# The "Tract" Class

Jeffrey H. Gove*
Research Forester
USDA Forest Service
Northern Research Station
271 Mast Road
Durham, New Hampshire 03824 USA
e-mail: jgove@fs.fed.us or e-mail: jhgove@unh.edu

Thursday 15th November, 2012
3:05pm

## Contents



## 1 Introduction

This class forms the basis for the actual tract on which the areal sampling is conducted. It is simple in concept, being a rectangular area with given cell resolution in which the cells themselves are always square. This last limitation was imposed to make things simple (one could always define another class that would allow rectangular cells.) Underneath, as we shall soon see, the class is actually very rich in functionality because it is a subclass of "RasterLayer" in the `raster` package.

Subclasses at the moment include only the "bufferedTract" class, but as we see in the diagram below (Figure 1), others are planned.

*Phone: (603) 868-7667; Fax: (603) 868-7604.

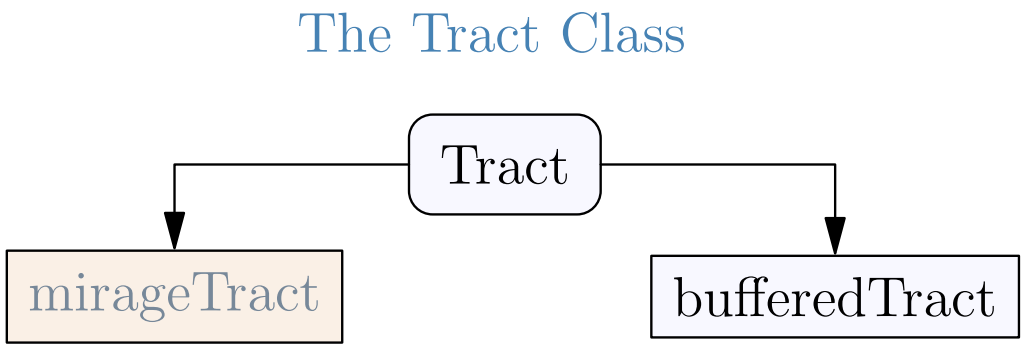


Figure 1: An overview of the "Tract" class, including a potential "mirageTract" subclass for handling the 'mirage' method of boundary overlap.

## 2 The "Tract" Class

The class itself is quite basic; there are only two slots that have been added to the "RasterLayer" class—these are itemized below...

```
R> getClass('Tract')


Class "Tract" [package "sampSurf"]

Slots:

Name:       description            units             area             file
Class:        character        character          numeric       .RasterFile

Name:              data           legend          history            title
Class: .SingleLayerData    .RasterLegend             list        character

Name:            extent          rotated         rotation             ncols
Class:           Extent          logical        .Rotation          integer

Name:             nrows              crs                z
Class:          integer              CRS             list

Extends:
```

```
Class "RasterLayer", directly
Class "Raster", by class "RasterLayer", distance 2
Class "BasicRaster", by class "RasterLayer", distance 3

Known Subclasses: "bufferedTract"
```

## 2.1 Class slots

- *description*: Some descriptive text about this class.
- *units*: A character string specifying the units of measure. Legal values are "English" and "metric."
- *area*: The tract area in either square meters or square feet.

However, if we look at the actual class it was derived from, we see much more detail available in the listing of slots above. There are many methods associated with "RasterLayer" objects. These are covered in the vignettes and documentation for that class. All of them are available for use on an object of class "Tract" or any of its subclasses.

## 2.2 Object creation

The "Tract" class has more than one constructor. The constructors are differentiated in a call to `Tract` by their signatures. The single signature argument in each case is the first argument, called `obj`.

### 2.2.1 GridTopology-like constructor

The first constructor allows for tract dimensions in terms of cell resolution, origin and extents in both $x$ and $y$ directions the way one would specify a "GridTopology" object in the `sp` package. The signature argument is of class "missing" in this case; therefore, the first three arguments shown below are used to specify the underlying tract grid.[1]

In the first case, for example, we might have...

[1] Please note that because the first argument is missing, you must either specify all arguments by name as in the example, or alternatively explicitly list the missing signature `object` as: `Tract(,0.5, cd, c(-5,-5))` to invoke the constructor.

```
R> cd = c(x=200, y=200)
R> tr = Tract(cellSize = 0.5,
+             cellDims = cd,
+             cellCenter = c(-5,-5),
+             data = runif(prod(cd)),
+             description = 'a simple example tract')
R> validObject(tr)


[1] TRUE


R> tr


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

class       : Tract
dimensions  : 200, 200, 40000  (nrow, ncol, ncell)
resolution  : 0.5, 0.5  (x, y)
extent      : -5.25, 94.75, -5.25, 94.75  (xmin, xmax, ymin, ymax)
coord. ref. : NA
data source : in memory
names       : surf
values      : 1.1077384e-05, 0.99998033  (min, max)


R> is(tr, 'RasterLayer')


[1] TRUE


R> summary(tr)


Object of class: Tract
------------------------------------------------------------
a simple example tract
------------------------------------------------------------
```

```
Min.    1.1077e-05
1st Qu. 2.5028e-01
Median  5.0169e-01
Mean    4.9984e-01
3rd Qu. 7.4827e-01
Max.    9.9998e-01
```

Here we have created a 200 by 200 cell raster grid with resolutoin of 0.5 meters (i.e., 1 hectare), origin at $(-5, -5)$ meters. The data values for all cells are initialized to zero by default. However, here we have shown how you can initialize them with a vector of correct length, in this case simply random uniform values.

It is trivial to plot the object, and there is not much to show in this case, but it's more interesting than a zero-valued surface at least...

```
R> plot(tr)
```

Note that the origin is expressed in terms of the center of the cell that holds it. You can see this in the above by noting the $x$,$y$ extents in comparison to the `cellCenter` argument specified. This particular arrangement for the constructor actually derives from the `sp` package and the construction of "SpatialGridDataFrame" objects[2].

### 2.2.2 Maximum extents constructor

In the second constructor the signature argument should be a numeric vector of length two with names "`x`" and "`y`." These specify the overall *maximum* limits in these coordinates, the minimum is always assumed to be $(0, 0)$. The `cellSize` argument is also required.

```
R> tr2 = Tract(c(x=20, y=20), cellSize=0.25)
R> validObject(tr2)
```

```
[1] TRUE
```

### 2.2.3 Bounding box constructor

The third constructor uses a valid bounding box to set the tract limits. The signature argument is therefore of class "matrix" and must conform to a `bbox` with row names "`x`" and "`y`," and column

[2] This constructor was used in prototype code used in Gove and Van Deusen (2011), it has been kept here.

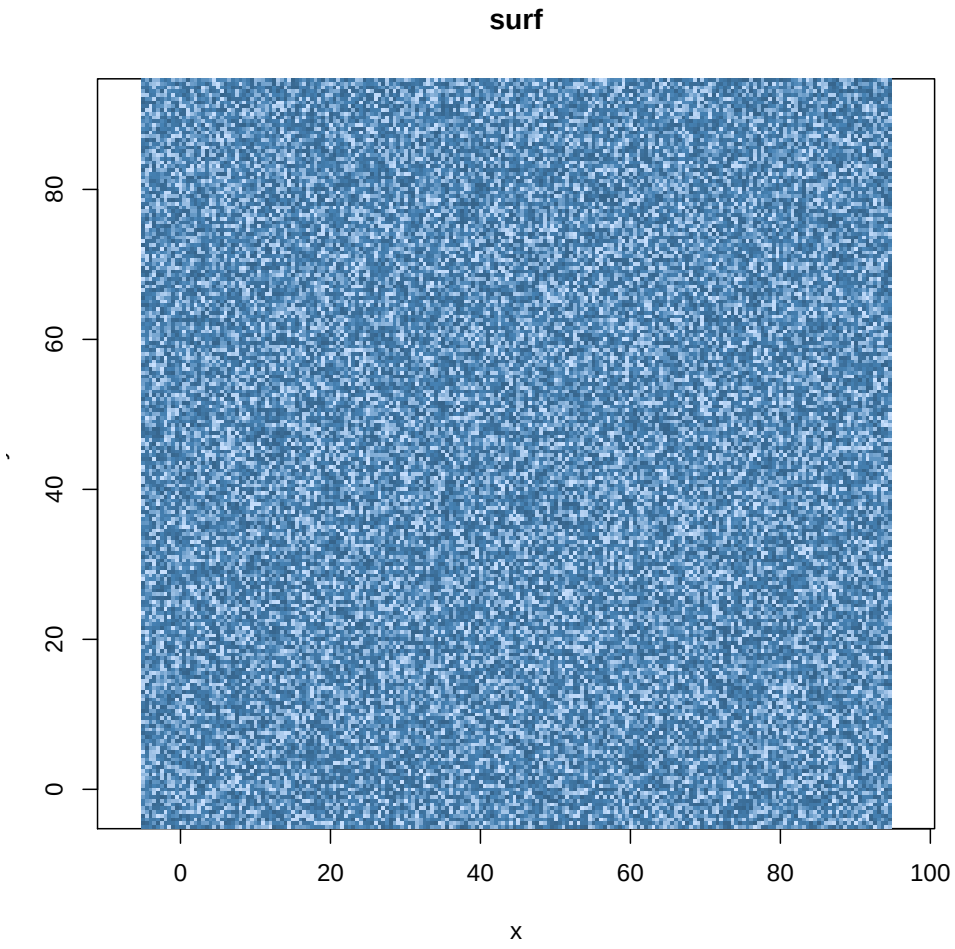


Figure 2: Simple generic "Tract" object.

names "`min`" and "`max`." It must also be a $2 \times 2$ matrix with minima less than the maxima. One can set the minima to something other than $(0, 0)$ with this constructor.

```
R> bb = bbox(tr2)
R> bb[,1] = c(2,3)
R> bb
```

```
  min max
x   2  20
y   3  20
```

```
R> tr3 = Tract(bb, cellSize=0.5)
```

### 2.2.4 RasterLayer constructor

The final "Tract" constructor has signature argument of class "RasterLayer" from the `raster` package. This allows one to directly generate a tract from a given "RasterLayer" object. There are several constructors for such objects that are documented in the `raster` package help files. Suffice it to say that this should add all the flexibility required for making objects of class "Tract".

```
R> ex = extent(tr3)
R> rl = raster(ex, nrows=34, ncols=36, crs=NA)
R> tr4 = Tract(rl)
R> tr4
```

```
------------------------------------------------------------
object of class Tract
------------------------------------------------------------
Measurement units = metric
Area in square meters = 306 (0.0306 hectares)

class       : Tract
dimensions  : 34, 36, 1224  (nrow, ncol, ncell)
resolution  : 0.5, 0.5  (x, y)
extent      : 2, 20, 3, 20  (xmin, xmax, ymin, ymax)
coord. ref. : NA
```

This is the only constructor that does not utilize the first, so it does not echo the "Grid Topology" since the raster object is already created—we simply add to it in this constructor for a valid "Tract" object.

Note that the `raster` constructor will allow one to make rectangular grid cells. This is not allowed in `sampSurf` and will cause an error if tried in the above constructor; only square grid cells are allowed in `sampSurf`.

## 3 The "bufferedTract" Class

The "bufferedTract" class simply adds a buffer to the tract. The buffer can be thought of most commonly as being internal; that is, it specifies an area within which the individual "Stem" object locations (tree or log centers) should fall in order to contain all of the inclusion zones. In other words, we are assuming that we will be placing logs randomly within this buffer so that there is no eventual slopover of the inclusion zones at the boundary. But beyond this, the buffer can be any width, it is up to the user.

Here's a look at the class structure, class "bufferedTract" is seen to be a subclass of "Tract"...

```
R> getClass('bufferedTract')
```

```
Class "bufferedTract" [package "sampSurf"]

Slots:

Name:        bufferRect         spBuffer      description            units
Class:           matrix  SpatialPolygons        character        character

Name:              area             file             data           legend
Class:          numeric      .RasterFile .SingleLayerData    .RasterLegend

Name:           history            title           extent          rotated
Class:             list        character           Extent          logical

Name:          rotation            ncols            nrows              crs
Class:        .Rotation          integer          integer              CRS

Name:                 z
Class:             list

Extends:
Class "Tract", directly
Class "RasterLayer", by class "Tract", distance 2
Class "Raster", by class "Tract", distance 3
Class "BasicRaster", by class "Tract", distance 4
```

## 3.1 Class slots

The following slots are added to the "Tract" class definition...

- *bufferRect*: A matrix holding the buffer rectangle extents in the form of a bounding box that is used by the `sp` package.
- *spBuffer*: The buffer in polygon form as a "SpatialPolygons" object.

## 3.2 Object creation

A "bufferedTract" object can be created very simply from a "Tract" object by specifying a reasonable buffer width in the constructor. In the following we make a buffered tract and add several down logs whose center locations fall within the buffer rectangle...

```
R> tr = setValues(tr, rep(0, ncell(tr)))
R> btr = bufferedTract(10, tr)
R> dlogs = downLogs(25, btr)

R> plot(btr, bufferColor='navy', lty='dotted', gridColor='grey90')
R> plot(dlogs, add=TRUE)
```

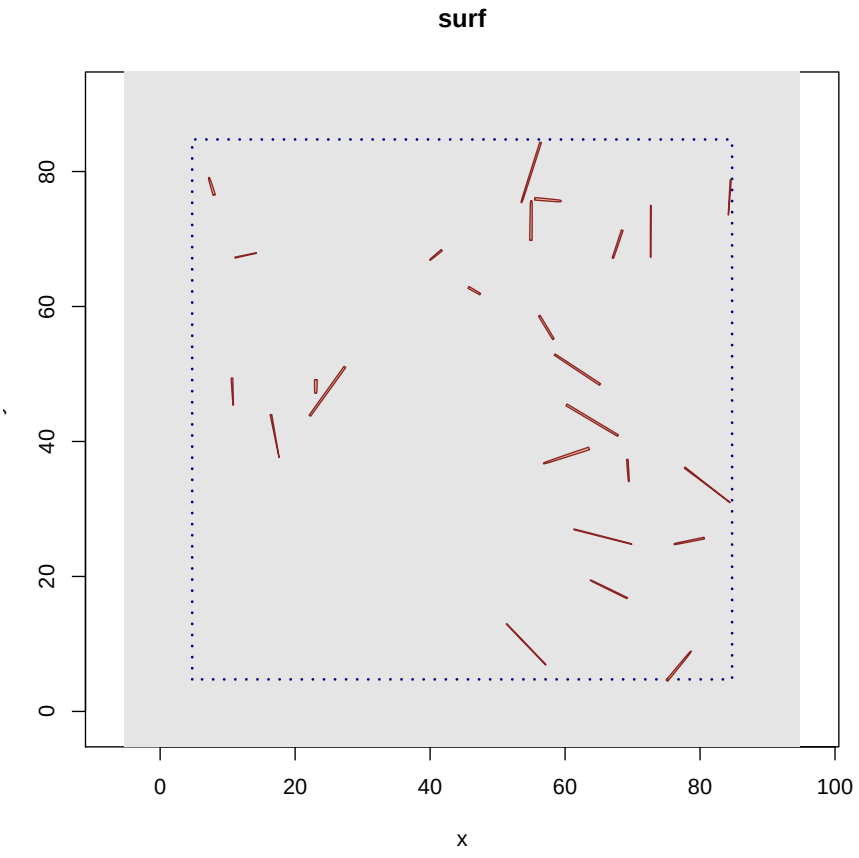


Figure 3: Simple generic "bufferedTract" object with some random logs.

Note in the above that we can make a collection of "downLogs" by simply specifying the number of logs and a "bufferedTract" object, which supplies the bounding box of the buffer area from which

to draw the logs.

## References

J. H. Gove and P. C. Van Deusen. On fixed-area plot sampling for downed coarse woody debris. *Forestry*, 84(2):109–117, 2011. 5