The "InclusionZone" 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

Monday 20th June, 2011
3:10pm

Contents

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

1 Introduction

Inclusion zones in general are the area surrounding an object in which the sample point (e.g., the fixed-radius plot center) can land and sample the object. Therefore, they have a clearly defined area and perimeter. In addition, for PPS methods, they depend on some attribute of the object, such as length in the case of a down log. This class and its subclasses bring together objects from the "ArealSampling" class and the "Stem" class, to define the more useful subclasses corresponding to actual sampling methods for standing trees or down logs.

The "InclusionZone" object that will be defined through its subclasses will eventually instantiate an object that therefore contains both the inclusion zone and the stem itself. These components will be stored in separate slots in the object, so they can be accessed individually as needed, but for plotting and summary purposes, the object will be treated as the union of the two component objects that define the inclusion zone and stem.

Figure 1 presents an overview of the class hierarchy arrangement. Round-cornered boxes represent virtual classes, while the "standingTreeIZ" class does not exist yet. A container class ("downLogIZs") for "downLogIZ" objects is also shown, it will handle collections of objects of any of the "downLogIZs" subclasses.

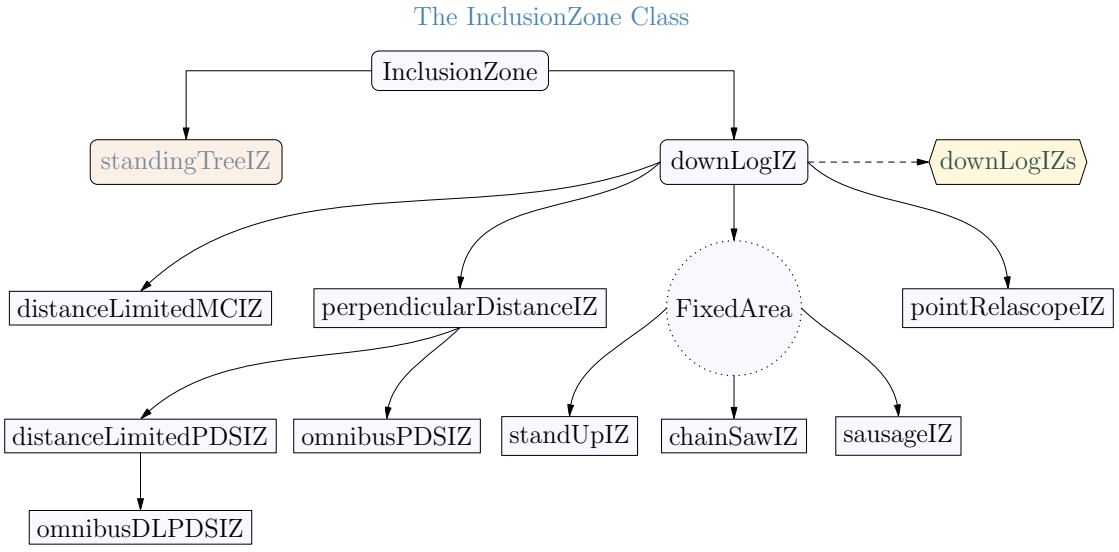

Figure 1: An overview of the "InclusionZone" class. "FixedArea" is not part of the class structure, but simply a subheading for the three subclasses following.

2 The "InclusionZone" Class

This is a virtual class which simply defines the main attributes of the subclasses. Note that this class is extremely basic, and it will have two direct subclasses, much like the "Stem" class, for standing tree- and down log-based sampling methods.

The base class is defined with the slots...

```
R> showClass('InclusionZone')
```

```
Virtual Class "InclusionZone" [package "sampSurf"]

Slots:

Name:   description        units         bbox      spUnits    puaBlowup
Class:    character    character       matrix          CRS      numeric

Name:  puaEstimates    userExtra
Class:         list          ANY

Known Subclasses:
Class "downLogIZ", directly
Class "standUpIZ", by class "downLogIZ", distance 2
Class "chainSawIZ", by class "downLogIZ", distance 2
Class "sausageIZ", by class "downLogIZ", distance 2
Class "pointRelascopeIZ", by class "downLogIZ", distance 2
Class "perpendicularDistanceIZ", by class "downLogIZ", distance 2
Class "omnibusPDSIZ", by class "perpendicularDistanceIZ", distance 3
Class "distanceLimitedIZ", by class "downLogIZ", distance 2
Class "distanceLimitedMCIZ", by class "distanceLimitedIZ", distance 3
Class "distanceLimitedPDSIZ", by class "perpendicularDistanceIZ", distance 3
Class "omnibusDLPDSIZ", by class "distanceLimitedPDSIZ", distance 4
```

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."
- *bbox*: The bounding box enclosing the overall object and its inclusion zone. Since the inclusion zone may not include the entire object (as in the "standup" method for down logs), we require

it here. Most of the time, it is actually redundant as the "SpatialPolygons" slot containing the inclusion zone in the subclasses will have this same information.

- *spUnits*: A valid string of class "CRS" denoting the spatial units coordinate system ("?CRS" for more information) as in package `sp`.
- *puaBlowup*: The per unit area blowup or expansion factor, based on an acre or hectare depending upon the units of measure.
- *puaEstimates*: A list of estimates for per unit area quantities associated with the "Stem" component of the object.
- *userExtra*: This can be anything else the user wants to associate with the object. Normally, it might be in the form of a `list` object, but can be anything. The user has complete control over this, it will not be used in any of the methods applied to the class, it is there for extra information storage as desired.

3 The "downLogIZ" Class

This is another small definitional step. It simply starts the bifurcation for subclasses that are related to "ArealSampling" classes for down logs. One slot is added in the process, that for a "downLog" object. Please note that this class is also *virtual*, and so just helps form the fact that its subclasses are "InclusionZone" classes for downed logs.

```
R> getClass('downLogIZ')
```

```
Virtual Class "downLogIZ" [package "sampSurf"]

Slots:

Name:       downLog  description        units         bbox      spUnits
Class:      downLog    character    character       matrix          CRS

Name:     puaBlowup puaEstimates    userExtra
Class:      numeric         list          ANY

Extends: "InclusionZone"

Known Subclasses:
Class "standUpIZ", directly
Class "chainSawIZ", directly
Class "sausageIZ", directly
```

```
Class "pointRelascopeIZ", directly
Class "perpendicularDistanceIZ", directly
Class "distanceLimitedIZ", directly
Class "omnibusPDSIZ", by class "perpendicularDistanceIZ", distance 2
Class "distanceLimitedMCIZ", by class "distanceLimitedIZ", distance 2
Class "distanceLimitedPDSIZ", by class "perpendicularDistanceIZ", distance 2
Class "omnibusDLPDSIZ", by class "distanceLimitedPDSIZ", distance 3
```

4 The "standUpIZ" Class

Gove and Van Deusen (2011) recognize three main protocols for sampling down logs with fixed-area plots. The so-called "standup" method is probably the most commonly used. Basically, the standup method includes the log if the center of the large-end of the log lands within the sample plot. Therefore, the inclusion zone can be envisioned as centered on that point of the log. The subclass is defined as...

```
R> getClass('standUpIZ')
```

```
Class "standUpIZ" [package "sampSurf"]

Slots:

Name:  circularPlot      downLog  description        units         bbox
Class: circularPlot      downLog    character    character       matrix

Name:       spUnits    puaBlowup puaEstimates    userExtra
Class:          CRS      numeric         list          ANY

Extends:
Class "downLogIZ", directly
Class "InclusionZone", by class "downLogIZ", distance 2
```

It is seen to be a direct extension of the "downLogIZ" class, having added a slot for the "circularPlot" object.

4.1 Object creation

The constructor for the subclass takes a "downLog" object and a fixed-plot radius, the "circularPlot" object is created from the latter using the same units as those in the "downLog" object...

```
R> dl = downLog(species='red maple', logLen=10, buttDiam=10,
+               topDiam=2, center=c(x=3,y=2), logAngle=pi/4,
+               units='English', vol2wgt=30.6, wgt2carbon=0.5)
R> su = standUpIZ(dl, 5)
R> summary(su)


Object of class: standUpIZ
------------------------------------------------------------
inclusion zone for "standup" method
------------------------------------------------------------

InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: 554.62315 per acre

  Object bounding box...
        min       max
x -5.5330166 6.5944595
y -6.5349045 5.5944595


downLog component estimates...
  Spatial ID: log:s9n42zx1
  Number of logs: 554.62315 per acre
  Volume: 1531.5143 cubic feet per acre
  Surface area: 9878.5009 square feet per acre
  Coverage area: 3142.8645 square feet per acre
  Length: 5546.2315 feet per acre
  Biomass (woody): 46864.337 per acre
  Carbon content: 23432.169 per acre

standUpIZ...
  use "summary" on the circularPlot slot for details


R> summary(su@circularPlot)


Object of class: circularPlot
------------------------------------------------------------
fixed area circular plot
------------------------------------------------------------

ArealSampling...
```

```
  units of measurement:  English

circularPlot...
  radius = 5 feet
  area = 78.539816 square feet (0.001803 acres)
  spatial units:  NA
  spatial ID: su:91sxzc67
  location (plot center)...
    x coord:  -0.53553391
    y coord:  -1.5355339
  Number of perimeter points: 101 (closed polygon)
```

In the above example, the summary shows the extents of the object. Note that summaries of the "downLog" and "circularPlot" slots can easily be generated as shown.

4.2 Plotting the object

The `plot` generic function has also been extended to be able to handle plotting of the objects of the "standUpIZ" class. The arguments are again detailed in the help page, but here is a simple example...

```
R> plot(su, axes=TRUE, showPlotCenter=TRUE, cex=2, showNeedle=TRUE)
```

A couple things are worth noting in Figure 2. The overall bounding box includes the two objects, the inclusion zone (fixed-area plot) and the log, as is also shown in the above summary. The center of the plot is correctly aligned with the center of the large-end of the log. In addition, everything is relative to the location of the exact center of the log as defined in the "downLog" class. The correct juxtaposition of the inclusion zone is shown with regard to the log's rotation angle (the `logAngle` slot in the object). Again this information is all self-contained within the object. For example, in this object, the log angle can be found simply as...

```
R> su@downLog@logAngle
```

```
[1] 0.78539816
```

```
R> identical(dl@logAngle, su@downLog@logAngle)
```

```
[1] TRUE
```

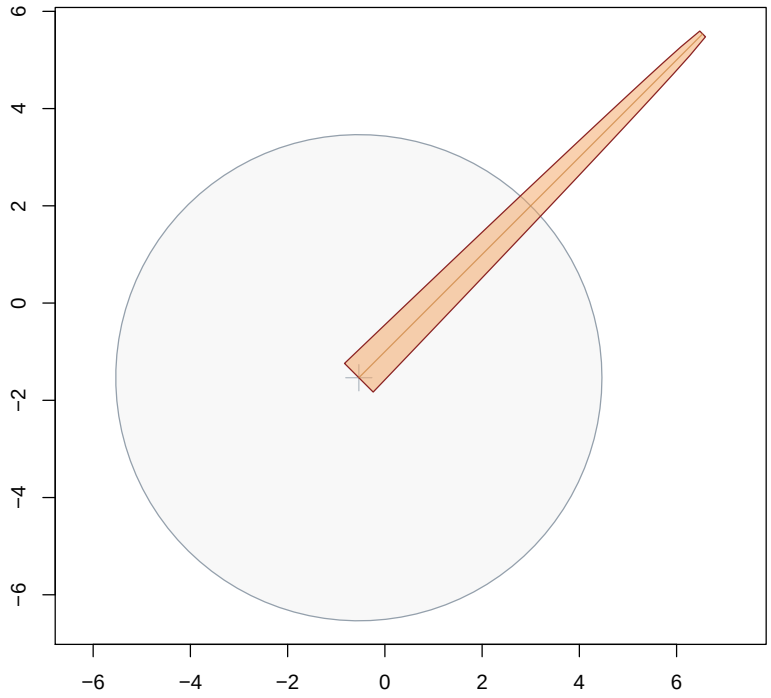

Figure 2: A "standupIZ" object.

5 The "chainSawIZ" Class

A second, but probably less commonly used protocol is what Gove and Van Deusen (2011) call the "chainsaw" method. With this method, the intersection of the plot with the log defines a 'sliver' of the log which is measured for volume. They discuss three main sub-protocols for the implementation of this method. This class essentially allows implementation of all three, tough only the sausage-based protocol is currently implemented as it is the most useful. The class extends "downLog" directly...

```
R> getClass('chainSawIZ')
```

```
Class "chainSawIZ" [package "sampSurf"]

Slots:
```

```
Name:     circularPlot          sliver            bolt         downLog
Class:    circularPlot SpatialPolygons            list         downLog

Name:      description           units            bbox         spUnits
Class:       character       character          matrix             CRS

Name:        puaBlowup    puaEstimates       userExtra
Class:         numeric            list             ANY

Extends:
Class "downLogIZ", directly
Class "InclusionZone", by class "downLogIZ", distance 2
```

wherein the following slots have been added...

5.1 Class slots

- *circularPlot*: An object of that class.
- *sliver*: A "SpatialPolygons" object representing the intersection (sliver) between the circular plot and the log.
- *bolt*: A `list` defining the 'minimal bounding bolt' within the log that fully encompases the sliver. The slots in the `list` are defined as[1]...
  - *rotBolt*: A matrix representation of the minimal bounding bolt.
  - *boltVol*: The bolt's volume.
  - *sectVol*: The volume of the sliver section.
  - *area*: A vector containing the proportion of the bolt that the sliver encompasses, the bolt polygon area, and the sliver polygon area.
  - *boltLen*: The length of the bolt at its longest point, which is also the sliver section length.
  - *boltSA*: The bolt's surface area.
  - *sectSA*: The sliver's surface area.
  - *boltCA*: The bolt's coverage area.
  - *sectCA*: The sliver's coverage area.
  - *boltBms*: The bolt's biomass content (can be `NA`).
  - *sectBms*: The sliver's biomass content (can be `NA`).

[1] See the `chainsawSliver` function for more details.

- *boltCarbon*: The bolt's carbon content (can be `NA`).
- *sectCarbon*: The sliver's carbon content (can be `NA`).
- *spBolt*: A "SpatialPolygons" object representing the minimal bounding bolt within the log.

5.2 Object creation

The constructor for the subclass takes a "downLog" object, a fixed-plot radius, and a center point for the fixed-radius plot...

```
R> cs = chainSawIZ(dl, 2, c(x=7.8, y=3.9), runQuiet=FALSE)
```

```
Percentage sliver is of bolt area = 18.075158
Bolt volume (not expanded) = 0.019930252
Section/sliver volume (not expanded) = 0.0036024245
```

```
R> summary(cs)
```

```
Object of class: chainSawIZ
------------------------------------------------------------
inclusion zone for "chainsaw" method
------------------------------------------------------------

InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: 3466.3947 per acre

  Object bounding box...
         min       max
x -0.83016173 9.8000000
y -1.83016173 5.8997483


downLog component estimates...
  Spatial ID: log:s9n42zx1
  Number of logs: 3466.3947 per acre
  Volume: 12.487425 cubic feet per acre
  Surface area: 427.35022 square feet per acre
  Coverage area: 135.60665 square feet per acre
```

```
  Length: 3051.8683 feet per acre
  Biomass (woody): 382.11521 per acre
  Carbon content: 191.05761 per acre
  The above estimates are based on the expanded sliver portion.

  The following are unexpanded...
    Sliver area is 0.18075158 of bounding bolt
    Total log volume: 2.7613602 cubic feet
      Bounding bolt volume: 0.019930252 cubic feet
      Sliver volume: 0.0036024245 cubic feet
    Total log surface area: 17.811195 square feet
      Bounding bolt surface area: 0.68206196 square feet
      Sliver surface area: 0.12328377 square feet
    Total log coverage area: 5.6666667 square feet
      Bounding bolt coverage area: 0.2164317 square feet
      Sliver coverage area: 0.039120371 square feet
    Total log length: 10 feet
      Bounding bolt length: 0.88041571 feet
    Total log biomass: 84.497622
      Bounding bolt biomass: 0.60986572
      Sliver biomass: 0.11023419
    Total log carbon: 42.248811
      Bounding bolt carbon: 0.30493286
      Sliver carbon: 0.055117095

chainSawIZ...
  use "summary" on the circularPlot slot for sample plot details
```

The `summary` method shows a bit more information about the resulting object than with other protocols. Most importantly, it shows information about the sliver size and volume, etc., and the minimal bounding bolt's size with respect to the entire log. We can see from this example, that we truly must have just cut a little sliver off, since its volume is so small. However, in the next example, we take a good chunk of the log in the plot intersection...

```
R> cs2 = chainSawIZ(dl, plotRadius=3, c(x=2, y=2), runQuiet=FALSE)
```

```
Percentage sliver is of bolt area = 97.811679
Bolt volume (not expanded) = 1.5331178
Section/sliver volume (not expanded) = 1.4995682
```

The constructor was requested to print the necessary information for comparison, and it will be verified by the graphics below.

5.3 Plotting the object

The `plot` generic function has also been extended to be able to handle plotting of the objects of the "chainSawIZ" class. The arguments are again detailed in the help page, but here is a simple example...

```
R> plot(cs, axes=TRUE, showNeedle=TRUE, showPlotCenter=TRUE)
R> plot(cs2, add=TRUE, showPlotCenter=TRUE, showLog=FALSE, izColor=NA)
```

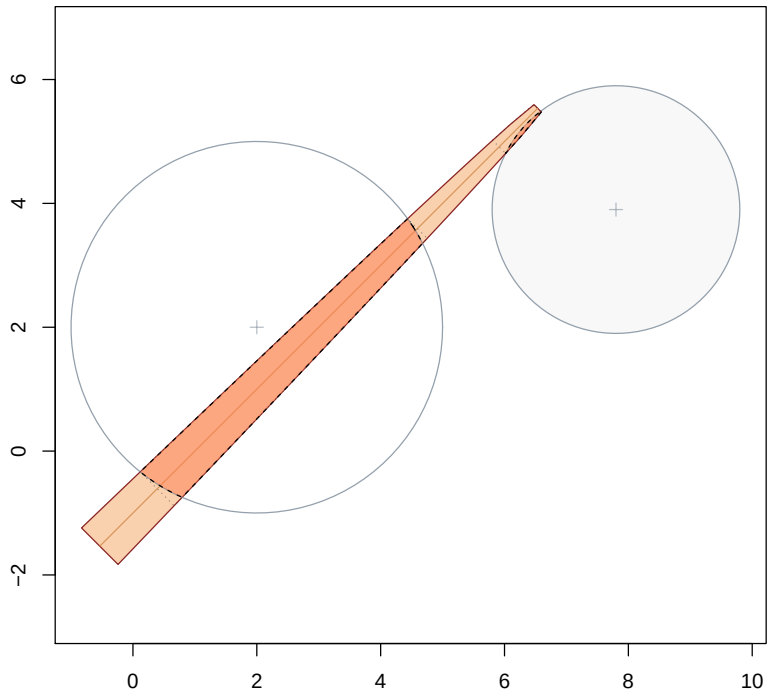

Figure 3: "chainsawIZ" objects comparison.

Figure 3 presents a comparison of the two "chainSawIZ" objects created above, using the same log for reference. As mentioned above, the second object takes a much larger slice of the log for an estimate of volume than the first. It is also worth noting that the first example shows an intersection where the log's 'needle' is not encountered. This would therefore not qualify as falling into the sausage-shaped inclusion zone for the overall log, unless it also intersected the end of the log (see Gove and Van Deusen (2011) for details). In the Figure, the minimal bounding bolts are

delineated by the dotted lines, and the slivers are clearly seen in the intersection, also shown as dashed.

6 The "sausageIZ" Class

Another method, which allows the log to be sampled if any part of it's "needle" falls inside the circular fixed-area plot is called the "sausage" method by Gove and Van Deusen (2011). It is a direct subclass of "downLogIZ"...

```
R> getClass('sausageIZ')
```

```
Class "sausageIZ" [package "sampSurf"]

Slots:

Name:          sausage           radius             area        perimeter
Class:          matrix          numeric          numeric SpatialPolygons

Name:    pgSausageArea          downLog      description            units
Class:         numeric          downLog        character        character

Name:             bbox          spUnits        puaBlowup    puaEstimates
Class:          matrix              CRS          numeric             list

Name:        userExtra
Class:             ANY

Extends:
Class "downLogIZ", directly
Class "InclusionZone", by class "downLogIZ", distance 2
```

wherein the following slots have been added...

6.1 Class slots

- *sausage*: A matrix representation of the sausage inclusion zone in homogeneous coordinates. This can be manipulated and plotted as desired for easy access to the inclusion zone where needed.

- *radius*: The radius for the fixed-area plot that determines the sausage inclusion zone area.
- *area*: The exact area of the inclusion zone.
- *perimeter*: This is the inclusion zone perimeter as a "SpatialPolygons" object.
- *pgSausageArea*: This is the area of the sausage inclusion zone as calculated from the polygon in the `perimeter` slot using the "SpatialPolygons" object. As such, it is an approximation of the true area of the inclusion zone, which is given in the `area` slot. This just enables us to see how close the graphic representation is to the real area.

6.2 Object creation

The constructor for the subclass takes a "downLog" object and a fixed-plot radius, much like the "standUpIZ" class. From this the inclusion zone is determined...

```
R> sa = sausageIZ(dl, 5)
R> summary(sa)
```

```
Object of class: sausageIZ
------------------------------------------------------------
inclusion zone for downed log "sausage" sampling method
------------------------------------------------------------

InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: 243.97919 per acre

  Object bounding box...
        min       max
x -5.5348916 11.535534
y -6.5348916 10.535534


downLog component estimates...
  Spatial ID: log:s9n42zx1
  Number of logs: 243.97919 per acre
  Volume: 673.71443 cubic feet per acre
  Surface area: 4345.5609 square feet per acre
  Coverage area: 1382.5488 square feet per acre
  Length: 2439.7919 feet per acre
  Biomass (woody): 20615.662 per acre
```

```
  Carbon content: 10307.831 per acre

sausageIZ...
  Spatial ID: saus:c4jd769a
  radius = 5 feet
  area = 178.53982 square feet (0.004099 acres)
  Number of perimeter points: 101 (closed polygon)
```

```
R> sa@pgSausageArea
```

```
[1] 178.48489
```

We note from the above that there is a little bias in the area of the inclusion zone when computed from the "SpatialPolygons" object. This is small, but shows the approximate nature of the inclusion zone perimeter. Using more points to define the half-circles on each end of the zone allows this area estimate to converge to the actual area. Using too few points obviously results in larger bias. This should be kept in mind when constructing grids for sampling surface simulations eventually with these objects, because the perimeter will determine what grid cell centers are within the inclusion zone, and hence get assigned a surface attribute value.

6.3 Plotting the object

The `plot` generic function has also been extended to be able to handle plotting of the objects of the "sausageIZ" class. The arguments are again detailed in the help page, but here is a simple example...

```
R> plot(sa, axes=TRUE, showLogCenter=TRUE, cex=2, showNeedle=TRUE)
R> dl2 = downLog(species='white pine', logLen=4, buttDiam=6,
+                topDiam=0, center=c(x=1,y=4), logAngle=3*pi/4, units='English'
+               )
R> sa2 = sausageIZ(dl2, 5)
R> plot(sa2, add=TRUE, izCol=NA)
R> plot(su@circularPlot@perimeter, lty='dashed', border='grey50', add=TRUE)
```

Figure 4 illustrates the sausage inclusion zones for two down logs. In the case of the longest log, the standup inclusion zone perimeter is shown for comparison.

One other thing to keep in mind under sausage sampling is that the log and inclusion zone centers coincide. Therefore, plotting the log center will identify them both using `showLogCenter=TRUE`;

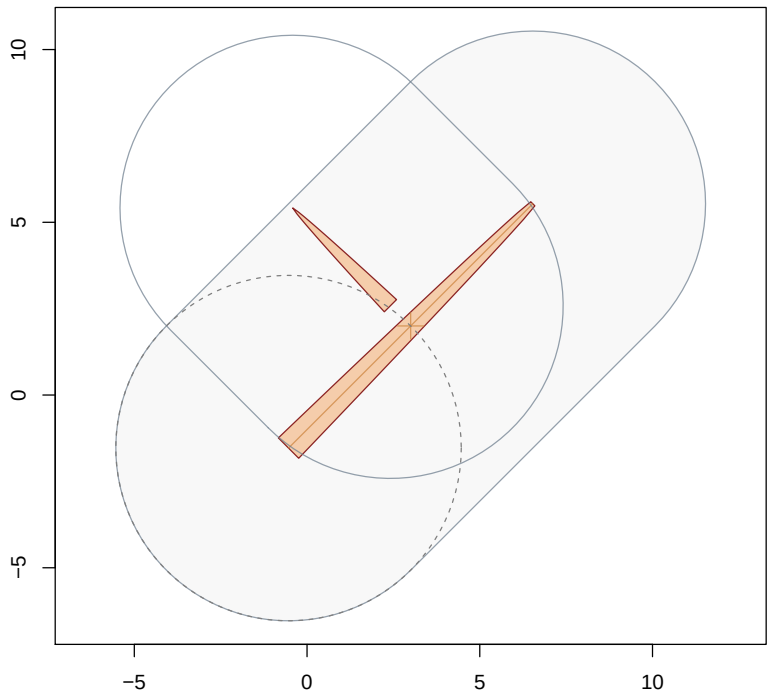

Figure 4: "sausageIZ" objects with "standUpIZ" perimeter for comparison.

using `showPlotCenter=TRUE` does nothing at present. If you really want to also display the plot center, it can always be accessed via the `downLog` slot in the object itself: the `location` slot in this object is an object of class "SpatialPoints" and therefore can be accessed as the inclusion zone plot center.

7 The "pointRelascopeIZ" Class

This class combines the usual information from an individual "downLog" object along with a given "pointRealscope" areal sampling object to create an inclusion zone for the point relascope sampling method (Gove et al. 1999, Gove et al. 2001). Recall that the inclusion area is proportional to the squared-length, so both objects are required in the class. It is a direct subclass of "downLogIZ"...

```
R> getClass('pointRelascopeIZ')
```

```
Class "pointRelascopeIZ" [package "sampSurf"]

Slots:

Name:              prs      dualCircle          radius            area
Class:  pointRelascope          matrix         numeric         numeric

Name:        perimeter      pgDualArea     dualCenters         downLog
Class: SpatialPolygons         numeric          matrix         downLog

Name:      description           units            bbox         spUnits
Class:       character       character          matrix             CRS

Name:        puaBlowup    puaEstimates       userExtra
Class:         numeric            list             ANY

Extends:
Class "downLogIZ", directly
Class "InclusionZone", by class "downLogIZ", distance 2
```

7.1 Class slots

The following slots have been added to this subclass...

- *prs*: An object of class "pointRelascope" that furnishes the information about the point relascope method.
- *dualCircle*: A matrix representation of the inclusion zone in homogensous coordinates. This can be manipulated and plotted as desired for easy access to the inclusion zone where needed.
- *radius*: The radius in the appropriate units of each of the dual circles comprising the inclusion zone.
- *area*: The exact area of the inclusion zone.
- *perimeter*: This is the inclusion zone perimeter as a "SpatialPolygons" object.
- *pgDualArea*: This is the area of the inclusion zone as calculated from the polygon in the `perimeter` slot using the "SpatialPolygons" object. As such, it is an approximation of the true area of the inclusion zone, which is given in the `area` slot. This just enables us to see how close the graphic representation is to the real area.
- *dualCenters*: The center points of the dual circles stored as a matrix (not in homogeneous form).

7.2 Object creation

The constructor for the subclass takes a "downLog" object and an object of class "pointRelascope". From this the inclusion zone is determined...

```
R> (rw.angle = .StemEnv$rad2Deg(2*atan(1/4)))
```

```
[1] 28.072487
```

```
R> prs4 = pointRelascope(rw.angle, units='English')
R> iz.prs = pointRelascopeIZ(dl, prs4)
R> iz.prs
```

```
Object of class: pointRelascopeIZ
------------------------------------------------------------
inclusion zone for down log point relascope sampling method
------------------------------------------------------------

InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: 62.908019 per acre

  Object bounding box...
         min       max
x -14.253728 20.253728
y -15.253728 19.253728


downLog component estimates...
  Spatial ID: log:s9n42zx1
  Number of logs: 62.908019 per acre
  Volume: 173.7117 cubic feet per acre
  Surface area: 1120.467 square feet per acre
  Coverage area: 356.47878 square feet per acre
  Length: 629.08019 feet per acre
  Biomass (woody): 5315.5781 per acre
  Carbon content: 2657.789 per acre

pointRelascopeIZ...
  Spatial ID: prs:g76pu0h4
  dual circle radius = 10.625 feet
```

```
  area = 692.43954 square feet (0.0159 acres)
  Number of perimeter points: 201 (closed polygon)
```

```
R> iz.prs@pgDualArea
```

```
[1] 692.15325
```

Again, as in the sausage method, we have relatively good agreement in this example betweem the actual area of the inclusion zone and that calculated from the "SpatialPolygons" object. And again, the fact that the latter can be adjusted through the number of points in each dual circle should be kept in mind when running simulations (see `?pointRelascopeIZ`). This is a very small angle, and draws in logs from far away. Likewise, it will encompass a good number of grid cells in the final sampling surface simulation; therefore, it makes sense to have the polygon and true area agree well, so we are not excluding grid cells. If we try to cut the number of points in each circle down we see that the area does not agree as well...

```
R> iz.prs = pointRelascopeIZ(dl, prs4, nptsCircle=50)
R> c(iz.prs@area, iz.prs@pgDualArea)
```

```
[1] 692.43954 691.27141
```

This is something to be kept in mind, the smaller relascope angle, the more points are required on the perimeter, or some bias from the approximation to the inclusion zone polygon could show up in the simulations.

7.3 Plotting the object

The `plot` generic function has also been extended to be able to handle plotting of the objects of the "pointRelascopeIZ" class. The arguments are again detailed in the help page (`methods?plot`), a simple example follows...

```
R> plot(iz.prs, axes=TRUE, showDualCenters=TRUE, showLogCenter=TRUE,
+      cex=2, showNeedle=TRUE)
R> (rw.angle = .StemEnv$rad2Deg(2*atan(1/2)))
```

```
[1] 53.130102
```

```
R> prs2 = pointRelascope(rw.angle, units='English')
R> iz.prs = pointRelascopeIZ(dl, prs2)
R> plot(perimeter(iz.prs), add=TRUE, lty='dashed', border=.StemEnv$izBorderColor)
```

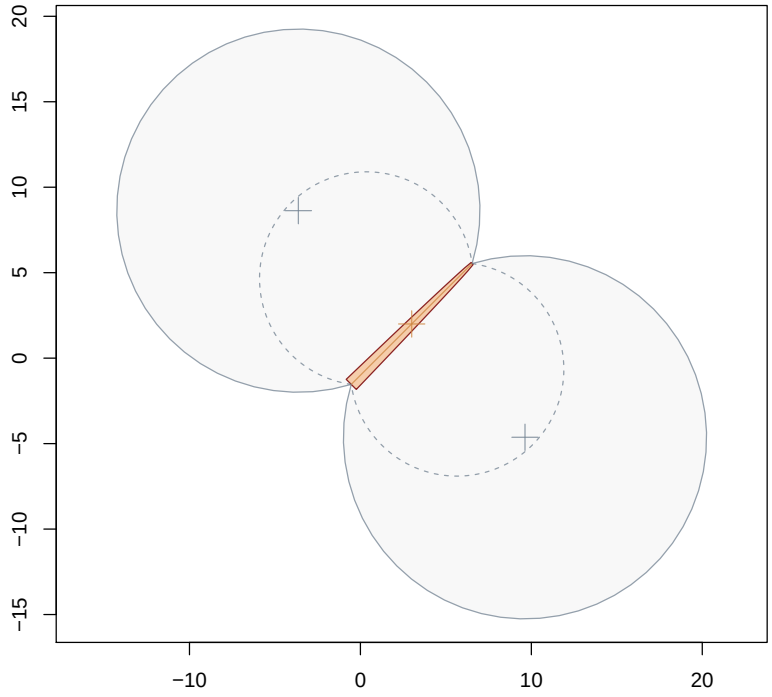

Figure 5: "pointRelascopeIZ" objects with different angles for comparison. Also shown are the dual circle centers for the smaller angle (larger inclusion zone).

8 The "perpendicularDistanceIZ" Class

This class combines the usual information from an individual "downLog" object along with a given "perpendicularDistance" areal sampling object to create an inclusion zone for the perpendicular distance sampling method (Williams and Gove 2003, Williams et al. 2005, Ducey et al. 2008). Recall that the inclusion area is proportional to some function of diameter along the entire length, so both objects are required in the class. It is a direct subclass of "downLogIZ"...

```
R> getClass('perpendicularDistanceIZ')
```

```
Class "perpendicularDistanceIZ" [package "sampSurf"]

Slots:

Name:                    pds               izPerim                  area
Class: perpendicularDistance                matrix               numeric

Name:              perimeter                 pgArea               pdsType
Class:       SpatialPolygons               numeric             character

Name:                downLog           description                 units
Class:               downLog             character             character

Name:                   bbox               spUnits             puaBlowup
Class:                matrix                   CRS               numeric

Name:           puaEstimates             userExtra
Class:                  list                   ANY

Extends:
Class "downLogIZ", directly
Class "pdsIZNull", directly
Class "InclusionZone", by class "downLogIZ", distance 2

Known Subclasses:
Class "omnibusPDSIZ", directly
Class "distanceLimitedPDSIZ", directly
Class "omnibusDLPDSIZ", by class "distanceLimitedPDSIZ", distance 2
```

8.1 Class slots

The following slots have been added to this subclass...

- *pds*: An object of class "perpendicularDistance" that furnishes the information about the perpendicular distance method.
- *izPerim*: A matrix representation of the inclusion zone in homogensous coordinates. This can be manipulated and plotted as desired for easy access to the inclusion zone where needed.
- *area*: The exact area of the inclusion zone.
- *perimeter*: This is the inclusion zone perimeter as a "SpatialPolygons" object.

- *pgArea*: This is the area of the inclusion zone as calculated from the polygon in the `perimeter` slot using the "SpatialPolygons" object. As such, it is an approximation of the true area of the inclusion zone, which is given in the `area` slot. This just enables us to see how close the graphic representation is to the real area.
- *pdsType*: Specifies the type of perpendicular distance sampling used: volume, surface or coverage area.

8.2 Object creation

As noted above, the `pdsType` slot stores the type of PDS sample we have used in the sense of sampling with probability proportional to either volume, surface area or coverage area. In what follows, we demonstrate only sampling with PP to volume, which is the default. However, if one would like to use either of the other two protocols, simply include the `pdsType` argument in the constructor, with one of the following as the specification to this argument...

```
R> .StemEnv$pdsTypes
```

```
[1] "volume"       "surfaceArea"  "coverageArea"
```

The constructor for the subclass takes a "downLog" object and an object of class "perpendicularDistance". From this the inclusion zone is determined...

```
R> pdsEng = perpendicularDistance(kpds=10, units='English')
R> iz.pdsv = perpendicularDistanceIZ(dl, pdsEng)
R> c(iz.pdsv@area, iz.pdsv@pgArea)
```

```
[1] 55.227204 55.212106
```

```
R> iz.pdsv
```

```
Object of class: perpendicularDistanceIZ
------------------------------------------------------------
inclusion zone for down log perpendicular distance sampling
------------------------------------------------------------

InclusionZone...
```

```
  Units of measurement:  English
  Per unit area blowup factor: 788.74172 per acre

  Object bounding box...
        min       max
x -4.3922031 6.6898007
y -5.3922031 5.6898007


downLog component estimates...
  Spatial ID: log:s9n42zx1
  Number of logs: 788.74172 per acre
  Volume: 2178 cubic feet per acre
  Surface area: 14048.432 square feet per acre
  Coverage area: 4469.5364 square feet per acre
  Length: 7887.4172 feet per acre
  Biomass (woody): 66646.8 per acre
  Carbon content: 33323.4 per acre

perpendicularDistanceIZ...
  PDS type: volume
  Spatial ID: pds:0egy4d15
  area = 55.227204 square feet (0.001268 acres)
  Number of perimeter points: 43 (closed polygon)
```

First, we create an "ArealSampling" object of class "perpendicularDistance", and subsequently create the inclusion zone object for the respective log. The area comparison between the theoretically correct area of the inclusion zone and that based on the polygon approximation is shown next. Surprisingly, the polygon approximation is quite good as it is based on the taper data in the "downLog" object. This is something to be kept in mind for simulations, if too few points are used to determine the taper, the inclusion zone can be poorly determined because it uses these taper points directly. So if a set of simulations produces some measureable bias, poor taper data restricting the underlying grid cells in the final sampling surface may be the cause under PDS. In the above example the taper slot in the log had 21 taper measurements, so the inclusion zone polygon has...

```
R> nrow(dl@taper)*2+1
```

```
[1] 43
```

perimeter points in the closed polygon (first and last point are repeated).

8.3 Plotting the object

The `plot` generic function has also been extended to be able to handle plotting of the objects of the "perpendicularDistanceIZ" class. The arguments are again detailed in the help page (`methods?plot`), a simple example follows...

```
R> plot(iz.pdsv, axes=TRUE, showLogCenter=TRUE, cex=2, showNeedle=TRUE)
R> pdsEng2 = perpendicularDistance(kpds=5, units='English')
R> iz.pdsv2 = perpendicularDistanceIZ(dl, pdsEng2)
R> plot(perimeter(iz.pdsv2), add=TRUE, lty='dashed', border=.StemEnv$izBorderColor)
```

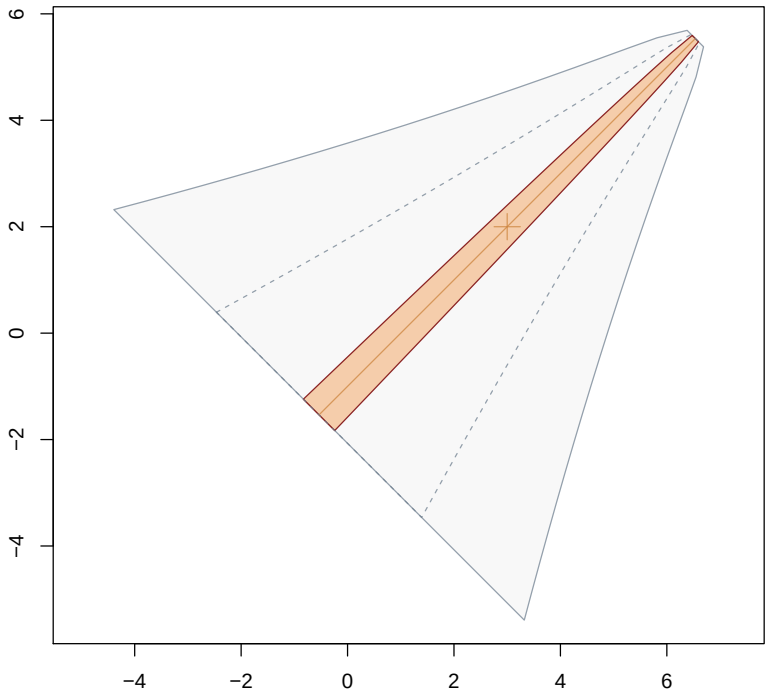

Figure 6: "perpendicularDistanceIZ" objects with different K PDS values for comparison.

8.4 Omnibus PDS

Omnibus PDS is described by Ducey et al. (2008). It allows us to get estimates of attributes other than the one of the three primary selection attibutes of volume, surface area or coverage area. The "omnibusPDSIZ" class definition is *exactly* the same as the "perpendicularDistanceIZ" class. In fact it is a subclass of the latter that adds no new slots, so we can refer directly to the information in the last section for the class definition.

8.4.1 Object creation

Object creation is also the same for omnibus. This is because the inclusion zone is exactly the same. However, the eventual sampling surface inside the inclusion zone will be variable in height, so there is no one overall estimate of the quantities of interest that can be calculated. This is because the estimates for each point within the inclusion zone rely on some function of diameter at the perpendicular. More will be covered on this in "*The InclusionZoneGrid Class*" vignette.

```
R> iz.opdsv = omnibusPDSIZ(dl, pdsEng)
R> iz.opdsv
```

```
Object of class: omnibusPDSIZ
------------------------------------------------------------
inclusion zone for down log omnibus perpendicular distance sampling
------------------------------------------------------------

InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: 788.74172 per acre

  Object bounding box...
         min       max
x -4.3922031 6.6898007
y -5.3922031 5.6898007


downLog component estimates...
  Spatial ID: log:s9n42zx1
  Number of logs: NA per acre
  Volume: NA cubic feet per acre
  Surface area: NA square feet per acre
  Coverage area: NA square feet per acre
  Length: NA feet per acre
```

```
  Biomass (woody): NA per acre
  Carbon content: NA per acre

omnibusPDSIZ...
  PDS type: volume
  Spatial ID: opds:u1t08qp2
  area = 55.227204 square feet (0.001268 acres)
  Number of perimeter points: 43 (closed polygon)
```

Note particularly that the "downLog" component estimates are all `NA` reflecting the above point concerning the variable sampling surface within the inclusion zone.

8.4.2 Plotting the object

A plot of this object would look *exactly* like the wide zone in Figure 6, so there is no need to repeat that here.

9 The "distanceLimitedIZ" Class

Distance limited sampling comes in two forms. The first (DLS) applies normal field protocols under this method. The second, distance limited Monte Carlo Sampling (DLMCS), uses crude Monte Carlo for the field protocol (Gove et al., 2012). The inclusion zones are exactly the same for each protocol, it is only the field measurements that differ—much like the two "flavors" of PDS and distance limited PDS presented below. Therefore, here we simply show how to create inclusion zones and what they look like. Basically, DLS is a PP to length sampling method that invokes a perpendicular distance limit, creating a rectangular inclusion zone that has the same length as the log. It is also relevant because it plays a part in distance limited PDS (both canonical and omnibus), which will be discussed in the next section.

```
R> getClass('distanceLimitedIZ')
```

```
Class "distanceLimitedIZ" [package "sampSurf"]

Slots:

Name:              dls         izPerim            area       perimeter
Class: distanceLimited          matrix         numeric SpatialPolygons
```

```
Name:           pgArea         downLog     description           units
Class:         numeric         downLog       character       character

Name:             bbox         spUnits       puaBlowup    puaEstimates
Class:          matrix             CRS         numeric            list

Name:        userExtra
Class:             ANY

Extends:
Class "downLogIZ", directly
Class "dlsIZNull", directly
Class "InclusionZone", by class "downLogIZ", distance 2

Known Subclasses: "distanceLimitedMCIZ"
```

9.1 Class slots

The following slots have been added to this subclass...

- *dls*: An object of class "distanceLimited" that furnishes the information about the distance limited "ArealSampling" method.
- *izPerim*: A matrix representation of the inclusion zone in homogensous coordinates. This can be manipulated and plotted as desired for easy access to the inclusion zone where needed.
- *area*: The exact area of the inclusion zone.
- *perimeter*: This is the inclusion zone perimeter as a "SpatialPolygons" object.
- *pgArea*: This is the area of the inclusion zone as calculated from the polygon in the `perimeter` slot using the "SpatialPolygons" object. As such, it is an approximation of the true area of the inclusion zone, which is given in the `area` slot. This just enables us to see how close the graphic representation is to the real area.

The "distanceLimitedMCIZ" class for DLMCS has exactly the same slots as above.

9.2 Object creation

Object creation is with its own constructor as usual and is the same for either field protocol, we use the DLMCS version as an example; to generate an object of class "distanceLimitedIZ", just use `distanceLimitedIZ` in place of the constructor below...

```
R> dlsEng = distanceLimited(6, units='English')
R> iz.dlmcs = distanceLimitedMCIZ(dl, dls=dlsEng)
R> iz.dlmcs


Object of class: distanceLimitedMCIZ
------------------------------------------------------------
inclusion zone for down log DLMC sampling
------------------------------------------------------------

InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: 3630 per acre

  Object bounding box...
         min        max
x -4.7781746 10.7781746
y -5.7781746  9.7781746


downLog component estimates...
  Spatial ID: log:s9n42zx1
  Number of logs: 363 per acre
  Volume: NA cubic feet per acre
  Surface area: NA square feet per acre
  Coverage area: NA square feet per acre
  Length: 3630 feet per acre
  Biomass (woody): NA per acre
  Carbon content: NA per acre
  (Note: NAs signify location-dependent attributes.)

distanceLimitedMCIZ...
  Spatial ID: dlmc:o196tu4g
  distance limit = 6 feet
  area = 120 square feet (0.002755 acres)
  Number of perimeter points: 5 (closed polygon)
```

Note that we do not need to explicitly specify the units as 'English' in the costructor function because it takes the units from the "downLog" and "distanceLimited" objects passed. Also note that most of the estimates are `NA`, reflecting the variable surface height for these attributes. This would not be the case under DLS (`distanceLimitedIZ`), where the usual constant estimates apply creating a flat surface (Gove et al., 2012).

9.3 Plotting the object

The `plot` generic function has also been extended to be able to handle plotting of the objects of the "distanceLimitedIZ" and "distanceLimitedMCIZ" classes. The arguments are again detailed in the help page (`methods?plot`), a simple example follows...

```
R> plot(iz.dlmcs, axes=TRUE, showLogCenter=TRUE, cex=2, showNeedle=TRUE)
R> dlsEng2 = distanceLimited(3, units='English')
R> iz.dlmcs2 = distanceLimitedMCIZ(dl, dls=dlsEng2)
R> plot(perimeter(iz.dlmcs2), add=TRUE, lty='dashed', border=.StemEnv$izBorderColor)
```

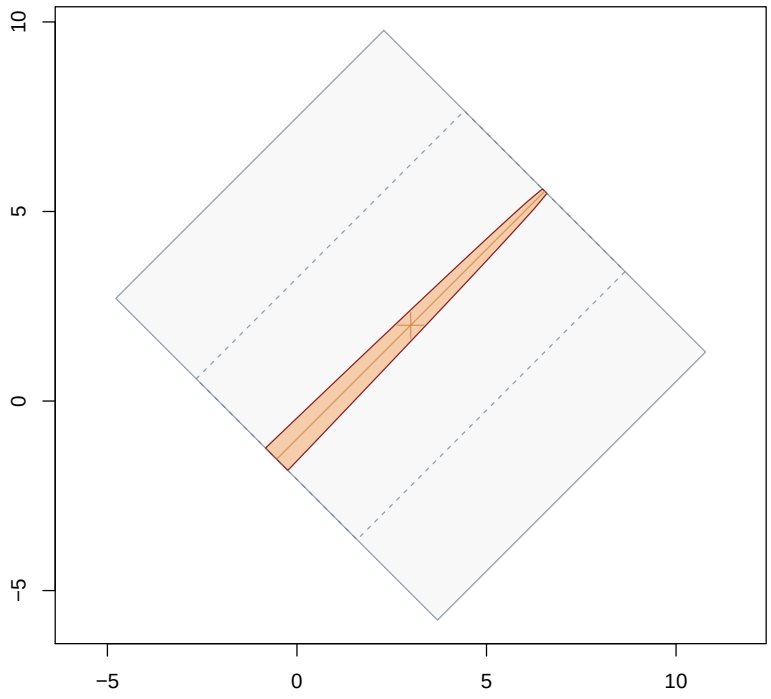

Figure 7: "distanceLimitedMCIZ" object.

Note that in Figure 7 we have plotted two inclusion zones of different widths for comparison. Again, the inclusion zones for DLSusing `distanceLimitedIZ`, would look exactly the same.

10 The "distanceLimitedPDSIZ" Class

Distance limited PDS (DLPDS) was initially developed by Ducey et al. (2005). It established a distance limit, beyond which we truncate the PDS inclusion zone, effectively limiting the search distance for logs of any size. This class combines the usual information from an individual "downLog" object along with "perpendicularDistance" and "distanceLimited" areal sampling objects, to create an inclusion zone. The inclusion zone thus may have both PDS and DLS components to it. Notice we say it *may* have components of both forms included, but does not have to, there are three possibilities: *(i)* the inclusion zone is all PDS, *(ii)* the inclusion zone is all DLS, or *(iii)* it is a combination of the two. Which of the three are true for a given log depends on the distance limit chosen and the size of the log, which will determine whether there is any limiting diameter separating the two sampling types.

```
R> showClass('distanceLimitedPDSIZ')
```

```
Class "distanceLimitedPDSIZ" [package "sampSurf"]

Slots:

Name:                   dls           dlsDiameter               pdsPart
Class:       distanceLimited               numeric             pdsIZNull

Name:               dlsPart               pdsFull                   pds
Class:            dlsIZNull             pdsIZNull perpendicularDistance

Name:               izPerim                  area             perimeter
Class:               matrix               numeric       SpatialPolygons

Name:               pgArea               pdsType               downLog
Class:              numeric             character               downLog

Name:          description                 units                  bbox
Class:           character             character                matrix

Name:               spUnits             puaBlowup          puaEstimates
Class:                  CRS               numeric                  list

Name:             userExtra
Class:                   ANY

Extends:
Class "perpendicularDistanceIZ", directly
```

```
Class "downLogIZ", by class "perpendicularDistanceIZ", distance 2
Class "pdsIZNull", by class "perpendicularDistanceIZ", distance 2
Class "InclusionZone", by class "perpendicularDistanceIZ", distance 3

Known Subclasses: "omnibusDLPDSIZ"
```

Note that this class is a direct subclass of "perpendicularDistanceIZ".

10.1 Class slots

The following slots have been added to this subclass...

- *dls*: An object of class "distanceLimited" that furnishes the information about the distance limited "ArealSampling" method.
- *dlsDiameter*: The limiting diameter where the delineation (if any) occurs between PDS and DLS protocols within the log (see below for an example and calculation).
- *pdsPart*: Either a fully valid "perpendicularDistanceIZ" (or subclass, i.e., "omnibusPDSIZ" object) or `NULL` if no component of this class exists for the log.
- *dlsPart*: Either a fully valid "distanceLimitedIZ" (or subclass, i.e., "distanceLimitedMCIZ" object) or `NULL` if no component of this class exists for the log.
- *pdsFull*: This holds a fully valid "perpendicularDistanceIZ" (or subclass, i.e., "omnibusPDSIZ" object) delineating the full PDS zone as it would appear if the entire log were treated as a PDS sampling object. Note that it can be useful to compare this to the DLPDS object graphically.
- *pds*: An object of class "perpendicularDistance" that furnishes the information about the perpendicular distance method.
- *izPerim*: A matrix representation of the inclusion zone in homogensous coordinates. This can be manipulated and plotted as desired for easy access to the inclusion zone where needed.
- *area*: The exact area of the inclusion zone.
- *perimeter*: This is the inclusion zone perimeter as a "SpatialPolygons" object.
- *pgArea*: This is the area of the inclusion zone as calculated from the polygon in the `perimeter` slot using the "SpatialPolygons" object. As such, it is an approximation of the true area of the inclusion zone, which is given in the `area` slot. This just enables us to see how close the graphic representation is to the real area.
- *pdsType*: Specifies the type of perpendicular distance sampling used: volume, surface or coverage area.

Now we can more fully look at the three component possibilities mentioned above with respect to the fields in the object...

(i) In the case where the inclusion zone is just a regular PDS object, we will have: `pdsPart` and `dlsPart` both `NULL`, and `pdsFull` a redundant copy of the full PDS portion of the object itself.

(ii) If the inclusion zone is all DLS, with no PDS component, then the overall object will 'act like' a "distanceLimitedIZ" object. The `pdsPart` slot will be `NULL`, while the `dlsPart` slot will carry a "distanceLimitedIZ" object and the `pdsFull` slot will again hold a full PDS inclusion zone class object (which is not applicable in this case, but is still retained for possible comparison, see next point).

(iii) Finally, if both components are involved in the log's inclusion zone, then the overall object contains the "hybrid" inclusion zone information. Both the `pdsPart` and `dlsPart` slots will contain the respective objects making up their components of the class, and the `pdsFull` slot will again hold a full PDS inclusion zone class object *as if* PDS were used on the entire log. In this case, for example, we could plot either the `pdsPart` or `dlsPart` objects to show the delineation between the two zones, and we could also plot the comparable PDS object for the whole log by plotting the object in the `pdsFull` slot.

It is important to realize in the above, that the slots containing the component parts (if they exist) are valid objects of the correct type in and of themselves. Thus, for example, they also contain the portion of the log that corresponds to the component part of the overall inclusion zone and will piece together to form the whole.

10.2 Object creation

Object creation is with its own constructor as usual, though note that we need both types of "ArealSampling" objects to define the two regions...

```
R> dlsEng3 = distanceLimited(2, units='English')
R> iz.dlpds = distanceLimitedPDSIZ(dl, pds=pdsEng2, dls=dlsEng3)
R> iz.dlpds
```

```
Object of class: distanceLimitedPDSIZ
------------------------------------------------------------
inclusion zone for down log distance limited PDS
------------------------------------------------------------
```

```
InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: NA per acre

  Object bounding box...
        min       max
x -1.9497475 6.6126673
y -2.9497475 5.6126673


downLog component estimates...
  Spatial ID: log:s9n42zx1
  Number of logs: 7053.8319 per acre
  Volume: 9496.6655 cubic feet per acre
  Surface area: 59010.92 square feet per acre
  Coverage area: 18775.168 square feet per acre
  Length: 31798.847 feet per acre
  Biomass (woody): 290597.96 per acre
  Carbon content: 145298.98 per acre

perpendicularDistanceIZ...
  PDS type: volume
  Spatial ID: dlpds:9qf83h4v
  area = 25.755564 square feet (0.0005913 acres)
  Number of perimeter points: 37 (closed polygon)
  (The above summary is for the entire DLPDS region)

distanceLimitedPDSIZ...
  distance limit = 2 feet
  limiting diameter = 0.71364965 feet (8.564 in)
  distance limited component available = TRUE
  PDS component available = TRUE
  Summaries of individual DLPDS components can be viewed seperately


R> class(iz.dlpds@pdsPart)


[1] "perpendicularDistanceIZ"
attr(,"package")
[1] "sampSurf"


R> class(iz.dlpds@dlsPart)
```

```
[1] "distanceLimitedIZ"
attr(,"package")
[1] "sampSurf"
```

The summary above shows the information for the overall object under the "perpendicularDistanceIZ" section of the report. Following this, in the "distanceLimitedPDSIZ" section, is more specific information about the current class' extension. For example, the limiting diameter is listed along with information indicating whether or not the `pdsPart` and `dlsPart` slots are available within the object.

Note that the limiting diameter is determined differently under each of the different PDS PPS variables; *viz.,* its formula in feet or meters for each is...

$$d_l^* = \sqrt{\frac{4 * D_l}{\pi K_{\text{PDS}}}} \qquad \text{PP volume}$$

$$d_l^* = \frac{D_l}{\pi K_{\text{PDS}}} \qquad \text{PP Surface Area}$$

$$d_l^* = \frac{D_l}{K_{\text{PDS}}} \qquad \text{PP Coverage Area}$$

where D_l is the distance limit in feet or meters. In the above example, the limiting diameter in inches is calculated as...

```
R> sqrt(4*dlsEng3@distanceLimit/(pi*pdsEng2@kpds))*12
```

```
[1] 8.5637958
```

which matches what was shown in the report above. Therefore, since the above diameter is between the butt (10 in) and top (2 in) diameters, both methods are present in the log. If we used a larger limiting distance then we would get a case where the entire log is sampled with PDS...

```
R> sqrt(4*dlsEng2@distanceLimit/(pi*pdsEng2@kpds))*12
```

```
[1] 10.488465
```

Likewise, the following shows the distance limit we would have to use in order for the entire log to be considered under only DLMCS...

```
R> iz.dlpds@downLog@topDiam^2 * pdsEng2@kpds*pi/4
```

```
[1] 0.10908308
```

10.3 Plotting the object

The `plot` generic function has also been extended to be able to handle plotting of the objects of the "distanceLimitedPDSIZ" class. The arguments are again detailed in the help page (`methods?plot`), a simple example follows...

```
R> plot(perimeter(iz.dlpds@pdsFull), axes=TRUE, lty='dotdash',
+       border=.StemEnv$izBorderColor)
R> plot(iz.dlpds, add=TRUE, showLogCenter=TRUE, cex=2, showNeedle=TRUE)
R> plot(perimeter(iz.dlpds@pdsPart), add=TRUE, lty='dashed',
+       border=.StemEnv$izBorderColor)
```

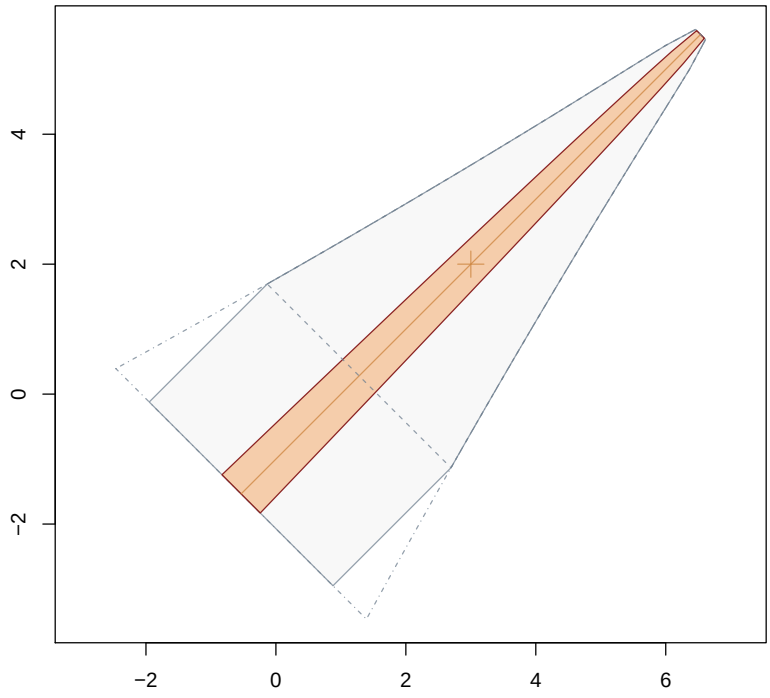

Figure 8: "distanceLimitedPDSIZ" object showing the PDS component delineation (dashed) and the full PDS inclusion zone (dot-dashed) for comparison.

10.4 The omnibus DLPDS

Omnibus DLPDS has not been written up yet. It allows us to get estimates of attributes other than the one of the three primary selection attibutes of volume, surface area or coverage area within the PDS component (if present) of a DLPDS inclusion zone. The "omnibusDLPDSIZ" class definition is *exactly* the same as the "distanceLimitedPDSIZ" class. In fact it is a subclass of the latter that adds no new slots, so we can refer directly to the information in the last section for the class definition.

The main difference between "distanceLimitedPDSIZ" and "omnibusDLPDSIZ" class objects is in the contents of the "Part" slots. The `pdsPart` slot will hold an object of class "omnibusPDSIZ", as will the `pdsFull` slot. In addition, the `dlsPart` slot will hold an object of class "distanceLimitedMCIZ" rather than "distanceLimitedIZ". As mentioned above, some of these slots can also be `NULL` if the component is not part of the inclusion zone for a given log.

10.4.1 Object creation

Object creation is also the same for omnibus. This is because the inclusion zone is exactly the same. However, the PDS component of the sampling surface inside the inclusion zone will be variable in height, so there is no one overall estimate of the quantities of interest that can be calculated—just as in "omnibusPDSIZ". This is because the estimates for each point within the inclusion zone rely on some function of diameter at the perpendicular. More will be covered on this in "*The InclusionZoneGrid Class*" vignette.

```
R> iz.odlpds = omnibusDLPDSIZ(dl, pds=pdsEng2, dls=dlsEng3)
R> iz.odlpds
```

```
Object of class: omnibusDLPDSIZ
------------------------------------------------------------
inclusion zone for down log omnibus distance limited PDS
------------------------------------------------------------

InclusionZone...
  Units of measurement:  English
  Per unit area blowup factor: NA per acre

  Object bounding box...
         min       max
x -1.9497475 6.6126673
y -2.9497475 5.6126673
```

```
downLog component estimates...
  Spatial ID: log:s9n42zx1
  Number of logs: NA per acre
  Volume: NA cubic feet per acre
  Surface area: NA square feet per acre
  Coverage area: NA square feet per acre
  Length: NA feet per acre
  Biomass (woody): NA per acre
  Carbon content: NA per acre

omnibusPDSIZ...
  PDS type: volume
  Spatial ID: odlpds:p8w053qo
  area = 25.755564 square feet (0.0005913 acres)
  Number of perimeter points: 37 (closed polygon)
  (The above summary is for the entire DLPDS region)

omnibusDLPDSIZ...
  distance limit = 2 feet
  limiting diameter = 0.71364965 feet (8.564 in)
  distance limited component available = TRUE
  PDS component available = TRUE
  Summaries of individual DLPDS components can be viewed seperately
```

```
R> class(iz.odlpds@pdsPart)
```

```
[1] "omnibusPDSIZ"
attr(,"package")
[1] "sampSurf"
```

```
R> class(iz.odlpds@dlsPart)
```

```
[1] "distanceLimitedMCIZ"
attr(,"package")
[1] "sampSurf"
```

Note particularly that the "downLog" component estimates are all `NA` reflecting the above point concerning the variable sampling surface within the inclusion zone. Also, compare the class types for the objects stored in the `dlsPart` and `pdsPart` listed above with that for the canonical DLPDS example above.

10.4.2 Plotting the object

A plot of this object would look *exactly* like that of Figure 8, so there is no need to repeat that here.

11 Container Classes

As we've mentioned in other vignettes, there needs to be a mechanism to have multiple versions of, e.g., "downLogIZ" objects or subclasses thereof stored in a population of sorts. This means there needs to be class and constructor methods, along with summary, plot, etc. One could just use a `list` structure, and this would work fine, but then there would be no ability to associate methods. So it is better to adopt an S4 class structure, albeit simple, that can encapsulate the behavior desired.

11.1 Class "izContainer"

This is a virtual base container class that may be extended for specific containers of different types.

```
R> showClass('izContainer')
```

```
Class "izContainer" [package "sampSurf"]

Slots:

Name:       iZones       units        bbox description
Class:        list   character      matrix   character

Known Subclasses: "downLogIZs"
```

At present, it holds a list of "InclusionZone" subclass objects in the `iZones` slot, and the overall bounding box for the collection or population, along with a description and units for all objects. The bounding box is useful in plotting and will also be used in the sampling surface routines.

This is not yet a fully functional 'container' class in the sense of *Java* or *C++*. For example, there is no functionality for adding, deleting, or replacing container objects at present. If you need to do any of this, it is best to just remake the collection (simple enough) with the new changes or you will need to worry about changing the bounding box as well.

11.2 Class "downLogIZs"

This container holds a collection of "downlogIZ" objects along with some spatial information. It is a usable direct descendant of the "izContainer" class...

```
R> showClass('downLogIZs')
```

```
Class "downLogIZs" [package "sampSurf"]

Slots:

Name:       iZones       units        bbox description
Class:        list   character      matrix   character

Extends: "izContainer"
```

11.2.1 Class construction

In keeping with the previous naming conventions, the constructor function for this class is `downLogIZs`. A collection can be created via the following steps, though there are alternatives in each case...

1. *Measured population*:
   (a) Use the field measurements to make individual "downLog" objects, and then place them into a `list` container.
   (b) Create valid "downLogIZ" subclass objects for each log—they all must be of the same class.
   (c) Use the "downLogIZs" constructor on this list of logs, to create the collection.
2. *Synthetic population*:
   (a) Draw a sample of "downLog"s randomly within some sample area, using the "downLogs" constructor.
   (b) Then create a `list` of "downLogIZ" subclass objects.
   (c) Finally, group these into the container class with "downLogIZs".

The above are essentially the same in practice, as similar steps are followed whether a synthetic or real population of logs is used. At present, one must specify objects of exactly the same subclass in

the `list` constructor. That is, you can not mix sampling methods as demonstrated below. Please see the "downLogs" constructor and *"The Stem Class"* vignette for more options in each of the first steps above.

In the first example, we will pretend that `dl` and `dl2` are logs from a field sample, and we then have made sausage inclusion zones for these in the example above. Therefore, we follow the rest of the steps...

```
R> sal = list(sa,sa2)
R> sapply(sal, class)
```

```
[1] "sausageIZ" "sausageIZ"
```

```
R> izs = downLogIZs(sal, description = 'two inclusion zones...\nboth sausage')
R> sapply(izs@iZones, class)
```

```
[1] "sausageIZ" "sausageIZ"
```

```
R> summary(izs)
```

```
Container object of class: downLogIZs
-------------------------------------------------------------
two inclusion zones...
both sausage
-------------------------------------------------------------
There are 2 inclusion zones in the population
Inclusion zones are of class: sausageIZ
Units of measurement:  English
Summary of inclusion zone areas in square feet...
       Min.     1st Qu.      Median        Mean     3rd Qu.        Max.
 118.540000  133.540000  148.540000  148.540000  163.540000  178.540000
        var        SDev
1800.000000   42.426407

Encapulating bounding box...
         min       max
x -5.5348916 11.535534
y -6.5348916 10.535534
```

The above example is fine, because both of the member objects in the list used to construct the "downLogIZs" object were of class "sausageIZ". However, in the following example, we are mixing sampling methods, which is not allowed...

```
R> sal2 = list(su,sa)
R> iizs = tryCatch(downLogIZs(sal2),  error = function(e) e)
R> if(is(iizs, 'simpleError')) cat(iizs[[1]],'\n')
```

```
invalid class "downLogIZs" object: You can not mix inclusion classes in the population!
```

```
R> sapply(sal2,class)
```

```
[1] "standUpIZ" "sausageIZ"
```

It is very simple to draw a simulated population of logs plus their inclusion zone. As shown below, this can be done for a log population of any size. Since the logs are stored within the objects as a list structure, we can apply the list-based object constructors via `lapply` to return a new list that can directly be converted to a container object...

```
R> dls = downLogs(6, units='English', buttDiam=c(6,20))
R> dls.su = lapply(dls@logs, 'standUpIZ', plotRadius=5)
R> izs.su = downLogIZs(dls.su)
R> dls.sa = lapply(dls@logs, 'sausageIZ', plotRadius=5)
R> izs.sa = downLogIZs(dls.sa)
```

In this case we have created two more container objects for a set of "downLog"s within a "downLogs" object, the first for the sausage method, and the second for the standup method.

11.3 Plotting the object

There is a plot method for "downLogIZs" objects that works as usual.

In the case of the first example, where the "downLogIZs" object was created using 'field data,' we have generate Figure 9...

```
R> plot(izs, axes=TRUE, showLogCenter=TRUE, cex=2, showNeedle=TRUE)
```

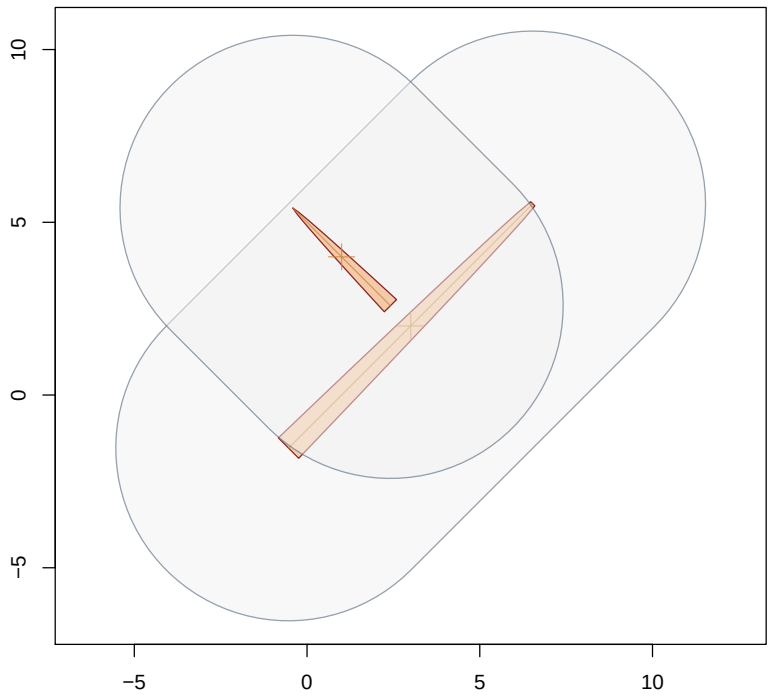

Figure 9: "downLogIZs" object with "sausageIZ" objects in the container.

In the second example, where we use synthetic logs, we generate Figure 10 showing the "standUpIZ" inclusion zones superimposed over the "sausageIZ" inclusion zones, purely for comparison...

```
R> plot(izs.sa, axes=TRUE, showLogCenter=TRUE, cex=2, showNeedle=TRUE)
R> plot(izs.su, add=TRUE, izColor=NA)
```

Note that in generating Figure 10, we have plotted the sausage collection before the standup collection, because the former has a larger overall bounding box that will contain all of the collected objects, regardless of their "InclusionZone" classes. Note again, however, the restriction that within a collection, all of the objects must share the same "InclusionZone" class.

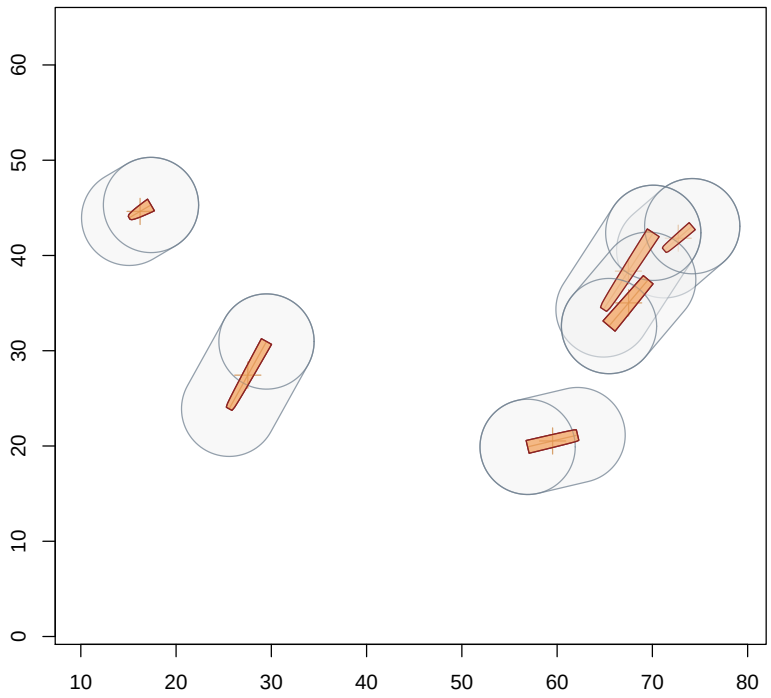

Figure 10: "downLogIZs" object of "sausageIZ" objects, with "standUpIZ" objects superimposed.

References

M. J. Ducey, M. S. Williams, S. Roberge, R. S. Kenning, and J. H. Gove. Distance limited perpendicular distance sampling for coarse woody material: Theory and field results. *Unpublished*, 2005. 30

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. 20, 25

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, 8, 12, 13

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. 16

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. 16

J. H. Gove, M. J. Ducey, and H. T. Valentine. A distance limited method for sampling downed coarse woody debris. *Journal of Applied Ecology*, 2012. (In preparation). 26, 28

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. 20

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. 20