.0
```

```
R> plot(hpscvIZG)
```

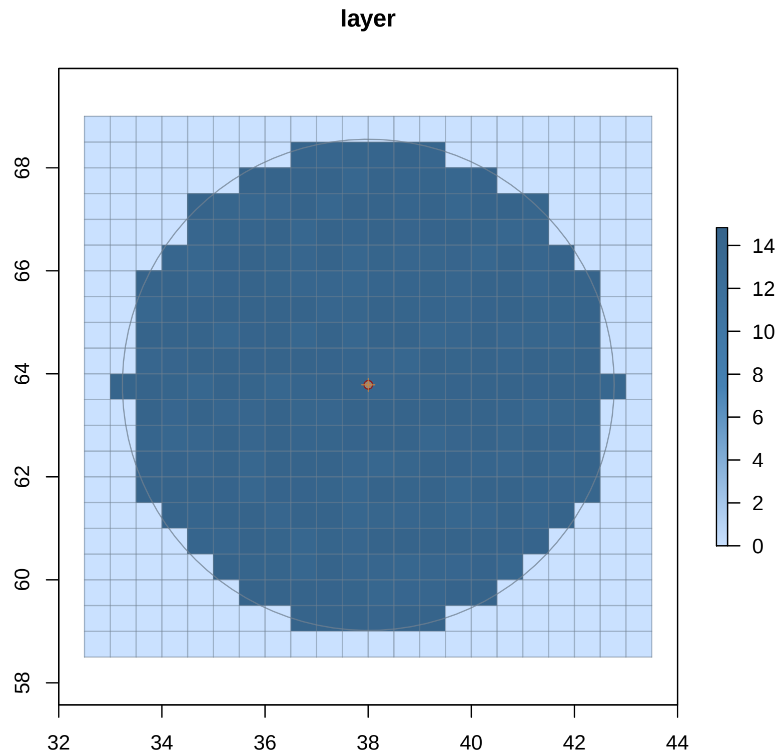

Figure 21: An "InclusionZoneGrid" object based on a "horizontalPointCVIZ" object.

19 Example: The "horizontalLineIZ" Class

This section presents an example based on horizontal line sampling for a standing tree.

```
R> aGauge = angleGauge(baf=5) #metric
R> hliz = horizontalLineIZ(sttr@trees$tree.1, angleGauge=aGauge, lineLength = 20,
+                                orientation = 35)
R> (hlsIZG = izGrid(hliz, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
horizontalLineIZ inclusion zone grid object
------------------------------------------------------------
```

```
InclusionZone class: horizontalLineIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 1634
Cell dimensions: (nrows=43, ncol=38)
Grid cell values**...
  gridValues Freq
1          0  589
2       <NA> 1045
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume Density basalArea surfaceArea depth
Min.     6.824   67.68     1.451       211.8     1
1st Qu.  6.824   67.68     1.451       211.8     1
Median   6.824   67.68     1.451       211.8     1
Mean     6.824   67.68     1.451       211.8     1
3rd Qu.  6.824   67.68     1.451       211.8     1
Max.     6.824   67.68     1.451       211.8     1
--Note: either biomass or carbon (or both) had all NAs because no conversion
        factor was supplied, these columns have been deleted above.

  Encapulating bounding box...
   min  max
x 28.5 47.5
y 53.0 74.5
```

```
R> plot(hlsIZG, estimate='basalArea')
```

20 Example: The "criticalHeightIZ" Class

This section presents an example based on critical height sampling (CHS) for a standing tree. Note the similarity to the "horizontalPointIZ" example; but it only is used to estimate volume...

```
R> aGauge = angleGauge(baf=3) #metric
R> chsiz = criticalHeightIZ(sttr@trees$tree.1, angleGauge=aGauge,
+                            referenceHeight='dbh')
R> (chsIZG = izGrid(chsiz, btr))
```

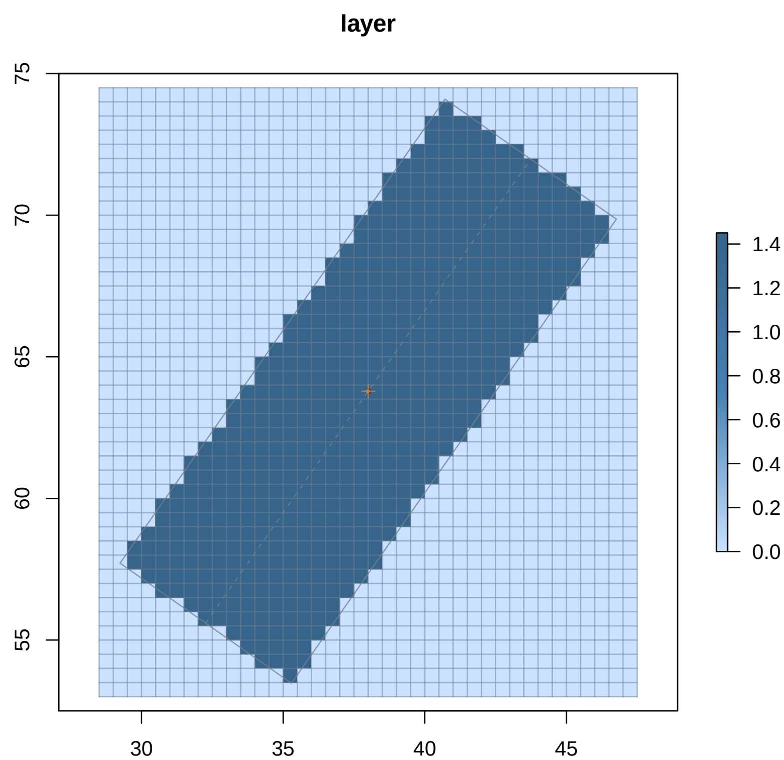

Figure 22: An "InclusionZoneGrid" object based on a "horizontalLineIZ" object.

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
criticalHeightIZ inclusion zone grid object
------------------------------------------------------------

InclusionZone class: criticalHeightIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 462
Cell dimensions: (nrows=21, ncol=22)
Grid cell values**...
  gridValues Freq
1          0  282
2       <NA>  180
**Note: data slot values get swapped with zero-valued grid cells as necessary.
```

```
Per unit area estimates in the data slot (for cells inside IZ only)...
        volume depth
Min.     4.213     1
1st Qu.  8.295     1
Median  12.615     1
Mean    13.652     1
3rd Qu. 18.591     1
Max.    28.492     1
--Note: either biomass or carbon (or both) had all NAs because no conversion
        factor was supplied, these columns have been deleted above.
        Other columns with all NAs have also been removed.

  Encapulating bounding box...
   min  max
x 32.5 43.5
y 58.5 69.0
```

```
R> plot(chsIZG)
```

As with a number of the other sampling methods discussed, CHS generates a variable-height surface within the inclusion zone. This is the first sampling method for standing trees that we have encountered that has this property.

21 Example: The "importanceCHSIZ" Class

Importance sampling can be coupled with CHS, producing an improved estimator (Lynch and Gove, 2013). Note in the results below the differences in the surface between importance critical height sampling (ICHS) and CHS for volume...

```
R> ichsiz = importanceCHSIZ(sttr@trees$tree.1, angleGauge=aGauge,
+                            referenceHeight='dbh')
R> (ichsIZG = izGrid(ichsiz, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
importanceCHSIZ-based inclusion zone grid object
------------------------------------------------------------
```

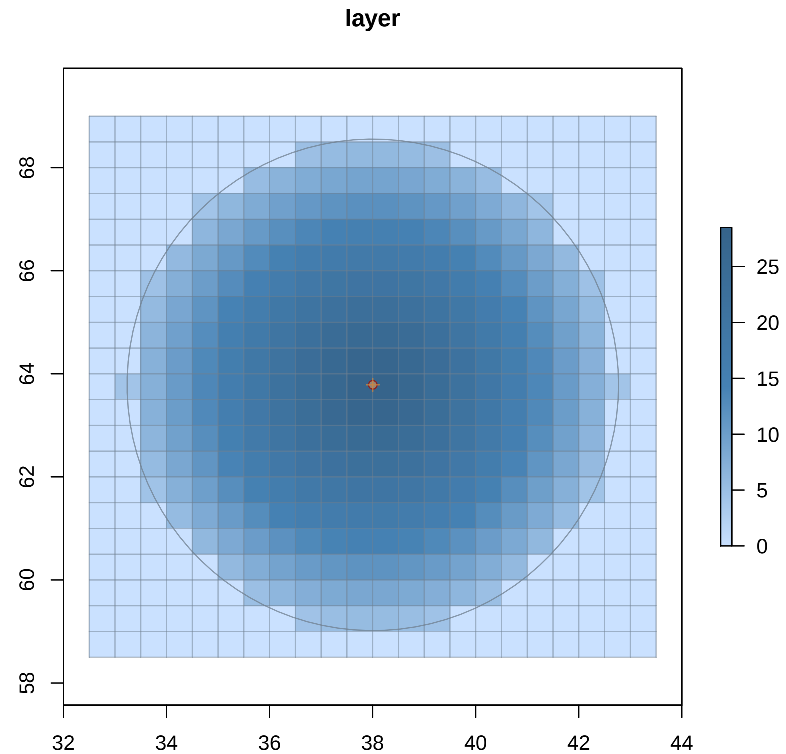

Figure 23: An “InclusionZoneGrid” object based on a “criticalHeightIZ” object.

```
InclusionZone class: importanceCHSIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 462
Cell dimensions: (nrows=21, ncol=22)
Grid cell values**...
  gridValues Freq
1          0  282
2       <NA>  180
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume depth
Min.     11.78     1
1st Qu.  12.99     1
```

```
Median   13.94     1
Mean     14.14     1
3rd Qu.  15.21     1
Max.     17.62     1
--Note: either biomass or carbon (or both) had all NAs because no conversion
        factor was supplied, these columns have been deleted above.
        Other columns with all NAs have also been removed.

  Encapulating bounding box...
   min  max
x 32.5 43.5
y 58.5 69.0
```

```
R> plot(ichsIZG)
```

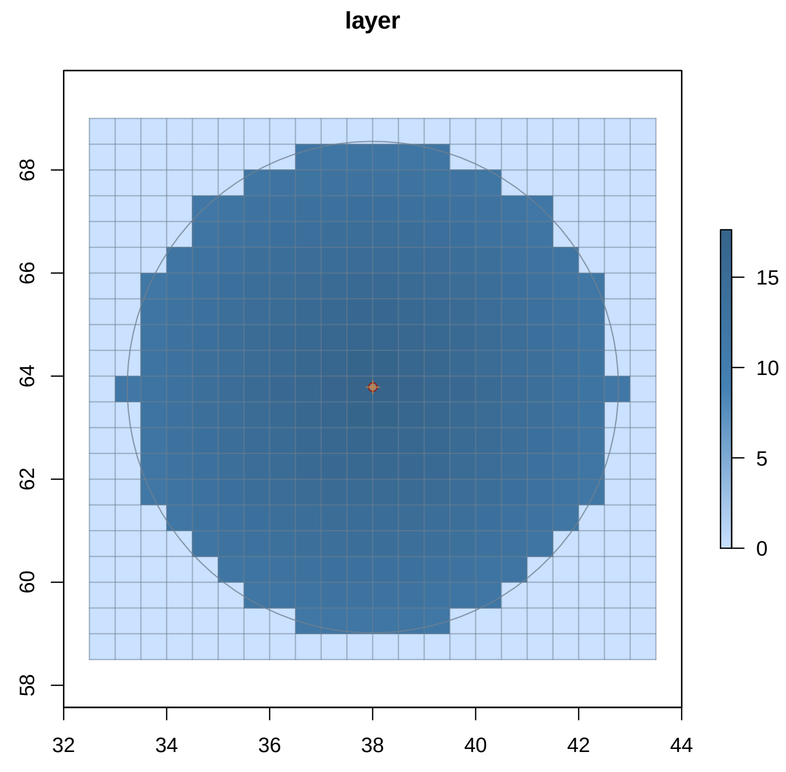

Figure 24: An "InclusionZoneGrid" object based on a "importanceCHSIZ" object.

If you view the surface in Figure 24 using `plot3D` (§ 25), you will see the slight unevenness that is reported in the summary for the object above—it is not completely flat even though it appears so in Figure 24. However, it is much closer to level than CHS, and hence a more efficient method variance-wise.

22 Example: The "antitheticICHSIZ" Class

An antithetic variant to ICHS moves the sampling point on the bole to facilitate measurement (Lynch and Gove, 2013). Again note in the results below the differences in the surface between antithetic importance critical height sampling (AICHS), ICHS, and CHS for volume...

```
R> aichsiz = antitheticICHSIZ(sttr@trees$tree.1, angleGauge=aGauge,
+                             referenceHeight='dbh')
R> (aichsIZG = izGrid(aichsiz, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
importanceCHSIZ-based inclusion zone grid object
------------------------------------------------------------

InclusionZone class: antitheticICHSIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 462
Cell dimensions: (nrows=21, ncol=22)
Grid cell values**...
  gridValues Freq
1          0  282
2       <NA>  180
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume depth
Min.     12.91     1
1st Qu.  13.52     1
Median   13.95     1
Mean     14.09     1
3rd Qu.  14.52     1
Max.     16.31     1
```

```
--Note: either biomass or carbon (or both) had all NAs because no conversion
        factor was supplied, these columns have been deleted above.
        Other columns with all NAs have also been removed.

  Encapulating bounding box...
   min  max
x 32.5 43.5
y 58.5 69.0
```

```
R> plot(aichsIZG)
```

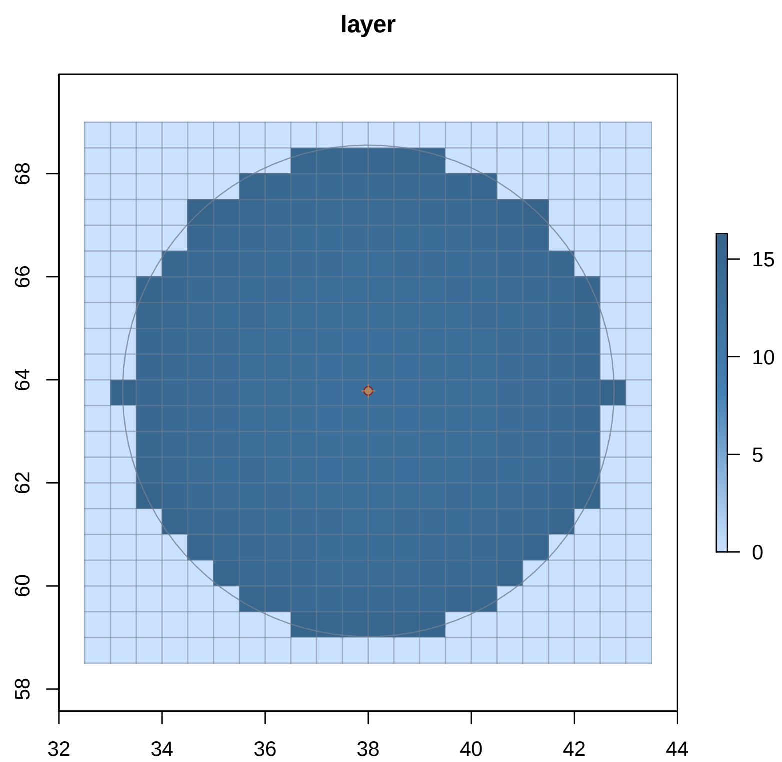

Figure 25: An “InclusionZoneGrid” object based on a “antitheticICHSIZ” object.

23 Example: The “pairedAICHSIZ” Class

The last variant that is related to CHS is paired antithetic importance critical height sampling (PAICHS). It combines two measurements on the bole in the estimator (Lynch and Gove, 2013). Again note in the results below the differences in the surface between PAICHS, AICHS, ICHS, and CHS for volume...

```
R> paichsiz = pairedAICHSIZ(sttr@trees$tree.1, angleGauge=aGauge,
+                            referenceHeight='dbh')
R> (paichsIZG = izGrid(paichsiz, btr))
```

```
Object of class: InclusionZoneGrid
------------------------------------------------------------
importanceCHSIZ-based inclusion zone grid object
------------------------------------------------------------

InclusionZone class: pairedAICHSIZ
  units of measurement:  metric

Grid class: RasterLayer
Number of grid cells = 462
Cell dimensions: (nrows=21, ncol=22)
Grid cell values**...
  gridValues Freq
1          0  282
2       <NA>  180
**Note: data slot values get swapped with zero-valued grid cells as necessary.

Per unit area estimates in the data slot (for cells inside IZ only)...
        volume depth
Min.     13.38     1
1st Qu.  13.77     1
Median   14.33     1
Mean     14.12     1
3rd Qu.  14.40     1
Max.     14.69     1
--Note: either biomass or carbon (or both) had all NAs because no conversion
        factor was supplied, these columns have been deleted above.
        Other columns with all NAs have also been removed.

  Encapulating bounding box...
   min  max
```

```
x 32.5 43.5
y 58.5 69.0
```

```
R> plot(paichsIZG)
```

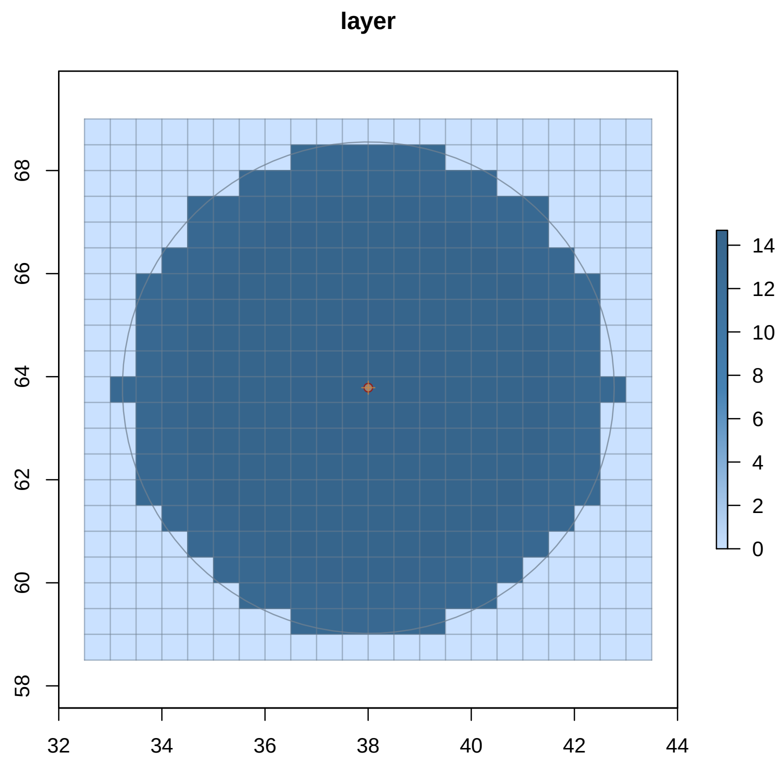

Figure 26: An "InclusionZoneGrid" object based on a "pairedAICHSIZ" object.

24 The "mirageInclusionZoneGrid" Class

This class is a subclass of the "InclusionZoneGrid" class, as shown in Figure 1. Just as we required a new "Tract" subclass to handle the mirage method, we also require a new subclass to handle the building of a sampling surface when one wants to apply the mirage method. This also necessitates a different constructor function, `izGridMirage`, to create these objects.

The class is defined with the slots...

```
R> showClass('mirageInclusionZoneGrid')
```

```
Class "mirageInclusionZoneGrid" [package "sampSurf"]

Slots:

Name:         slopOver   north.polygon      north.grid   south.polygon
Class:         logical          SPNULL          RLNULL          SPNULL

Name:       south.grid    east.polygon       east.grid    west.polygon
Class:          RLNULL          SPNULL          RLNULL          SPNULL

Name:        west.grid izGrid.extended     description              iz
Class:          RLNULL         izgNULL       character   InclusionZone

Name:             grid            data            bbox
Class:     RasterLayer      data.frame          matrix

Extends:
Class "InclusionZoneGrid", directly
Class "izgNULL", by class "InclusionZoneGrid", distance 2
```

24.1 Class slots

There are several new slots that have been added to this subclass, these are defined below. The remaining slots for the superclass are defined in § 2.

- *slopover*: This slot contains a `logical` vector of length four which flags where there is any boundary overlap in the cardinal `c('north','south','east','west')` directions. The value corresponding to each direction is `TRUE` if so, `FALSE` otherwise. Note that one can easily check this vector to determine whether the associated `*.polygon` and `*.grid` slots are non-`NULL`.
- *north.polygon*: This slot contains an object of class "SPNULL". If `slopover['north']`, then this slot holds the external portion of the inclusion zone polygon ("SpatialPolygons" object) due to boundary slopover on the north side. Otherwise, it is `NULL`.
- *north.grid*: This slot contains an object of class "RLNULL". If `slopover['north']`, then this slot holds the external portion of the extended grid ("RasterLayer" object) on the north side. Otherwise, it is `NULL`.
- *south.polygon*: The same as the `north.poylgon` slot, but for south.
- *south.grid*: The same as the `north.grid` slot, but for south.

- *east.polygon*: The same as the `north.poylgon` slot, but for east.
- *east.grid*: The same as the `north.grid` slot, but for east.
- *west.polygon*: The same as the `north.poylgon` slot, but for west.
- *west.grid*: The same as the `north.grid` slot, but for west.
- *grid.extended*: This slot contains an object of class "izgNULL". This is an "InclusionZoneGrid" object showing the extended grid (the underlying grid is padded to extend beyond the tract boundaries if there is slopover) that encompases any slopover regions external to the boundary, where applicable. Otherwise, it is `NULL` if there is no slopover on any side.

Note that in the case of the `grid.extended` slot, the contents of the object will be different depending on what is passed to the `izGridMirage` constructor in the `truncateOverlap` argument.[6] If one uses `truncateOverlap = TRUE`, then the cells external to the tract will be assigned the background value. If `truncateOverlap = FALSE`, then the cells that fall within the inclusion zone, but are external to the tract, will retain their estimates. This is often desirable to be able to see how these values were then folded back into the tract. Do keep in mind that this full extended object with external estimates included should not be used for any estimation as there will be too much attribute density associated with the full inclusion zone. More information, as well as examples, can be found in the vignette Gove (2013b).

25 Using `plot3D`

The `plot3D` generic was extended to handle objects of class "InclusionZoneGrid". Its use is simple, just remember to use the `estimate` argument to specify the desired surface attribute to be rendered...

```
R> plot3D(izgODLPDS, estimate='surfaceArea')
```

Please be aware that the `plot3D` method adjusts the x and y axes by default to scale the image for presentation. This scaling is very helpful for viewing, but tends to distort the two axes relative to the height of the surface. To view the true units and hence see what the surface looks like without any scaling, use the `adjust=FALSE` argument to display the surface in `plot3D`.

[6] See `methods?izGridMirage` for details.

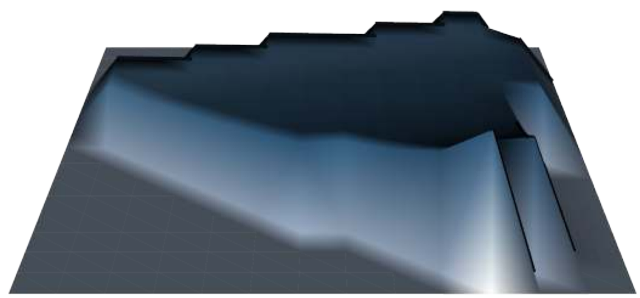

Figure 27: Representation of the sampling surface (via an "InclusionZoneGrid" object) for a single log sampled with "omnibusDLPDS".

References

M. J. Ducey, M. S. Williams, J. H. Gove, and H. T. Valentine. Simultaneous unbiased estimates of multiple downed wood attributes in perpendicular distance sampling. *Canadian Journal of Forest Research*, 38:2044–2051, 2008. 19, 21, 30

M. J. Ducey, M. S. Williams, J. H. Gove, S. Roberge, and R. S. Kenning. Distance limited perpendicular distance sampling for coarse woody material: Theory and field results. *Forestry*, 86: 119–128, 2013. 27, 32

J. H. Gove. *The "InclusionZone" Class*, 2012. URL `http://CRAN.R-project.org/package=sampSurf`. `sampSurf` package vignette. 36

J. H. Gove. *Monte Carlo Sampling Methods in* ***sampSurf***, 2013a. URL `http://CRAN.R-project.org/package=sampSurf`. `sampSurf` package vignette. 36

J. H. Gove. *The Mirage Method in `sampSurf`*, 2013b. URL http://sampsurf.r-forge.r-project.org/. `sampSurf` vignette, only available on R-Forge. 54

J. H. Gove. *The "lapSurf" Class*, 2013c. URL http://CRAN.R-project.org/package=sampSurf. `sampSurf` package vignette. 4

J. H. Gove and P. C. Van Deusen. On fixed-area plot sampling for downed coarse woody debris. *Forestry*, 84(2):109–117, 2011. 9, 13, 14

J. H. Gove, A. Ringvall, G. Ståhl, and M. J. Ducey. Point relascope sampling of downed coarse woody debris. *Canadian Journal of Forest Research*, 29(11):1718–1726, 1999. 18

J. H. Gove, M. J. Ducey, A. Ringvall, and G. Ståhl. Point relascope sampling: A new way to assess down coarse woody debris. *Journal of Forestry*, 4:4–11, 2001. 18

J. H. Gove, M. J. Ducey, and H. T. Valentine. A distance limited method for sampling downed coarse woody debris. *Forest Ecology and Management*, 282:53–62, 2012. 23

J. H. Gove, M. J. Ducey, H. T. Valentine, and M. S. Williams. A comprehensive comparison of the perpendicular distance method for sampling downed coarse woody debris. *Forestry*, 86:129–143, 2013. 28

T. B. Lynch and J. H. Gove. An antithetic variate to facilitate upper-stem height measurements for critical height sampling with importance sampling. *Canadian Journal of Forest Research*, 43: 1151–1161, 2013. 46, 49, 51

M. S. Williams and J. H. Gove. Perpendicular distance sampling: an alternative method for sampling downed coarse woody debris. *Canadian Journal of Forest Research*, 33:1564–1579, 2003. 19

M. S. Williams, M. J. Ducey, and J. H. Gove. Assessing surface area of coarse woody debris with line intersect and perpendicular distance sampling. *Canadian Journal of Forest Research*, 35: 949–960, 2005. 19